

A Source Book from The Open Group

**The Single UNIX[®] Specification:
The Authorized Guide to Version 3, 2nd Edition**

The Open Group

Copyright © June 2004, The Open Group

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owners.

A Source Book from The Open Group

The Single UNIX[®] Specification: The Authorized Guide to Version 3, 2nd Edition

ISBN: 1-931624-47-X

Document Number: G041

Published in the U.S. by The Open Group, June 2004.

Any comments relating to the material contained in this document may be submitted by email to:

OGspecs@opengroup.org

Contents

Chapter 1	The Single UNIX Specification.....	1
1.1	Introduction.....	1
1.2	Background	1
1.3	The Value of Standards.....	2
1.4	The Single UNIX Specification	2
1.5	Benefits for Application Developers.....	3
1.6	Benefits for Users	3
1.7	The Common API Specification.....	3
1.8	The Single UNIX Specification, Version 2.....	5
1.9	The Single UNIX Specification, Version 3	5
1.10	IEEE Std 1003.1	8
1.11	ISO/IEC 9945	8
1.12	The Austin Group.....	8
1.13	Summary	9
Chapter 2	The Open Brand.....	11
2.1	Product Standards	12
2.2	Conformance Statements	12
2.3	Testing.....	13
2.4	The Single UNIX Specification, Version 3	13
Chapter 3	The Single UNIX Environment.....	15
3.1	Introduction.....	15
3.2	Formal Standards Alignment.....	16
3.3	Portability Codes.....	16
3.4	Option Groups	22
3.5	Common Directories and Devices	22
3.6	Environment Variables	23
3.7	YACC Grammars as Specifications	23
3.8	Regular Expressions	24
3.9	File Access	24
3.10	Programming Environment.....	24
3.10.1	Interface Counts	24
3.10.2	C-Language Support.....	25
3.10.3	Feature Test Macros and Name Space Issues	25
3.10.4	Error Numbers	26
3.10.5	Standard I/O Streams	26
3.10.6	STREAMS	26

3.10.7	XSI Interprocess Communication	26
3.10.8	Realtime	27
3.10.9	Threads	27
3.10.10	General Terminal Interface	27
3.10.11	How to Read an XSH Reference Page	27
3.11	Commands and Utilities Environment	28
3.11.1	Shell Command Language	28
3.11.2	Symbolic Links	28
3.11.3	File Format Notation	28
3.11.4	How to Read an XCU Reference Page	29
3.12	Networking Services Environment	29
3.13	Terminal Interfaces Environment	30
3.14	Internationalization	30
Chapter 4	The Single UNIX Specification, Version 3	33
4.1	Base Definitions (XBD)	33
4.2	System Interfaces (XSH)	34
4.3	Shell and Utilities (XCU)	34
4.4	Rationale (XRAT)	35
4.5	X/Open Curses (XCURSES)	35
Chapter 5	System Interfaces	37
5.1	Base Documents	37
5.2	Overview of Changes	38
5.3	System Interfaces by Category	44
5.4	XSH Option Groups	50
5.5	Options Policy	52
Chapter 6	Shell and Utilities	55
6.1	Options in XCU	55
6.2	Functional Overview	55
6.3	Base Documents	57
6.4	Overview of Changes	57
6.5	Exclusion of Utilities	59
Chapter 7	Headers	63
7.1	Header and Name Space Rules	63
7.1.1	ISO C Headers	63
7.1.2	POSIX.1-2001 Base Headers	64
7.1.3	XSI Headers	65
7.2	Names Safe to Use	65
7.2.1	Reserved Names	66
7.3	Base Documents	66
7.4	Overview of Changes	66

Chapter 8	Terminal Interfaces	69
8.1	Functional Overview.....	69
8.2	Curses in an Issue 6 Environment.....	70
Chapter 9	System Interface Table.....	73
Chapter 10	Utilities Interface Table	99
Chapter 11	Headers Interface Table	105
Chapter 12	XCURSES Interface Table	109
Chapter 13	System Interfaces Migration	119
Chapter 14	Utilities Migration.....	344
Chapter 15	Headers Migration.....	391
Chapter 16	ISO C Migration.....	425
16.1	Introduction.....	425
16.2	Language Changes.....	425
16.2.1	New Keywords.....	425
16.2.2	New Types.....	426
16.2.3	Type Qualifiers.....	426
16.2.4	Boolean	426
16.2.5	Universal Character Names	427
16.2.6	inline	427
16.2.7	Predefined Identifiers	427
16.2.8	Compound Literals.....	428
16.2.9	Designated Initializers	428
16.3	Decimal Integer Constants.....	428
16.3.1	String Literals.....	429
16.4	Implicit Declarations.....	429
16.4.1	sizeof.....	429
16.4.2	Multiplicative Operators.....	429
16.4.3	Enumeration Specifiers	429
16.5	Variable Length Array.....	430
16.5.1	Array Declarations	430
16.5.2	Array Type Compatibility	430
16.5.3	Incomplete Array Structure Members	430
16.5.4	Blocks.....	431
16.5.5	The for Statement.....	431
16.5.6	errno.....	431
16.6	Comments	431
16.6.1	Hexadecimal Floating-Point Constants	432
16.6.2	Predefined Macros.....	432
16.6.3	Source File Inclusion.....	432
16.6.4	Translation-Time Arithmetic.....	432

16.6.5	Minimum Maximum Line Length.....	432
16.6.6	Case-Sensitive Identifiers	432
16.6.7	line Directive	432
16.6.8	Empty Argument Macros	432
16.6.9	Pragmas	433
16.6.10	Translation Limits	433
16.6.11	Token Pasting	433
16.6.12	Variadic Macros	433
16.6.13	va_copy()	434
16.7	Headers	434
16.8	Integer Types	434
16.8.1	Exact-Width Integer Types	435
16.8.2	Minimum-Width Integer Types	435
16.8.3	Fastest Minimum-Width Integer Types	435
16.8.4	Integer Types Capable of Holding Object Pointers	435
16.8.5	Greatest-Width Integer Types	435
16.8.6	Limits of Specified-Width Integer Types	437
16.8.7	Macros	437
16.9	Complex Numbers	437
16.9.1	Trigonometric Functions	438
16.9.2	Hyperbolic Functions	439
16.9.3	Exponential and Logarithmic Functions	440
16.9.4	Power and Absolute-Value Functions	440
16.9.5	Manipulation Functions	440
16.10	Other Mathematical Changes	441
16.10.1	Classification Macros	441
16.10.2	Trigonometric Functions	442
16.10.3	Hyperbolic Functions	442
16.10.4	Exponential and Logarithmic Functions	443
16.10.5	Nearest Integer Functions	445
16.10.6	Remainder Functions	446
16.10.7	Manipulation Functions	446
16.10.8	Comparison Macros	447
16.11	Floating-Point Environment Support	447
16.11.1	Exceptions	448
16.11.2	Rounding	448
16.11.3	Environment	449
16.12	Type-Generic Math	449
16.12.1	Unsuffixes Functions With a C-Prefixed Counterpart	450
16.12.2	Unsuffixes Functions Without a C-Prefixed Counterpart	450
16.13	Other Library Changes	450
16.13.1	Wide-String Numeric Conversion Functions	454
16.14	Annexes	454
Appendix A	CD-ROM Instructions.....	457
A.1	About the CD-ROM.....	457
A.2	Contents	457
A.3	Viewing the Files.....	457

Appendix B	CD-ROM Terms and Conditions	459
	Index	461

About This Document

This book is designed for users, developers, and implementors of open systems. It provides complete information on what's new in the Single UNIX Specification, Version 3, including IEEE Std 1003.1-2001, with comprehensive reference material on every aspect. Finally, it includes a CD-ROM with the complete 4,000 page Single UNIX Specification, Version 3 (which includes IEEE Std 1003.1-2001) in HTML and PDF formats. This second edition has been updated to include coverage of the changes in Technical Corrigendum 1 (2003) and Technical Corrigendum 2 (2004).

This book is structured as follows:

- Chapter 1 describes the Single UNIX Specification, giving an overview of its evolution to become the Single UNIX Specification, Version 3. It describes the benefits of the specification and the analysis and initiatives that took place to arrive at the latest revision.
- Chapter 2 describes the Open Brand program and the Open Brand in relation to the Single UNIX Specification, Version 3.
- Chapter 3 describes how the Single UNIX Specification environment is organized and where to look for more detailed information in the Single UNIX Specification.
- Chapter 4 gives an overview of the documents that comprise the different parts of the Single UNIX Specification, Version 3, and how they are organized. This includes an overview of the contents of each of the documents and their relation to each other.
- Chapter 5 covers the C language programming interfaces that are the system interfaces, describing the base documents, an overview of the changes in this version, and lists of the new functions. It also describes the options and groups the interfaces into categories to give the reader an understanding of what facilities are available.
- Chapter 6 covers the shell and utilities, describing the base documents, an overview of the changes in this version, and lists of the new features. It also contains rationale of why certain utilities are excluded, and also a description of the options.
- Chapter 7 describes the contents of the headers included in XBD. This describes the base documents, an overview of the changes in this version, and lists of the new features.
- Chapter 8 describes the contents of X/Open Curses, Issue 4, Version 2.
- Chapter 9 gives a quick reference table of all the system interfaces defined in the Base Specifications, Issue 6, complete with an indication of their availability in the XSI extension, POSIX.1-2001 Base, UNIX 98, UNIX 95, POSIX.1-1996, POSIX.2-1992, C99, C89, SVID 3, and 4.3BSD.
- Chapter 10 gives a quick reference table of all the utility interfaces defined in the Base Specifications, Issue 6, complete with an indication of their availability in the XSI extension, POSIX.1-2001 Base, UNIX 98, UNIX 95, POSIX.1-1996, POSIX.2-1992, SVID 3, and 4.3BSD.

- Chapter 11 gives a quick reference table of the headers defined in the Base Specifications, Issue 6, complete with an indication of their availability in the XSI extension, POSIX.1-2001 Base, UNIX 98, UNIX 95, POSIX.1-1996, POSIX.2-1992, C99, and C89.
- Chapter 12 gives a quick reference table of the XCURSES interfaces, complete with an indication of their availability in other specifications.
- Chapter 13 provides migration information on the changes made to system interfaces in Issue 6 that might affect an application program.
- Chapter 14 provides migration information on the changes made to the utilities in Issue 6 that might affect an application program.
- Chapter 15 provides migration information on the changes made to the headers in Issue 6 that might affect an application program.
- Chapter 16 provides migration information on the changes made to the ISO C Language for the ISO/IEC 9899: 1999 standard.

The book also contains the following appendixes:

- Appendix A covers a number of topics concerning installation and contents of the accompanying CD-ROM.
- Appendix B contains the Terms and Conditions that apply to use of the accompanying CD-ROM.

About The Open Group

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.

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

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/bookstore.

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.

Trademarks

Boundaryless Information Flow™ and TOGAF™ are trademarks and Motif®, Making Standards Work®, OSF/1®, The Open Group®, UNIX®, and the “X” device are registered trademarks of The Open Group in the United States and other countries.

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

The Open Group acknowledges that there may be other company names and products that might be covered by trademark protection and advises the reader to verify them independently.

Acknowledgements

The Open Group gratefully acknowledges:

- The Austin Group for developing the Single UNIX Specification, Version 3.
- Finnbar P. Murphy for writing Chapter 16, the ISO C Language migration chapter.

Referenced Documents

The following documents are referenced in this guide:

ANSI C

American National Standard for Information Systems: Standard X3.159-1989, Programming Language C.

FIPS 151-2

Federal Information Procurement Standards (FIPS) 151-2, Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

HP-UX Manual

Hewlett-Packard HP-UX Release 9.0 Reference Manual, Third Edition, August 1992.

IEC 60559: 1989

IEC 60559: 1989, Binary Floating-Point Arithmetic for Microprocessor Systems (previously designated IEC 559: 1989). (Previously designated IEC 559.)

ISO/IEC 10646-1:2000

ISO/IEC 10646-1:2000, Information Technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.

ISO/IEC 10967-1:1994

ISO/IEC 10967-1:1994, Information Technology — Language-Independent Arithmetic — Part 1: Integer and Floating-Point Arithmetic.

ISO C (1990)

ISO/IEC 9899: 1990, Programming Languages — C, including Amendment 1:1995 (E), C Integrity (Multibyte Support Extensions (MSE) for ISO C).

ISO C (1999)

ISO/IEC 9899: 1999, Programming Languages — C, including Technical Corrigendum No. 1.

ISO POSIX-1: 1996

ISO/IEC 9945-1:1996, Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language] (identical to ANSI/IEEE Std 1003.1-1996). Incorporating ANSI/IEEE Stds 1003.1-1990, 1003.1b-1993, 1003.1c-1995, and 1003.1i-1995.

ISO POSIX-2: 1993

ISO/IEC 9945-2:1993, Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities (identical to ANSI/IEEE Std 1003.2-1992, as amended by ANSI/IEEE Std 1003.2a-1992).

Issue 1

X/Open Portability Guide, July 1985 (ISBN: 0-444-87839-4).

Issue 2

X/Open Portability Guide, January 1987:

- Volume 1: XVS Commands and Utilities (ISBN: 0-444-70174-5)
- Volume 2: XVS System Calls and Libraries (ISBN: 0-444-70175-3)

Issue 3

X/Open Specification, 1988, 1989, February 1992:

- Commands and Utilities, Issue 3 (ISBN: 1-872630-36-7, C211); this specification was formerly X/Open Portability Guide, Issue 3, Volume 1, January 1989, XSI Commands and Utilities (ISBN: 0-13-685835-X, XO/XPG/89/002)
- System Interfaces and Headers, Issue 3 (ISBN: 1-872630-37-5, C212); this specification was formerly X/Open Portability Guide, Issue 3, Volume 2, January 1989, XSI System Interface and Headers (ISBN: 0-13-685843-0, XO/XPG/89/003)
- Curses Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapters 9 to 14 inclusive; this specification was formerly X/Open Portability Guide, Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)
- Headers Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapter 19, Cpio and Tar Headers; this specification was formerly X/Open Portability Guide Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)

Issue 4

CAE Specification, July 1992, published by The Open Group:

- System Interface Definitions (XBD), Issue 4 (ISBN: 1-872630-46-4, C204)
- Commands and Utilities (XCU), Issue 4 (ISBN: 1-872630-48-0, C203)
- System Interfaces and Headers (XSH), Issue 4 (ISBN: 1-872630-47-2, C202)

Issue 4, Version 2

CAE Specification, August 1994, published by The Open Group:

- System Interface Definitions (XBD), Issue 4, Version 2 (ISBN: 1-85912-036-9, C434)
- Commands and Utilities (XCU), Issue 4, Version 2 (ISBN: 1-85912-034-2, C436)
- System Interfaces and Headers (XSH), Issue 4, Version 2 (ISBN: 1-85912-037-7, C435)

Issue 5

Technical Standard, February 1997, published by The Open Group:

- System Interface Definitions (XBD), Issue 5 (ISBN: 1-85912-186-1, C605)
- Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)
- System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)

Issue 6

Technical Standard, December 2001, published by The Open Group:

- Base Definitions (XBD), Issue 6, 2004 Edition (ISBN: 1-931624-43-7, C046)
- System Interfaces (XSH), Issue 6, 2004 Edition (ISBN: 1-931624-44-5, C047)
- Shell and Utilities (XCU), Issue 6, 2004 Edition (ISBN: 1-931624-45-3, C048)
- Rationale (Informative), Issue 6, 2004 Edition (ISBN: 1-931624-46-1, C049)

Referenced Documents

(See also POSIX.1:2001.)

MSE Working Draft

Working draft of ISO/IEC 9899:1990/Add3:Draft, Addendum 3 — Multibyte Support Extensions (MSE) as documented in the ISO Working Paper SC22/WG14/N205 dated 31 March 1992.

Open Brand

Brand Program Documentation, January 1998, Practical Guide to the Open Brand (ISBN: 1-85912-265-5, X981), published by The Open Group.

POSIX.1:1988

IEEE Std 1003.1-1988, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1:1990

IEEE Std 1003.1-1990, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1:1996

IEEE Std 1003.1-1996. IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1:2001

IEEE Std 1003.1-2001. IEEE Standard for Information Technology — Portable Operating System

(See also Issue 6.)

POSIX.1a

P1003.1a, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — (C Language) Amendment

POSIX.1d:1999

IEEE Std 1003.1d-1999, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 4: Additional Realtime Extensions [C Language].

POSIX.1g:2000

IEEE Std 1003.1g-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 6: Protocol-Independent Interfaces (PII).

POSIX.1j:2000

IEEE Std 1003.1j-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 5: Advanced Realtime Extensions [C Language].

POSIX.1q:2000

IEEE Std 1003.1q-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 7: Tracing [C Language].

POSIX.2b

P1003.2b, Standard for Information Technology — Portable Operating System Interface

(POSIX) — Part 2: Shell and Utilities — Amendment

POSIX.2d:-1994

IEEE Std 1003.2d:1994, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment 1: Batch Environment.

POSIX.13:-1998

IEEE Std 1003.13:1998, IEEE Standard for Information Technology — Standardized Application Environment Profile (AEP) — POSIX Realtime Application Support.

SVID, Issue 1

American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 1; Morristown, NJ, UNIX Press, 1985.

SVID, Issue 2

American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 2; Morristown, NJ, UNIX Press, 1986.

SVID, Issue 3

American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 3; Morristown, NJ, UNIX Press, 1989.

TMLA

Brand Program Documentation, January 1998, The Open Brand Trademark License Agreement (TMLA) (X982), published by The Open Group.

XNS, Issue 5.2

Technical Standard, January 2000, Networking Services (XNS), Issue 5.2 (ISBN: 1-85912-241-8, C808), published by The Open Group.

X/Open Curses, Issue 4, Version 2

CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), published by The Open Group.

The Single UNIX Specification

Many names have been applied to the work that has culminated in the Single UNIX Specification and the accompanying UNIX certification program. It began as the *Common API Specification*, became *Spec 1170*, and is now in its latest iteration the *Single UNIX Specification, Version 3*, published in a number of The Open Group's Technical Standards, the core of which are also IEEE Std 1003.1-2001.

This introductory chapter describes the Single UNIX Specification, giving an overview of its evolution to become the Single UNIX Specification, Version 3.

1.1 Introduction

This document introduces the Single UNIX Specification, Version 3 providing a brief history of the work that led to its creation, information on what's new in Version 3, migration information on the changes between Version 2 and Version 3, what is covered in each part of the specification, and how to use the documents to advantage to develop portable applications.

1.2 Background

Today, the cost of developing applications software is considerably greater than the cost of the hardware on which the applications run. Preserving this investment in an organization's critical applications, while freeing the organization to mix and match platforms as hardware costs fall, has led to the idea of "open systems". One of the cornerstones of open systems is portability—the ability to move an application's source code to another platform and rebuild the application without changing the source.

Software portability problems do not just affect an organization directly through its own applications. Hardware vendors often find their market limited by the applications that run on their platforms. Independent software vendors (ISVs) often limit themselves in the number of platforms on which they sell their products, due to the costs of maintaining multiple ports of the application for multiple platforms. Consumers are caught, often choosing hardware platforms based on the applications they need to purchase, but without the ability to know what new applications will be available on these platforms in the future.

The Single UNIX Specification, provides a single, open consensus specification to support development of portable applications. There is also a mark, or brand, that is used to identify those products that conform to the Single UNIX Specification. Both the specification and the trademark are managed and held in trust for the industry by The Open Group.

1.3 The Value of Standards

The UNIX system's increasing popularity spawned the development of a number of variations of the UNIX operating system in the 1980s, and the existence of these caused a mid-life crisis. Standardization had progressed slowly and methodically in domains such as telecommunications and third-generation languages; yet no one had addressed standards at the operating system level. For suppliers, the thought of a uniform operating environment was disconcerting. Consumer lock-in was woven tightly into the fabric of the industry. Individual consumers, particularly those with UNIX system experience, envisioned standardized environments, but had no way to pull the market in their direction.

However, for one category of consumer—governments—the standardization of the UNIX system was both desirable and within reach. Governments have clout and are the largest consumers of information technology products and services in the world. Driven by the need to improve commonality, both U.S. and European governments endorsed a shift to the UNIX system. The Institute of Electrical and Electronic Engineers POSIX family of standards, along with standards from ISO, ANSI, and others, led the way. Consortia such as the X/Open Company (merged with the Open Software Foundation in 1995 to form The Open Group) hammered out draft standards to accelerate the process.

In 1994, the definitive specification of what constitutes a UNIX system was finalized through X/Open Company's consensus process. The Single UNIX Specification was born, not from a theoretical, ivory tower approach, but by analyzing the applications that were in use in businesses across the world.

With the active support of government and commercial buyers alike, vendors began to converge on products that implement the Single UNIX Specification, and now all major vendors have products labeled UNIX 95, UNIX 98, or UNIX 03 which indicates that the vendor guarantees that the product conforms to the Single UNIX Specification.

Vendors continue to add value to the UNIX system, particularly in areas of new technology; however, that value will always be built upon a single, consensus standard. Meanwhile, the functionality of the UNIX system was established and the mid-life crisis was resolved. Suppliers today provide UNIX systems that are built upon the single, consensus standard.

1.4 The Single UNIX Specification

The Single UNIX Specification is a set of open, consensus specifications that define the requirements for a conformant UNIX system. The standardized programming environment provides a broad-based functional set of interfaces to support the porting of existing UNIX applications and the development of new applications. The environment also supports a rich set of tools for application development.

The Single UNIX Specification came into being when in 1994 Novell (who had acquired the UNIX systems business of AT&T/USL) decided to get out of that business. Rather than sell the business as a single entity, Novell transferred the rights to the UNIX trademark and the specification (that subsequently became the Single UNIX Specification) to The Open Group (at the time X/Open Company). Subsequently, it sold the source code and the product implementation (UNIXWARE) to SCO. The Open Group also owns the trademark UNIXWARE, transferred to them from SCO more recently.

Today, The Open Group's UNIX trademark may be applied to any operating system product that is guaranteed to meet the Single UNIX Specification. The Single UNIX Specification is designed to give software developers a single set of APIs to be supported by every UNIX system, and by making the specification freely available on the Web (www.unix.org), thousands of developers

are able to access this resource.

The most significant consequence of the Single UNIX Specification initiative is that it shifts the focus of attention away from incompatible UNIX system product implementations on to compliance with a single, agreed set of APIs. If an operating system meets the specification, and commonly available applications can run on it, then it can be reliably viewed as open.

1.5 Benefits for Application Developers

A single standard for the UNIX operating system means:

- Improved portability
- Faster development through the increased number of standard interfaces
- More innovation is possible, due to the reduced time spent porting applications

1.6 Benefits for Users

The Single UNIX Specification will evolve and develop in response to market needs protecting users' investment in existing systems and applications. The availability of the UNIX system from multiple suppliers gives users freedom of choice rather than being locked in to a single supplier. And the wide range of applications—built on the UNIX system's strengths of scalability, availability, and reliability—ensures that mission-critical business needs can be met.

1.7 The Common API Specification

The Common API Specification project started when several vendors (Sun Microsystems, IBM, Hewlett-Packard, Novell/USL, and OSF) joined together to provide a single unified specification of the UNIX system services. By implementing a single common definition of the UNIX system services, third-party independent software vendors (ISVs) would be able to more easily deliver strategic applications on all of these vendors' platforms at once.

The focus of this initiative was to deliver the core application interfaces used by current application programs. The economic driver that initiated the project was to ease the porting of existing successful applications. While the work was led by a central group of vendors, it received widespread support within the industry.

A two-pronged approach was used to develop the Common API Specification. First, a set of formal industry specifications was chosen to form the overall base for the work. This would provide stability and vendor neutrality, and lay a well charted course for future application development, taking advantage of the careful work that had gone into developing these specifications. It would also preserve the portability of existing applications already developed to these core models.

The X/Open XPG4 Base was chosen as the stable functional base from which to start. XPG4 Base supports the POSIX.1 system interface and the ANSI C/ISO C standard at its core. It also provides a rich set of 174 commands and utilities.

Many UNIX systems already conformed to this specification. To this base was added the traditional UNIX System V Interface Definition (SVID) Edition 3, Level 1 calls, and the OSF Application Environment Specification (AES) Full Use interface definitions. These represented the stable central core of the latter two specifications.

The second part of the approach was to incorporate interfaces that are acknowledged common practice but have not yet been incorporated into any formal specification or standard. The intent was to ensure that existing applications running on UNIX systems would port with relative ease to a platform supporting the Common API Specification. A survey of real-world applications was used to determine what additional interfaces would be required in the specification.

Fifty successful application packages were selected to be analyzed using the following criteria:

- Ranked in International Data Corporation's *Survey of Leading UNIX Applications*, 1992
- The application's domain of applicability was checked to ensure that no single application type (for example, databases) was overly represented
- The application had to be available for analysis either as source code, or as a shared or dynamic linked library

From the group of fifty, the top ten were selected carefully, ensuring that no more than two representative application packages in a particular problem space were chosen. The ten chosen applications were:

AutoCAD	FrameMaker	Island Write/Paint	SAS (4GL)	Teamwork
Cadence	Informix	Lotus 1-2-3	Sybase	WordPerfect

APIs used by the applications that were not part of the base specifications were analyzed:

- If an API was used by any of the top ten applications, it was considered for inclusion.
- If an API was not used by one of the top ten, but was used by any three of the remaining 40 applications, it was considered for inclusion.
- While the investigation of these 50 applications was representative of large complex applications, it was still not considered a broad enough survey, so an additional 3,500 modules were scanned. If an API was used at least seven times in modules that came from at least two platforms (to screen out vendor-specific libraries), then the interface was considered for inclusion.

The goal was to ensure that APIs in common use were included, even if they were not in the formal specifications that made up the base. Making the Common API Specification a superset of existing base specifications ensured that any existing applications should work unmodified. The sponsors of the work considered pruning the Common API Specification of interfaces from the base specifications that were not found to be in common use in the application survey.

This idea was rejected for two reasons.

While some of the interfaces in the base specifications were not yet considered common practice, their inclusion in the overall specification meant there existed clear signposts for future applications development work. Also, it was recognized that the applications chosen for the survey were still only a representative sample, and that many other applications not surveyed may use these interfaces.

When the survey was complete, there were 130 interfaces that did not already appear in the base specification. These interfaces seemed to be predominantly Berkeley BSD calls that had never been covered in XPG4 Base, the SVID, or the OSF AES, but did represent common practice in UNIX system applications developed originally on BSD-derived platforms. Such things as sockets and the 4.3BSD memory management calls were commonly used in many applications.

The resulting Common API Specification was impressive in its coverage. The top ten applications surveyed were completely covered. Of the remaining 40 application packages, the next 25 were within 5% of complete coverage. The software vendors involved all acknowledged

that it would be fairly straightforward for them to modify the 5% of applications to conform fully to the specification.

There were 1170 interfaces in the complete specification when the work was done (926 programming interfaces, 70 headers, 174 commands and utilities), and Spec 1170 was born.

Because of the breadth and origins of the specification, duplication of functionality existed. There were similar interfaces for doing the same thing in such areas as memory management (*bcopy()* versus *memmove()*) and creating temporary filenames (*tmpnam()* versus *mktemp()*). This duplication was allowed since it would increase the number of existing applications that would be portable in the new model. At the same time, certain functions were identified as the recommended practice for future development. There are cases where the duplicated functionality cannot coexist in the same application (for example, conflicting signals models), and it is important to ensure that the application is correctly configured if the expected behavior is to be observed.

In December 1993, Spec 1170 was delivered to X/Open for fasttrack processing into a proper industry supported specification. This work progressed through 1994 and culminated in publication of the Single UNIX Specification in October 1994. There were then more than 1170 interfaces in the specification as the review process shaped the document.

1.8 The Single UNIX Specification, Version 2

The Single UNIX Specification, Version 2 was published in February 1997. It incorporated standards for threads, realtime, removal of any architectural dependencies to permit 64-bit processing (which emerged from the Aspen Group), large file support, enhanced multibyte support, and addressed Year 2000 issues.

The five CAE specifications that make up the Single UNIX Specification, Version 2 are as follows:

- System Interface Definitions, Issue 5 (XBD5)
- System Interfaces and Headers, Issue 5 (XSH5)
- Commands and Utilities, Issue 5 (XCU5)
- Networking Services, Issue 5 (XNS5)
- X/Open Curses, Issue 4, Version 2

1.9 The Single UNIX Specification, Version 3

Work commenced in late 1998 on the Single UNIX Specification, Version 3. The core of the Single UNIX Specification, Version 3, collectively known as the *Base Specifications*, was developed, and is maintained, by a joint working group of members of the IEEE Portable Applications Standards Committee, members of The Open Group, and members of ISO/IEC Joint Technical Committee 1. This joint working group is known as the Austin Group.¹

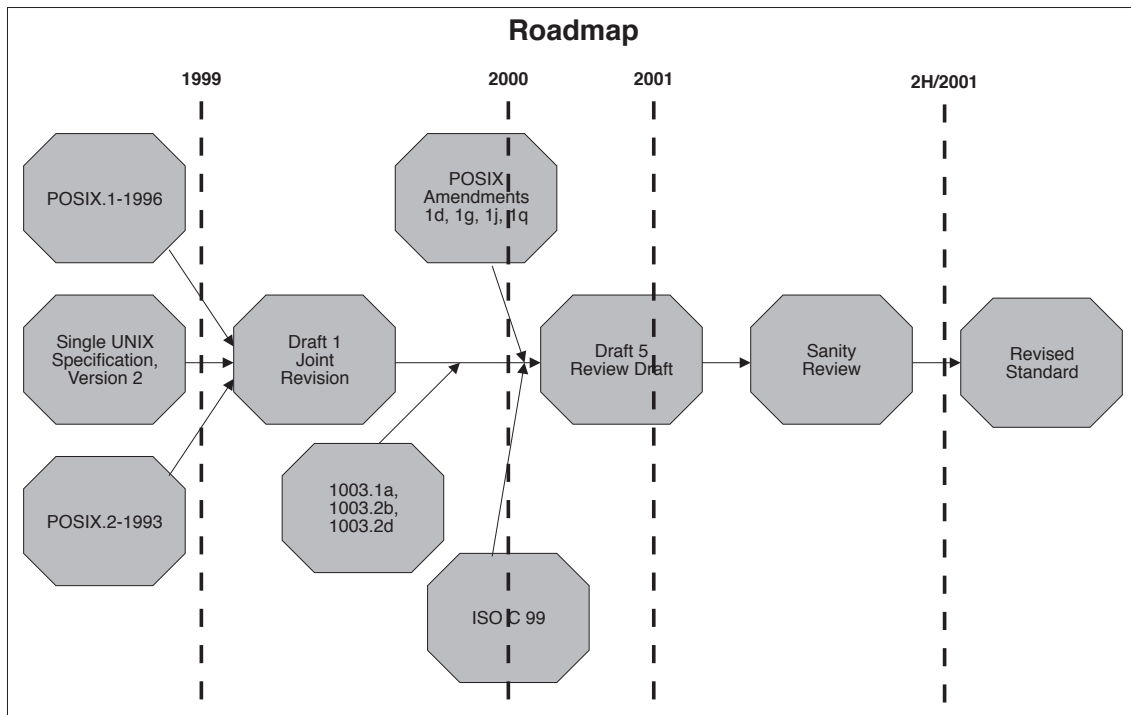
1. The Austin Group is named after the location of the inaugural meeting held at the IBM facility in Austin, Texas in September 1998.

The Austin Group arose out of discussions amongst the parties which started in early 1998, leading to an initial meeting and formation of the group in September 1998. This represented a seachange in attitude regarding development of two related specifications by three development organizations, which to date had been developed separately, often with the same standards developers involved.

The purpose of the Austin Group has been to revise, combine, and update the following standards: ISO/IEC 9945-1, ISO/IEC 9945-2, IEEE Std 1003.1, IEEE Std 1003.2, and the Base Specifications of The Open Group Single UNIX Specification.

After two initial meetings, an agreement was signed in July 1999 between The Open Group and the Institute of Electrical and Electronics Engineers (IEEE), Inc. to formalize the project, with the first draft of the revised specifications being made available at the same time. Under this agreement, The Open Group and IEEE agreed to share joint copyright of the resulting work. The Open Group has provided the chair and secretariat for the Austin Group.

The following figure shows the development roadmap from base documents through to approved standard:



This unique development has combined both the industry-led efforts and the formal standardization activities into a single initiative, and included a wide spectrum of participants including commercial, academia, government, and the open source communities.

The approach to specification development was one of “write once, adopt everywhere”, with the resulting set of specifications being approved as IEEE Std 1003.1-2001 (POSIX), The Open Group Base Specifications, Issue 6, and ISO/IEC 9945. This set of specifications forms the core of the Single UNIX Specification, Version 3.

The Base Specifications, Issue 6 consist of the following Technical Standards:

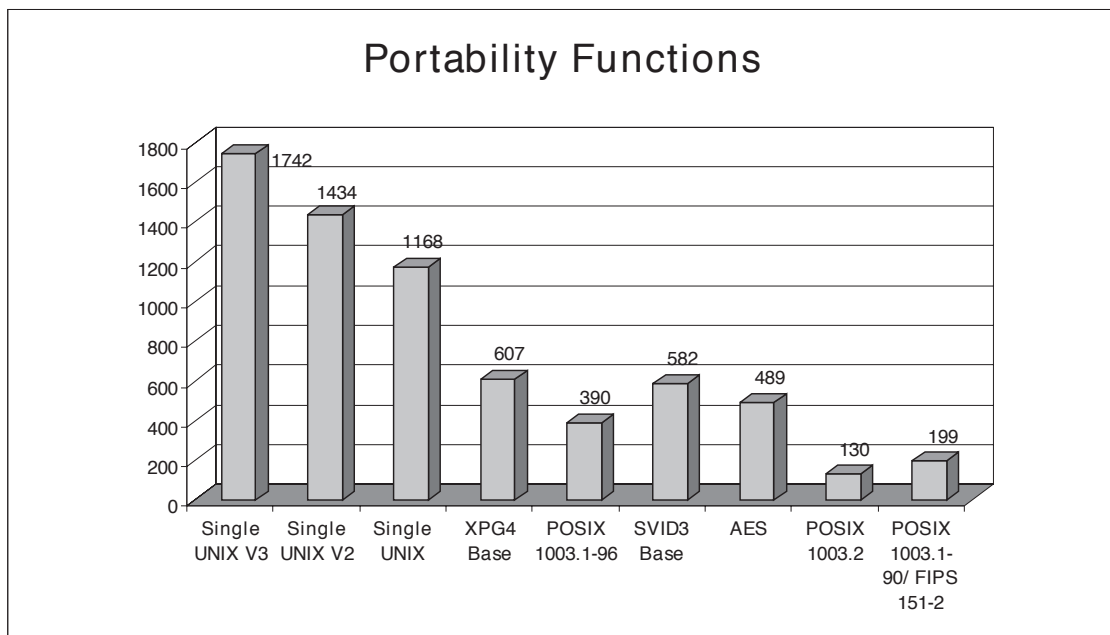
- Base Definitions, Issue 6 (XBD)
- Shell and Utilities, Issue 6 (XCU)
- System Interfaces, Issue 6 (XSH)
- Rationale (Informative)

The revision of the Base Specifications has tried to minimize the number of changes required to implementations which conform to the earlier versions of the approved standards to bring them into conformance with the current standard. Specifically, the scope of this work excluded doing any “new” work, but rather collecting into a single document what had been spread across a number of documents, and presenting it in what had been proven in practice to be a more effective way. Some changes to prior conforming implementations were unavoidable, primarily as a consequence of resolving conflicts found in prior revisions, or which became apparent when bringing the various pieces together.

However, since the revision now references the 1999 version of the ISO C standard, there are a number of unavoidable changes that have been made which will affect applications portability.

In addition to the Base Specifications, the Single UNIX Specification, Version 3 includes the X/Open Curses, Issue 4, Version 2 specification. Updates to X/Open Curses, Issue 4, Version 2 have been limited to production of a Corrigendum to allow it to exist in a Base Specifications, Issue 6 environment.

The following figure shows the interface counts for the different versions of the Single UNIX Specification and related documents.



1.10 IEEE Std 1003.1

The core of the Single UNIX Specification, Version 3 is also IEEE Std 1003.1-2001. IEEE Std 1003.1-2001 is a major revision and incorporates IEEE Std 1003.1-1990 (POSIX.1) and its subsequent amendments, and IEEE Std 1003.2-1992 (POSIX.2) and its subsequent amendments, combined with the core volumes of the Single UNIX Specification, Version 2. It is technically identical to The Open Group, Base Specifications, Issue 6; they are one and the same documents, the front cover having both designations. The final draft achieved 98% approval by the IEEE ballot group and was officially approved by the IEEE-SA Standards Board on December 6th 2001.

1.11 ISO/IEC 9945

The core of the Single UNIX Specification, Version 3 is also ISO/IEC 9945. ISO/IEC 9945 is the third edition of ISO/IEC 9945 and incorporates the former ISO/IEC 9945-1:1996 (POSIX-1) and ISO/IEC 9945-2:1993 (POSIX-2) plus subsequent IEEE amendments, combined with the core volumes of the Single UNIX Specification, Version 2. It is technically identical to The Open Group, Base Specifications, Issue 6 and IEEE Std 1003.1-2001; they are one and the same documents, the front cover having all the designations. The 2003 Edition of ISO/IEC 9945 was published on August 18th 2003.

ISO/IEC 9945 consists of the following parts, under the general title Information Technology — Portable Operating System Interface (POSIX):

- Part 1: Base Definitions
- Part 2: System Interfaces
- Part 3: Shell and Utilities
- Part 4: Rationale

1.12 The Austin Group

The Austin Group continues as the maintenance body for the Base Specifications. Since the initial publication of the Base Specifications, the Austin Group has produced two Technical Corrigenda making corrections to the document. Technical Corrigendum 1, also known as IEEE Std 1003.1-2001/Cor 1-2002 (identical to ISO/IEC 9945:2002/Cor 1:2003), was incorporated into the 2003 Edition of the document. Technical Corrigendum 2, also known as IEEE Std 1003.1-2001/Cor 2-2004 (identical to ISO/IEC 9945:2003/Cor 1:2004), is incorporated into the 2004 Edition.

Anyone wishing to participate in the Austin Group should contact the chair with their request. There are no fees for participation or membership. You may participate as an observer or as a contributor. You do not have to attend face-to-face meetings to participate; electronic participation is most welcome. For more information on the Austin Group and how to participate, see www.opengroup.org/austin.

1.13 Summary

The Single UNIX Specification offers benefits for all.

For users, products that support the Single UNIX Specification provide a powerful open platform for the development, implementation, and management of mission-critical computer systems. The specification facilitates freedom of choice across the widest possible range of systems, thereby protecting investment in existing software and data.

For software vendors, the Single UNIX Specification offers an opportunity to cut significantly the cost of developing and maintaining their products. It also expands the market opportunity by expanding the market itself, cutting time-to-market, and reducing product development costs.

For system vendors, the Single UNIX Specification unifies a previously fragmented market and offers the chance for all to compete on equal terms, while reducing the cost required to maintain technical incompatibilities.

The Open Brand

The Open Brand program is the means by which a vendor guarantees that a product complies with a particular Product Standard. It is built around the right to use a trademark to indicate that the product carries the guarantee, with trademark law as its legal basis. Once a product has been registered as conformant, the vendor is entitled to use the designated trademark in relation to that product. That right continues for so long as the product remains compliant and registered.

A vendor wishing to register a product must first sign the Trademark License Agreement (TMLA), thereby accepting its terms and conditions. This confers the right to use the designated trademark “on or in relation to” registered products, and the licensee warrants that any registered product fully conforms, and will continue to conform, to one or more Product Standards, and also commits to the speedy correction of any non-conformances.

The Open Brand program applies to computing system platforms, middleware, and other “service providers”—as opposed to the “consumers” of these services that would generally be regarded as an application of some kind. The focus is on the standards (and behavior) that apply to the interfaces presented to application programs through APIs, and, where appropriate, to the protocols used in communicating between applications and systems. Application software written to use the APIs defined by a Product Standard will be portable to, and interwork with, any other product registered as conformant to that Product Standard.

By calling for Open Brand registered products in their procurements, purchasers can ensure that they acquire products that are guaranteed to conform to the required Product Standards.

Anyone buying a registered product is guaranteed that:

- The product conforms to the identified Product Standard.
- The product conforms to all the specifications and standards, and complies with any special conditions, identified in the Product Standard.
- The product will continue to conform.

The product is guaranteed to remain conformant throughout fixes, functional enhancements, and performance improvements. If it does not continue to conform, and the non-conformance is not corrected in the predefined timescale, then the Product Registration, and the ability to use the trademark, will be lost.

- Any conformance problem will be fixed within the prescribed timescale.

In the real world, conformance problems may arise from time to time. The TMLA requires that these must be corrected in the defined timescale.

This chapter describes the Open Brand in relation to the Single UNIX Specification, Version 3.

Definitive details of the Open Brand program and the associated processes are published in *The Practical Guide to the Open Brand*, available in print from The Open Group, or on the Web at: www.opengroup.org/openbrand/Certification_Guide.

2.1 Product Standards

The Product Standard documents the conformance requirements that a product is required to meet. The Product Standard comprises a coherent set of system or product functionality which is implemented in a genuinely open manner by adherence to detailed technical specifications (plus other requirements, such as any options that must be implemented and any implementation actions that must be included). The specifications included in a Product Standard (and there may be one or more specifications) may be Open Group Technical Standards, specifications from other groups (such as IETF or OMG), other industry standards, or formal (*de jure*) standards.

Product Standards can include other Product Standards (although they can also refer directly to detailed technical standards) integrated together to form functionally more complete entities—they are intended to define a product in the form in which the buyer would want to buy and use it. The ability to combine the functionality of several Product Standards into a single conformance requirement (that is a “richer” Product Standard) greatly simplifies the buying of products guaranteed to conform through use of the Open Brand in procurement.

Products that conform to this type of Product Standard not only meet all the conformance requirements of the individual Product Standards, but also additional conformance requirements regarding the functionality concerned being provided in a fully harmonized and integrated way.

There is a complete list of all Product Standards to which products can be registered at: www.opengroup.org/openbrand/register/complist.htm.

New versions do not imply that the older version is necessarily obsolete. The compatibility relationship with earlier versions is identified in the Product Standard itself.

2.2 Conformance Statements

It is important that the application writer or purchaser has a clear and concise document that describes the way in which a given product conforms to the specifications with respect to these implementation options. The way in which a particular registered product has responded to these implementation options is documented in the product’s Conformance Statement.

The Conformance Statement uniquely identifies the product, describes the environment in which conformance was demonstrated (that is, the testing environment including any dependencies and services), and gives any additional information called for by the Product Standard.

The detailed specifications referenced by a Product Standard may explicitly allow for alternative behaviors, or may allow support for certain interfaces or elements of protocol to be optional; that is, their support is at the discretion of the implementor. There are also cases where numerical limits are defined, with a wide range of different values being possible; for example, the length of a password.

Conformance Statements are produced by completing a standard Questionnaire, one of which exists for each Product Standard. The Conformance Statement for a registered product has to be maintained by the vendor in an up-to-date state at all times.

Organizations procuring products carrying the Open Brand should look at the Conformance Statements for each product under consideration. This will allow the purchaser to clearly compare products from different vendors with respect to their support of options, and so on. All Conformance Statements can be browsed on-line at: www.opengroup.org/csq.

2.3 Testing

It is a requirement of the Open Brand that conformance of the product must have been demonstrated in a test environment and details of the environment used must be identified in the product's Conformance Statement. In addition, an audit can be requested at any time and require conformance to be demonstrated.

Where possible, stringent test suites are identified that must be successfully executed, and a test report submitted, before the product can be registered as conformant. Such test suites are identified in the Product Standard in the section headed *Indicator of Compliance*.

It should be noted, however, that the terms of the TMLA are such that the vendor warrants that the product conforms and commits to speedy correction of any non-conformances that may appear in the field. This is in addition to the requirement to have successfully run any designated test suite. The fact that a test suite may not have detected the non-conformance, or that the specific function is not covered by the test suite, does not reduce the commitment in any way.

Registration of branded products must be renewed annually. If a new test suite or new edition of a test suite is introduced as an indicator of compliance, a new test report is required as part of the renewal process at the first product renewal nine months after the introduction of the new or revised suite. This nine-month period gives the vendor time to build the test suite into the product release cycle.

This nine-month period applies only to existing releases of registered products. As new products or new product releases are brought to market, they must use the test suite criteria defined in the Product Standard at the time.

2.4 The Single UNIX Specification, Version 3

The Open Brand program includes the UNIX 03 label that can be applied to products that are guaranteed by the vendor to conform to the Single UNIX Specification, Version 3. The Open Brand for UNIX confers the right to associate the UNIX name with the product, and implies a warranty of conformance and a commitment to the timely rectification of any non-conformance.

At the time of writing there are two Product Standards for the Open Brand for the Single UNIX Specification, Version 3:

- The UNIX 03 Product Standard
- The UNIX 03 Server Product Standard

For the latest information on the Product Standards, refer to: www.opengroup.org/openbrand/register/complis.htm.

The UNIX 03 Product Standard is the foundation Product Standard for systems conforming to the Single UNIX Specification, Version 3. It is a significantly enhanced version of the UNIX 98 Product Standard. The mandatory enhancements include alignment with ISO/IEC 9899:1999, Programming Languages — C (ISO C), IEEE Std 1003.1-2001, and ISO/IEC 9945:2003. This Product Standard includes the following mandatory Product Standards:

- Internationalized System Calls and Libraries Extended V3
- Commands and Utilities V4
- C Language V2
- Internationalized Terminal Interfaces

The UNIX 03 Server Product Standard is a superset of the UNIX 03 Product Standard and adds interoperability to the UNIX 03 Product Standard to support Internet and intranet services. This includes support of network clients and the presence of a mandatory Java execution environment. The emphasis is towards services and applications support. A standard server will provide a set of core services to support Internet applications. The additional functionality over the UNIX 03 Product Standard includes:

- The Internet Protocol Suite
- Internet Protocol Version 6
- Secure Shell Protocol Version 2
- Java support
- Internet capabilities to support network clients

This Product Standard includes the following mandatory Product Standards:

- UNIX 03
- Internet Server V2

This Product Standard includes the following optional Product Standard:

- Internet Protocol Version 6

The Single UNIX Environment

The Single UNIX Specification programming environment provides a broad-based functional set of interfaces to support the porting of existing UNIX applications and the development of new applications. The environment also supports a rich set of tools for application development. This chapter describes the overall environment, programming, and utility considerations, and some of the global issues such as internationalization support, environment variables, and file access. It describes how the environment is organized and where to look for detailed information in the Single UNIX Specification. References give the name of the Single UNIX Specification document and section in which more detailed explanations and definitions can be found.

3.1 Introduction

The Single UNIX Specification programming environment began as the desire of a group of system vendors to provide real-world portability platforms to encourage software vendors to deliver their existing applications across all such UNIX environments. The resultant UNIX system environment has a much broader appeal than the original focus.

The Single UNIX Specification uses The Open Group Base Specifications, Issue 6 documentation as its core. The documentation is structured as follows:

- The Base Specifications, Issue 6, composed of:
 - Base Definitions, Issue 6 (XBD)
 - System Interfaces, Issue 6 (XSH)
 - Shell and Utilities, Issue 6 (XCU)
 - Rationale (Informative) (XRAT)
- X/Open Curses, Issue 4, Version 2 (XCURSES)

New for this version of the Single UNIX Specification is the incorporation of IEEE Std 1003.1 (POSIX) into the document set; The Open Group Base Specifications, Issue 6 and IEEE Std 1003.1-2001 are technically identical.

The base document for the joint revision of the documents was The Open Group's Base volumes of its Single UNIX Specification, Version 2. These were selected since they were a superset of the existing POSIX.1 and POSIX.2 specifications and had some organizational aspects that would benefit the audience for the new revision.

XBD contains all the general terms, concepts, and interfaces common to the Base Specifications, Issue 6, including utility conventions and C-language header definitions. It starts with a section on Scope and Terminology (Chapter 1) used throughout the document set, and has a section on Conformance (Chapter 2). Chapter 3 contains Definitions. Rather than being a simple compendium of acronyms and obscure references, this Definitions chapter describes all of the basic terms and definitions that apply throughout the Single UNIX Specification. Combined with Chapter 4, General Concepts, the differences between *character set* and

codeset, how *file access permissions* work, and how *pathname resolution* is accomplished are all described.

The new version now includes the addition of a separate Rationale document containing informative information. This complements the additional Rationale sections that have been included on the interface pages, and includes extended rationale that did not fit well into the rest of the document structure. It contains historical information concerning the contents of the Single UNIX Specification.

One thing that becomes apparent working with the Single UNIX Specification is its focus on application development. The Single UNIX Specification is similar to the User's and Programmer's Reference Manuals on Berkeley or System V systems. Matters of system management are not part of this specification. Directory organization is not discussed beyond the simple few directories and devices that applications generally use. User management discussions do not appear. There is no discussion of such files as */etc/passwd* or */etc/groups*, since an application's access to the information traditionally kept in these files is through programmatic interfaces such as *getpwnam()* and *getgrnam()*. Processes have *appropriate privileges*, and there is no concept of the "superuser" or "root".

3.2 Formal Standards Alignment

The Single UNIX Specification supports formal standards developed for applications portability. The following source code portability standards lie at the core of the Single UNIX Specification:

- IEEE Std 1003.1-2001

(This is technically identical to the Base Specifications, Issue 6; they are one and the same document.)

- ISO/IEC 9899: 1999, Programming Languages — C (ISO C)

The Single UNIX Specification fully aligns with these standards. With the merger of the Base Specifications and the POSIX standards, the Single UNIX Specification functional extensions beyond the required POSIX functionality are identified by the X/Open System Interface extension, denoted XSI. This can be thought of as a profile of the POSIX base standard, selecting certain options; for example, Threads. Where there are multiple interfaces to accomplish some task, the standards-based interfaces are clearly identified as the preferred way of doing things to support future portability.

3.3 Portability Codes

While great care has been taken to align the Single UNIX Specification with formal standards, it is still a superset specification that extends functionality or perhaps presents a more exact (and restrictive) definition. When this occurs, the text in the Single UNIX Specification is clearly marked by shading and a portability code appears in the margin (sometimes referred to as a margin code). Since the Base Specifications of the Single UNIX Specification are now merged with the POSIX standard, there are an increased number of codes to handle the underlying POSIX options, and these are referred to as Option Codes in XBD.

Programmers need to take care when using functionality that appears in a shaded area if they are developing applications that need to be maximally portable or portable beyond UNIX certified systems. For example, if functionality is marked with XSI in the margin, it will be available on all UNIX certified systems, but may not be available on systems only supporting the base POSIX.1 requirements. Alternatively, an application may depend on the exact format of

output from a particular utility whose output format is incompletely specified, as indicated by shading and OF marked in the margin. It is likely that an application is developed with a particular platform, or at least a well-defined set of platforms in mind. These codes are exceptionally useful to warn a developer of areas of potential problems.

There are fifty-six margin codes defined in total in XBD, Section 2.1.6, Options. Fifty of these reflect optional features defined within the POSIX base standard, eleven of which are mandatory in the Single UNIX Specification (CX, FSC, MF, MPR, THR, TSA, TSF, TSH, TSS, UP, and XSI). Three of the codes—MC1, MC2, and MC3—are shorthand notations for permutations of certain options. Three of the codes are special codes to denote other portability warnings, these being OB (obsolescent), OH (optional header), and OF (output incompletely specified).

All the codes including those of the POSIX options are listed below together with an indication of their status within the Single UNIX Specification:

ADV	Advisory Information
	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
AIO	Asynchronous Input and Output
	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
BAR	Barriers
	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Threads Option Group.
BE	Batch Environment Services and Utilities
	This identifies utilities and services that are optional in the Single UNIX Specification. They are defined in XCU and derived from IEEE Std 1003.2d-1994.
CD	C-Language Development Utilities
	This identifies a set of utilities that are optional in the Single UNIX Specification. They are defined in XCU.
CPT	Process CPU-Time Clocks
	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
CS	Clock Selection
	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
CX	Extension to the ISO C Standard
	Extensions beyond the ISO C standard are so marked. These extensions are mandatory on all systems supporting POSIX.1 and the Single UNIX Specification. This is included since the Single UNIX Specification defers to the ISO C standard, thereby allowing the extensions to be identified.

FD	FORTRAN Development Utilities This identifies a set of utilities that are optional in the Single UNIX Specification. They are defined in XCU.
FR	FORTRAN Runtime Utilities This identifies a set of utilities that are optional in the Single UNIX Specification. They are defined in XCU.
FSC	File Synchronization This identifies interfaces and additional semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
IP6	IPV6 This identifies additional semantics for networking interfaces relating to support of IP Version 6 that are optional in the Single UNIX Specification.
MC1	Advisory Information and either Memory Mapped Files or Shared Memory Objects This is a special margin code used as a shorthand notation.
MC2	Memory Mapped Files, Shared Memory Objects, or Memory Protection This is a special margin code used as a shorthand notation.
MC3	Memory Mapped Files, Shared Memory Objects, or Typed Memory Objects This is a special margin code used as a shorthand notation.
MF	Memory Mapped Files This identifies interfaces and additional semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
ML	Process Memory Locking This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
MLR	Range Memory Locking This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
MON	Monotonic Clock This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
MPR	Memory Protection This identifies interfaces and additional semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
MSG	Message Passing This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.

MX	IEC 60559 Floating-Point Option	This identifies additional semantics within Math interfaces that are optional in the Single UNIX Specification. Indicates support of the IEC 60559: 1989 standard floating-point.
OB	Obsolescent	Features marked as obsolescent are portable to all Single UNIX Specification platforms, but may be withdrawn in a future issue. This functionality should be avoided.
OF	Output Format Incompletely Specified	The format of a utility's output is incompletely specified, and therefore cannot be processed consistently by another application across multiple platforms.
OH	Optional Header	Certain headers are not required to be included in a source module on XSI-conforming systems, although the POSIX base standard may require it. These headers are marked in the SYNOPSIS sections.
PIO	Prioritized Input and Output	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification.
PS	Process Scheduling	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
RS	Raw Sockets	This identifies additional semantics for sockets that are optional in the Single UNIX Specification.
RTS	Realtime Signals Extension	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
SD	Software Development Utilities	This identifies a set of utilities that are optional in the Single UNIX Specification. They are defined in XCU.
SEM	Semaphores	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
SHM	Shared Memory Objects	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
SIO	Synchronized Input and Output	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
SPI	Spin Locks	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is

	part of the Advanced Realtime Threads Option Group.
SPN	Spawn This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
SS	Process Sporadic Server This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
TCT	Thread CPU-Time Clocks This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Threads Option Group.
TEF	Trace Event Filter This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Tracing Option Group.
THR	Threads This identifies interfaces and additional semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
TMO	Timeouts This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
TMR	Timers This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. They are part of the Realtime Option Group.
TPI	Thread Priority Inheritance This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is part of the Realtime Threads Option Group.
TPP	Thread Priority Protection This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is part of the Realtime Threads Option Group.
TPS	Thread Execution Scheduling This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is part of the Realtime Threads Option Group.
TRC	Trace This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Tracing Option Group.

TRI	Trace Inherit	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Tracing Option Group.
TRL	Trace Log	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Tracing Option Group.
TSA	Thread Stack Address Attribute	This identifies semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
TSF	Thread-Safe Functions	This identifies interfaces and additional semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
TSH	Thread Process-Shared Synchronization	This identifies interfaces and additional semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
TSP	Thread Sporadic Server	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Threads Option Group.
TSS	Thread Stack Address Size	This identifies semantics that are optional within the POSIX base standard but mandatory in the Single UNIX Specification.
TYM	Typed Memory Objects	This identifies interfaces and additional semantics that are optional in the Single UNIX Specification. This is new functionality added in this version of the specification, and is part of the Advanced Realtime Option Group.
UP	User Portability Utilities	This identifies interfaces and additional semantics in XCU that are optional within the POSIX base standard but mandatory in the Single UNIX Specification. These represent interactive utilities or semantics within utilities.
XSI	Extension	This code is introduced to support the merger of the Single UNIX Specification and the POSIX standards. The functionality marked XSI extends the core formal POSIX.1 requirements or makes mandatory something that is optional. This functionality is required to be present on all UNIX certified systems. Certain specific extensions (for example, Threads) have their own margin code.
XSR	XSI STREAMS	This identifies interfaces and additional semantics related to the support of STREAMS. This functionality is optional within this version of the Single UNIX Specification.

If the SYNOPSIS section of a reference page is shaded and marked with a portability code, then the entire interface is an extension. Refer to XBD, Section 1.5, Portability for more information.

3.4 Option Groups

The Single UNIX Specification includes a set of profiling options, allowing larger profiles of the options of the Base standard. In earlier versions of the Single UNIX Specification these were formerly known as Feature Groups. The Option Groups within the Single UNIX Specification are defined within XBD, Section 2.1.5.2, XSI Option Groups.

The Single UNIX Specification contains the following Option Groups:

- **Encryption**, covering the functions *crypt()*, *encrypt()*, and *setkey()*.
- **Realtime**, covering the functions from the IEEE Std 1003.1b-1993 Realtime extension.
- **Realtime Threads**, covering the functions from the IEEE Std 1003.1c-1995 Threads extension that are related to realtime functionality.
- **Advanced Realtime**, covering some of the non-threads-related functions from IEEE Std 1003.1d-1999 and IEEE Std 1003.1j-2000.
- **Advanced Realtime Threads**, covering some of the threads-related functions from IEEE Std 1003.1d-1999 and IEEE Std 1003.1j-2000.
- **Tracing**, covering the functionality from IEEE Std 1003.1q-2000.
- **XSI STREAMS**, covering the functionality and interfaces related to STREAMS, a uniform mechanism for implementing networking services and other character-based I/O as described in XSH, Section 2.6, STREAMS. This was mandatory in previous versions of the Single UNIX Specification, but is now optional in this version.
- **Legacy**, covering the functionality and interfaces which were mandatory in previous versions of the Single UNIX Specification, but are optional in this version.

3.5 Common Directories and Devices

The Single UNIX Specification describes an applications portability environment, and as such defines a certain minimal set of directories and devices that applications regularly use. The following directories are defined:

- /** The root directory of the file system.
- /dev** Contains the devices **/dev/console**, **/dev/null**, and **/dev/tty**.
- /tmp** A directory where applications can create temporary files.

The directory structure does not cross into such system management issues as where user accounts are organized or software packages are installed. Refer to XBD, Section 10.1, Directory Structure and Files for more information.

XBD, Chapter 10, Directory Structure and Devices also defines the mapping of control character sequences to real character values, and describes the actions an implementation must take when it cannot support certain terminal behavior.

3.6 Environment Variables

When a program begins, an environment is made available to it. The environment consists of strings of the form *name=value*, where *name* is the name associated with the environment variable, and its value is represented by the characters in *value*. UNIX systems traditionally pass information to programs through the environment variable mechanism. The Single UNIX Specification uses only uppercase characters and underscores to name environment variables, reserving for applications the name space of names containing lowercase characters.

A number of utilities and functions defined in the Single UNIX Specification use environment variables to modify their behavior. The ENVIRONMENT VARIABLES section of a utility's reference page describes any appropriate environment variables. Quite a number of environment variables (listed below) modify the behavior of more than a single utility.

ARFLAGS	IFS	MAILPATH	PS1
CC	LANG	MAILRC	PS2
CDPATH	LC_ALL	MAKEFLAGS	PS3
CFLAGS	LC_COLLATE	MAKESHELL	PS4
CHARSET	LC_CTYPE	MANPATH	PWD
COLUMNS	LC_MESSAGES	MBOX	RANDOM
DATEMSK	LC_MONETARY	MORE	SECONDS
DEAD	LC_NUMERIC	MSGVERB	SHELL
EDITOR	LC_TIME	NLSPATH	TERM
ENV	LDFLAGS	NPROC	TERMCAP
EXINIT	LEX	OLDPWD	TERMINFO
FC	LFLAGS	OPTARG	TMPDIR
FCEDIT	LINENO	OPTERR	TZ
FFLAGS	LINES	OPTIND	USER
GET	LISTER	PAGER	VISUAL
GFLAGS	LOGNAME	PATH	YACC
HISTFILE	LPDEST	PPID	YFLAGS
HISTORY	MAIL	PRINTER	
HISTSIZ	MAILCHECK	PROCLANG	
HOME	MAILER	PROJECTDIR	

3.7 YACC Grammars as Specifications

The Single UNIX Specification describes certain functionality and “little” languages with *yacc* grammars. The *yacc* utility (Yet-Another-Compiler-Compiler) is used to build language parsers, and the syntax of the *yacc* input language is designed for describing languages. Because of this, and its familiarity in the community, it has been used in the Single UNIX Specification to describe the shell (XCU, Chapter 2, Shell Command Language), the *awk* language (XCU, Chapter 4, Utilities, *awk*), the *bc* language (XCU, Chapter 4, Utilities, *bc*), the locale definition language (XBD, Chapter 7, Locale), basic and extended regular expressions (XBD, Chapter 9, Regular Expressions), and *terminfo* source format (X/Open Curses, Section 6.1.2, Formal Grammar).

These grammars are representations of the languages they describe, and a number of caveats apply:

- They are not complete input grammars to *yacc* itself.
- They are not based on any existing implementations, nor have they been tested as such.

- Were these partial grammars to be completed, there are no guarantees that they are the most efficient method of defining the respective languages for *yacc* to process.

Refer to XCU, Chapter 1 for more information.

3.8 Regular Expressions

Both Basic Regular Expressions (BREs) and Extended Regular Expressions (EREs) are described in XBD, Chapter 9, Regular Expressions and all of the utilities and interfaces that use regular expressions refer back to this definition.

Basic regular expressions:

csplit, ctags, ed, ex, expr, grep, more, nl, pax, pg, sed, vi

Extended regular expressions:

awk, egrep, grep -E, lex

The functions *regcomp()* and *regexexec()* in XSH, Chapter 3, System Interfaces implement regular expressions as defined in the Single UNIX Specification.

3.9 File Access

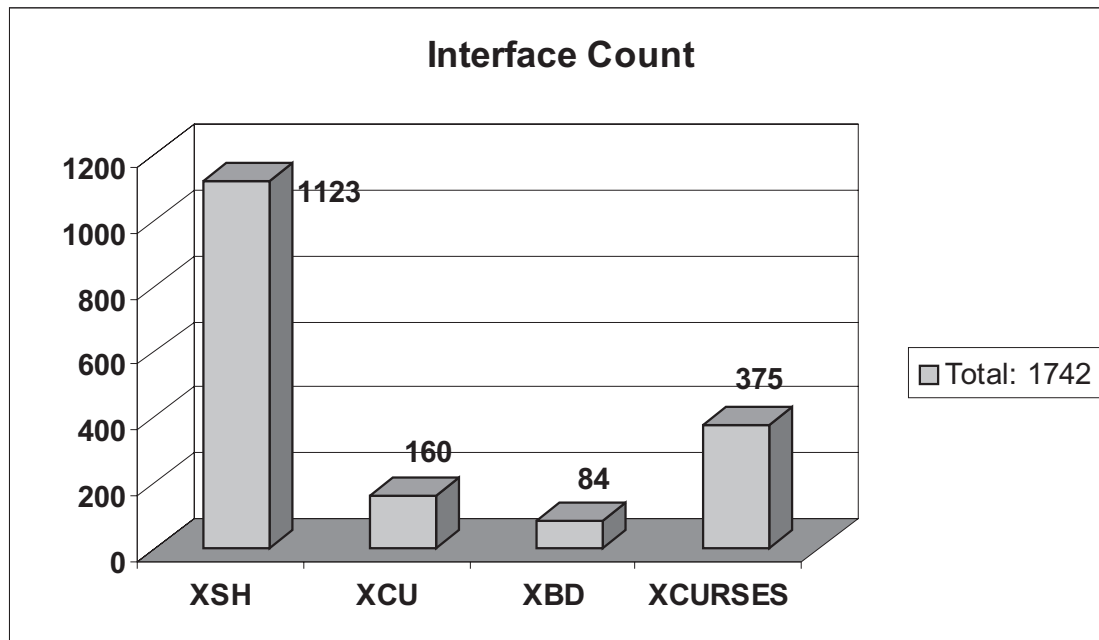
The Single UNIX Specification describes the interaction of symbolic links that may effect the file access in the file system. The behavior of symbolic links is fully specified with respect to their creation and use through the relevant XSH interfaces, such as *lstat()*, *readlink()*, *realpath()*, and *symlink()*, and pathname resolution (XBD, Chapter 4, General Concepts). The behavior is also specified with respect to the utilities described in XCU (this is new for this version of the Single UNIX Specification).

3.10 Programming Environment

The Single UNIX Specification and UNIX certified systems are tools for developing portable applications and for porting existing applications that were originally developed to run on UNIX systems. XSH defines the C-language programming environment, and the syntax and semantics of the interfaces. Feature test macros, name space issues, and the program interaction with the operating system are all described in the opening chapters of the document. The following introductions to these topics include references and additional explanations to orient an application developer with the information presented in XSH.

3.10.1 Interface Counts

There are now 1742 interfaces defined in the Single UNIX Specification, Version 3. The figure below gives the breakdown on a per-document basis. Also see the interface tables in Chapter 9 through Chapter 12 for detailed lists of the interfaces.



3.10.2 C-Language Support

The programming interfaces in XSH are described in C-language syntax, as defined in the ISO C standard, and presume a C-language compilation environment. (Implementations may make the functionality available through other programming languages, but this is not covered by the Single UNIX Specification.)

For an implementation to be a conforming UNIX certified system, it must support the XPG4 C Language component, which identifies the ISO C standard dialect. Implementations may additionally support the X/Open Common Usage C dialect as a migration strategy from previous XPG3 environments. X/Open Common Usage C is defined in Programming Languages, Issue 3, Chapters 1-4, and essentially refers to the C Language before the ISO C standard.

XCU defines *c99* as the interface to the C compilation environment. The *c99* interface is new to this version of the specification and an interface to the standard C compiler. The *c89* and *cc* utilities are no longer defined in this version of the Single UNIX Specification although implementations may additionally support them for backwards-compatibility.

3.10.3 Feature Test Macros and Name Space Issues

There are a number of tasks that must be done to effectively make the interface environment available to a program. A number of C-language macros, referred to as *feature test macros*, must be defined before any headers are included. These macros might more accurately be referred to as header configuration macros, as they control what symbols and prototypes will be exposed by the headers. The macro `_XOPEN_SOURCE` must be defined to a value of 600 to make available the functionality of the Single UNIX Specification, Version 3. With respect to POSIX functionality covered by the Single UNIX Specification, this is equivalent to defining the POSIX macro `{_POSIX_C_SOURCE}` to be 200112L.

Use of the `{_XOPEN_SOURCE}` macro should not be confused with the other feature test macros associated with Feature Groups and functionality, such as `{_XOPEN_UNIX}`. These feature test macros are the implementation's way of announcing functionality to the application.

In the first case (implementation-defined `{_XOPEN_SOURCE}=600`), the application program is announcing to the implementation its desire to have certain symbols and prototypes exposed. In the second case (implementation-defined `{_XOPEN_UNIX}`), the application program is testing the implementation to determine whether the functionality exists.

The UNIX system name space is well defined. All identifiers defined in the Single UNIX Specification appear in the headers described in XBD, Chapter 13, Headers (with the exception of *environ*). Tables in XSH, Section 2.2, The Compilation Environment clearly describe which identifier prefixes and suffixes are reserved for the implementation, which identifier macro prefixes may be used by the application programmer providing appropriate `#undef` statements are used to prevent conflicts, and which identifiers are reserved for external linkage.

3.10.4 Error Numbers

Each interface reference page lists in the ERRORS section possible error returns that may be tested in *errno* upon the unsuccessful completion of a function call. Each error return has a symbolic name (defined as a manifest constant in `<errno.h>`) which should always be used by the portable application, as the actual error values are unspecified. All of the error names are listed in XSH, Section 2.3, Error Numbers along with additional relevant information.

3.10.5 Standard I/O Streams

When a program starts, it has three I/O streams associated with it, namely *standard input* (the keyboard), *standard output*, and *standard error* (both associated with the display). These streams are already open for the process, and ready for I/O.

The mechanics of stream I/O, buffering, relationships to file descriptors, and inheritance across process creation are all discussed in XSH, Section 2.5, Standard I/O Streams.

3.10.6 STREAMS

STREAMS is a method of implementing network services and other character-based input/output mechanisms, with the STREAM being a full-duplex connection between a process and a device. STREAMS provides direct access to protocol modules, and optional protocol modules can be interposed between the process-end of the STREAM and the device-driver at the device-end of the STREAM. Pipes can be implemented using the STREAMS mechanism, so they can provide process-to-process as well as process-to-device communications.

XSH, Section 2.6, STREAMS introduces STREAMS I/O, the message types used to control them, an overview of the priority mechanism, and the interfaces used to access them. In this version of the specification, support for STREAMS functionality is a separate option, denoted by the XSR option marker.

3.10.7 XSI Interprocess Communication

The Single UNIX Specification describes a set of interprocess communications (IPC) primitives, namely message queues, semaphores, and shared memory. These functions are all derived from the SVID.

General information that is shared by all three mechanisms is described in XSH, Section 2.7, XSI Interprocess Communication. There, the common permissions mechanism is briefly introduced, describing the mode bits and how they are used to determine whether or not a process has access to read or write/alter the appropriate instance of one of the IPC mechanisms. All other relevant information is contained in the reference pages themselves.

3.10.8 Realtime

The Single UNIX Specification includes realtime functionality to support the source portability of applications with realtime requirements. This is derived from the POSIX Realtime Extension. Realtime is discussed in XSH, Section 2.8, Realtime. This section includes an overview of the functional areas: semaphores, process memory locking, memory mapped files and shared memory objects, priority scheduling, realtime signals, timers, interprocess communication, synchronous input/output, and asynchronous input/output.

3.10.9 Threads

XSH describes functionality to support multiple flows of control, called *threads*, within a process. These are derived from the POSIX Threads Extension, with additional X/Open extensions. Threads are discussed in XSH, Section 2.9, Threads, which includes an overview of the supported interfaces, threads implementation models, threads mutexes, threads attributes, thread scheduling, thread cancelation, and thread read-write locks.

3.10.10 General Terminal Interface

The general terminal interface is described in the Single UNIX Specification in XBD, Chapter 11, General Terminal Interface, providing a mechanism to control asynchronous communications ports. It is left to implementations as to whether or not they support network connections and synchronous communications ports. While the core definition comes from POSIX.1, it has been extended in the Single UNIX environment to support additional input and output modes.

While all of the interface details are contained in XSH, the mechanics of the terminal interface with respect to process groups, controlling terminals, input and output processing, input, output, and control modes, and special characters are described in XBD, Chapter 9, General Terminal Interface.

This interface should not be confused with the X/Open Curses interface that provides a terminal-independent way to update character screens.

3.10.11 How to Read an XSH Reference Page

Each reference page in XSH has a common layout of sections describing the interface. (Function interface descriptions in X/Open Curses follow the same layout.) This layout is similar to the manual page or “man” page format shipped with most UNIX systems, and each interface has SYNOPSIS, DESCRIPTION, RETURN VALUE, and ERRORS sections. These are the four sections that relate to conformance.

Additional sections contain considerable extra information for the application developer. The EXAMPLES sections provide source code examples of how to use certain interfaces. The APPLICATION USAGE sections provide additional caveats, issues, and recommendations to the developer. The SEE ALSO sections contain useful pointers to related interfaces and headers that a developer may wish to also read.

The FUTURE DIRECTIONS sections act as pointers to related work that may impact on the interface in the future, and often cautions the developer to architect the code to account for a change in this area. (A FUTURE DIRECTIONS section expresses current thinking and should not be considered a commitment to adopt the feature or interface in the future.)

A new RATIONALE section has been added to this version of the specification which includes historical information about an interface and why features were included or discarded in the definition.

The CHANGE HISTORY section describes when the interface was introduced, and how it has changed. This information can be useful when porting existing applications that may reflect earlier implementations of the interface.

Option Group labels in the reference page headers, and portability shading and margin marks are features already described in this document; they appear on the reference pages to guide an application developer when deciding how best an interface should be used. Refer to XSH, Section 1.8, Format of Entries for information on the exact layout.

3.11 Commands and Utilities Environment

The Single UNIX Specification describes 174 utilities supported on a UNIX certified system. These utilities provide a rich environment for building shell script applications, supporting program development (the C Language in particular), and providing a user portability environment.

XCU, along with the shell command language, symbolic links, file format notation, utility reference page layouts, and guidelines, are all introduced in this section. The following introductions to these topics include references and additional explanations to orient an application developer with the information presented in XCU.

3.11.1 Shell Command Language

The shell is a powerful and flexible programming language. A considerable number of utilities on early UNIX systems were actually shell programs.

The Single UNIX Specification shell is the standard POSIX shell. This shell is for the most part based on the Bourne shell with features from the KornShell, *ksh*.

The shell command language is defined in its entirety in XCU, Chapter 2, Shell Command Language. This is a very strict definition of the shell. Token recognition, word expansions, simple and compound commands, the shell grammar, the execution environment, and special built-ins are a few of the topics covered. It is not a guide to writing or porting shell scripts.

3.11.2 Symbolic Links

The Single UNIX Specification adds support for symbolic links in the file system. The programmatic interfaces that manipulate symbolic links (or *symlinks*) are all well defined in XSH and XCU, and *symlink* concepts with respect to issues like pathname resolution are discussed in XBD, Chapter 4, General Concepts. A UNIX certified system supports *symlinks*, and application programs may make use of them.

3.11.3 File Format Notation

Sections in the utility reference pages often require the expected input used by the utility or output it generates to be described. Additionally, information files used or created sometimes require description. The method used throughout XCU is a format description plus its arguments, similar to that used by the *printf()* function. These file format specifications are presented as:

```
"<format>"[ ,<arg1>, <arg2>, ..., <argn>]
```

The *format* specifier contains the format string a programmer might use to write the data. The specifiers should not be considered format strings that could be directly used in a call to *scanf()*. The conversion specifications in the format string are what would be expected by a developer

familiar with the *printf()* interface. Refer to XBD, Chapter 5, File Format Notation for more information.

3.11.4 How to Read an XCU Reference Page

Each reference page in XCU has a common layout of sections describing the interface. This layout, while similar to the manual page or “man” page format shipped with most UNIX systems, offers a more detailed view of the utility’s description.

As well as the SYNOPSIS and DESCRIPTION sections, each interface has OPTIONS, OPERANDS, STDIN (standard input format), INPUT FILES, ENVIRONMENT VARIABLES, ASYNCHRONOUS EVENTS (what signals are caught and the consequence of receiving signals), STDOUT (standard output format), STDERR, and OUTPUT FILES sections.

An EXTENDED DESCRIPTION will be used if the utility has a particularly long description; for example, if it supports its own language (*awk*), or provides a considerable number of subcommands (*mailx*).

Utilities generally return 0 upon successful completion, and a failed status as greater than 0. The EXIT STATUS sections specify this, but will also describe if particular values are returned in certain circumstances. In general, an application should be written to test for successful completion, rather than specific error returns.

The CONSEQUENCE OF ERRORS section describes what happens to such items as open files, process state, and the environment, should errors occur.

As with the XSH reference pages, additional sections contain considerable extra information for the application developer, and include EXAMPLES, APPLICATION USAGE, FUTURE DIRECTIONS, RATIONALE, SEE ALSO, and CHANGE HISTORY sections. The defaults for these sections, and additional detail about what each section specifies, are covered in XCU, Section 1.11, Utility Description Defaults.

3.12 Networking Services Environment

Unlike previous versions of the Single UNIX Specification, the Networking Services have now been integrated into the Base Specifications. This includes sockets and IP address resolution interfaces. The X/Open Transport Interface (XTI) is no longer a requirement in this version of the Single UNIX Specification. As with other functions in XSH, the application developer needs to define `{_XOPEN_SOURCE}` to be 600 prior to the inclusion of any Single UNIX Specification headers.

The *c99* compiler utilities recognize the additional `-I` operand for the **xnet** library.

3.13 Terminal Interfaces Environment

The Single UNIX Specification includes X/Open Curses. These interfaces provide a terminal-independent character screen update method. The functionality includes:

- Multibyte and wide-character support
- Color terminal support
- Wide and non-spacing character generalizations such that multi-column display characters and non-spacing characters are supported

New interfaces over Issue 3 are marked with ENHANCED CURSES in the reference page header, and shading and a portability code is used (EC) to indicate in the introductory chapters and reference pages the extended features.

Applications using the interfaces from X/Open Curses need to define the `{_XOPEN_SOURCE}` macro to be 600 prior to including any headers. These feature test macros ensure the appropriate name space is exposed in the headers.

On UNIX certified systems, the *c99* compiler recognizes the additional `-I` operand for the **curses** library.

3.14 Internationalization

There is a rich set of interfaces in the Single UNIX Specification to support internationalized applications development. Internationalization refers to developing an application without prior knowledge of the language, cultural information, or character set encoding scheme that will be used in the run-time environment. The application responds accordingly at run time for cultural or locale-specific conditions. The term “internationalization” is often shortened to simply “I18N”, for “I”—18 letters (nternationalizatio)—“N”.

Localization is the process of establishing a base of cultural and codeset data on a system such that it can be accessed. It is the method by which *locales* are created in such a way that internationalized programs can access the relevant information.

There are a number of factors that need to be considered when structuring an application to support multiple cultures:

- Cultural-specific data (for example, date, number, and monetary formats) needs to be defined in a locale, with the appropriate tools and interfaces available for creating and accessing the information.
- Provision for handling text strings in programs, and managing catalogs of these strings for multiple natural languages, needs to be made.
- Multibyte and wide-character handling interfaces are required to manage textual data that is not represented in single byte per character formats.
- Codeset conversion tools need to be supported.

All of these requirements are met in the Single UNIX Specification. Utilities exist to support the definition and display of locales (*localedef* and *locale*). Programming interfaces exist to access the locale-specific data, and others have been extended such that all the checking and manipulations done with characters and strings can be done with wide characters and wide-character strings. Interfaces to access message string catalogs (*catopen()*, *catgets()*, and *catclose()*) and a tool to define them (*gencat*) exist. Codeset conversions can be done at the utility (*iconv*) and program level (*iconv_open()*, *iconv()*, and *iconv_close()*).

Much of the I18N information is spread throughout the Single UNIX Specification Specification, with specific interface and utility descriptions appearing in XSH and XCU, respectively. Introductions to character and codeset issues and the locale definition language appear in XBD, Chapter 6, Character Set, and Chapter 7, Locale, respectively.

The Single UNIX Specification, Version 3

This chapter gives an overview of the documents that comprise the different parts of the Single UNIX Specification, Version 3 and how they are organized. The Single UNIX Specification, Version 3 is made up of the Base Specifications, Issue 6 and X/Open Curses, Issue 4, Version 2. The Base Specifications, Issue 6 comprise Base Definitions, System Interfaces, Shell and Utilities, and Rationale.

4.1 Base Definitions (XBD)

The XBD document is part of the Base Specifications, Issue 6. XBD provides common definitions for the Base Specifications of the Single UNIX Specification; therefore readers should be familiar with it before using the other parts of the Single UNIX Specification. The presence of this document reduces duplication in the other related parts of the Single UNIX Specification and ensures consistent use of terminology.

This document is structured as follows:

- Chapter 1 is an introduction which includes the scope of the Base Specifications, and the scope of the changes made in this revision. Normative references, terminology, and portability codes used throughout the Base Specifications are included in this chapter.
- Chapter 2 defines the conformance requirements, both for implementation and application conformance. For implementation conformance, this includes documentation requirements, conformance definitions for the core POSIX subset, conformance definitions for systems conforming to the Single UNIX Specification (denoted as the XSI extension), and option groups (previously known as feature groups).
- Chapter 3 contains the general terms and definitions that apply throughout the Base Specifications.
- Chapter 4 describes general concepts that apply throughout the Base Specifications.
- Chapter 5 describes the notation used to specify file input and output formats in XBD and XCU.
- Chapter 6 describes the portable character set and the process of character set definition.
- Chapter 7 describes the syntax for defining internationalization locales as well as the POSIX locale provided on all systems.
- Chapter 8 describes the use of environment variables for internationalization and other purposes.
- Chapter 9 describes the syntax of pattern matching using regular expressions employed by many utilities and matched by the *regcomp()* and *regexec()* functions. Both Basic Regular Expressions (BREs) and Extended Regular Expressions (EREs) are described in this chapter.

- Chapter 10 describes files and devices found on all systems and their semantics. For example, the device `/dev/null` is an infinite data source and data sink.
- Chapter 11 describes the asynchronous terminal interface for many of the functions in XSH and the `stty` utility in XCU.
- Chapter 12 describes the policies for command line argument construction and parsing. It contains the utility argument syntax used throughout XCU, and also utility syntax guidelines for naming of utilities and the specification of their arguments and option-arguments and operands.
- Chapter 13 defines the contents of headers which declare constants, macros, and data structures that are needed by programs using the services provided by the system interfaces defined in XSH. These are in the form of reference pages and are organized alphabetically.

4.2 System Interfaces (XSH)

The XSH document is part of the Base Specifications, Issue 6. XSH describes a set of system interfaces offered to application programs by systems conformant to this part of the Single UNIX Specification. Readers are expected to be experienced C language programmers, and to be familiar with the XBD document.

This document is structured as follows:

- Chapter 1 explains the status of this document and its relationship to other formal standards. The scope, conformance, and definitions sections are pointers to the XBD document; the sections are here to meet ISO/IEC rules regarding required sections. The terminology and portability codes are identical to the section in XBD and repeated here for ease of reference.
- Chapter 2 contains important concepts, terms, and caveats relating to the rest of this document. This includes information on the compilation environment, the name space, definitions of error numbers, signal concepts, standard I/O streams, STREAMS, XSI IPC, realtime, threads, sockets, tracing, and data types.
- Chapter 3 defines the functional interfaces to systems conformant to this part of the Single UNIX Specification. These are in the form of reference pages and are organized alphabetically.

4.3 Shell and Utilities (XCU)

The XCU² document is part of the Base Specifications, Issue 6. XCU describes the shell and utilities that are available to application programs on systems conformant to this part of the Single UNIX Specification. Readers are expected to be familiar with the XBD document.

This document is structured as follows:

- Chapter 1 explains the status of this document and its relationship to other formal standards, including the ISO C standard and also the XSH document. It also describes the utility limits, grammar conventions, defaults used by the utility descriptions, considerations for utilities in support of large files, and the list of required built-in utilities. The scope, conformance, and

2. The acronym “XCU” derives from the previous version of the specification which was called “Commands and Utilities”.

definitions sections are pointers to the XBD document; the sections are here to meet ISO/IEC rules regarding required sections. The terminology and portability codes are identical to the section in XBD and repeated here for ease of reference.

- Chapter 2 describes the command language—that is, the shell command language interpreter—used in systems conformant to the Single UNIX Specification.
- Chapter 3 describes a set of services and utilities that are implemented on systems supporting the Batch Environment option.
- Chapter 4 consists of reference pages for all utilities available on systems conforming to the Single UNIX Specification. These are in the form of reference pages and are organized alphabetically.

4.4 Rationale (XRAT)

The XRAT document is part of the Base Specifications, Issue 6. The XRAT document has been published to assist in the process of review and understanding of the main text. It contains historical information concerning the contents of the Base Specifications, Issue 6 and why features were included or discarded by the standard developers. It also contains notes of interest to application programmers on recommended programming practices, emphasizing the consequences of some aspects that may not be immediately apparent.

This document is organized in parallel to the normative documents of the Base Specification, with a separate part (Parts A, B, and C) for each of the three normative documents. In addition, two additional parts are included: Part D, Portability Considerations and Part E Subprofiling Considerations. The Portability Considerations chapter includes a report on the perceived user requirements for the Base Specification and how the facilities provided satisfy those requirements, together with guidance to writers of profiles on how to use the configurable options, limits, and optional behavior. The Subprofiling Considerations chapter satisfies the requirement that the document address subprofiling. This contains an example set of subprofiling options.

4.5 X/Open Curses (XCURSES)

XCURSES is not part of the Base Specifications, Issue 6. XCURSES describes a set of interfaces providing a terminal-independent method of updating character screens that are available to application programs on systems conformant to this part of the Single UNIX Specification. This document should be read in conjunction with The Open Group Corrigendum U056.

This document is structured as follows:

- Chapter 1 introduces Curses, gives an overview of enhancements that have been made to this version, and lists specific interfaces marked TO BE WITHDRAWN. This chapter also defines the requirements for conformance to this document and shows the generic format followed by interface definitions in Chapter 4.
- Chapter 2 describes the relationship between Curses and the C language, the compilation environment, and the X/Open System Interface (XSI) operating system requirements. It also defines the effect of the interface on the name space for identifiers and introduces the major data types that the interfaces use.

- Chapter 3 gives an overview of Curses. It discusses the use of some of the key data types and gives general rules for important common concepts such as characters, renditions, and window properties. It contains general rules for the common Curses operations and operating modes. This information is implicitly referenced by the interface definitions in Chapter 4. The chapter explains the system of naming the Curses functions and presents a table of function families. Finally, the chapter contains notes regarding use of macros and restrictions on block-mode terminals.
- Chapter 4 defines the Curses functional interfaces.
- Chapter 5 defines the contents of headers which declare constants, macros, and data structures that are needed by programs using the services provided by Chapter 4.
- Chapter 6 discusses the *terminfo* database which Curses uses to describe terminals. The chapter specifies the source format of a *terminfo* entry using a formal grammar, an informal discussion, and an example. Boolean, numeric, and string capabilities are presented in tabular form.
- Appendix A discusses the use of these capabilities by the writer of a *terminfo* entry to describe the characteristics of the terminal in use.
- The chapters are followed by a glossary, which contains normative definitions of terms used in the document.

System Interfaces

The Single UNIX Specification covers a broad spectrum of functionality, providing a rich environment for applications portability and portable applications development. This chapter covers the C language programming interfaces that are the system interfaces, describing the scope of the interfaces, an overview of the changes in this version, and lists of the new functions. It also groups the interfaces into categories to give the reader an understanding of what facilities are available.

5.1 Base Documents

The system interfaces provided in XSH, Issue 6 are drawn from the following base documents:

- IEEE Std 1003.1-1996 (POSIX.1) (incorporating IEEE Std 1003.1-1990 (POSIX.1), IEEE Std 1003.1b-1993, IEEE Std 1003.1c-1995, and IEEE Std 1003.1i-1995)

This is unchanged from the Single UNIX Specification, Version 2.

- The following amendments to IEEE Std 1003.1-1990 (POSIX.1):
 - IEEE P1003.1a draft standard (Additional System Services)
 - IEEE Std 1003.1d-1999 (Additional Realtime Extensions)
 - IEEE Std 1003.1g-2000 (Protocol-Independent Interfaces (PII))
 - IEEE Std 1003.1j-2000 (Advanced Realtime Extensions)
 - IEEE Std 1003.1q-2000 (Tracing)

The additions due to IEEE P1003.1a draft standard and IEEE Std 1003.1g-2000 have been added to the mandatory requirements for the Single UNIX Specification, Version 3. The additions from IEEE Std 1003.1d-1999, IEEE Std 1003.1j-2000, and IEEE Std 1003.1q-2000 have in general been added as options. The reader-writer locks from IEEE Std 1003.1j-2000 duplicate the reader-writer locks already contained in the Single UNIX Specification, Version 2 and are still mandatory.

- System Interface Definitions, Issue 5 (XBD5)
- System Interfaces and Headers, Issue 5 (XSH5) (in 2 Volumes)
- Networking Services, Issue 5.2 (XNS5.2)

This issue of the Networking Services specification superseded the version included in the Single UNIX Specification, Version 2. The main change was the addition of IPv6 functionality.

- ISO/IEC 9899: 1999, Programming Languages — C (ISO C)

This is the latest version of the C programming language standard and is a change from the Single UNIX Specification, Version 2.

5.2 Overview of Changes

The following sections describe changes made to XSH since Issue 5. Chapter 13 (on page 119) details the technical changes that have been made to the system interfaces from Issue 5.

Chapter 13 (on page 119) also details the changes to the POSIX base requirements.

Changes from Issue 5 to Issue 6

The following list summarizes the major changes that were made in XSH from Issue 5 to Issue 6:

- XSH is extensively revised so that it can be both an IEEE POSIX Standard and an Open Group Technical Standard.
- The POSIX System Interfaces requirements incorporate support of FIPS 151-2.
- The POSIX System Interfaces requirements are updated to align with some features of the Single UNIX Specification.
- A RATIONALE section is added to each reference page.
- Networking interfaces from Networking Services, Issue 5.2 (XNS5.2) are incorporated. The following parts of the XNS, Issue 5.2 specification are not included in this issue:
 - Part 3 (XTI)
 - Part 4 (Appendixes)
- Material from the following sections of IEEE Std 1003.1g-2000 has been included in this issue:
 - General terms related to sockets (Section 2.2.2)
 - Socket concepts (Sections 5.1 through 5.3, inclusive)
 - The *pselect()* function (Sections 6.2.2.1 and 6.2.3)
 - The *socketmark()* function (Section 5.4.13)
 - The **<sys/select.h>** header (Section 6.2)
- IEEE Std 1003.1d-1999 is incorporated.
- IEEE Std 1003.1j-2000 is incorporated.
- IEEE Std 1003.1q-2000 is incorporated.
- The IEEE P1003.1a draft standard is incorporated. The IEEE P1003.1a draft standard was withdrawn by the IEEE Standards Board upon the approval of the revision to IEEE Std 1003.1.
- Existing functionality is aligned with the ISO/IEC 9899:1999 standard.
- New functionality from the ISO/IEC 9899:1999 standard is incorporated.
- IEEE PASC Interpretations are applied. The final text for IEEE Interpretations can be located by following the links from www.pasc.org/interps.
- The Open Group Corrigenda and Resolutions are applied.

- From XSH, Issue 5 the following legacy interfaces, headers, and external variables have been removed in Issue 6:

advance(), *brk()*, *chroot()*, *compile()*, *cuserid()*, *gamma()*, *getdtablesize()*,
getpagesize(), *getpass()*, *getw()*, *putw()*, *re_comp()*, *re_exec()*, *regcmp()*, *sbrk()*,
sigstack(), *step()*, *wait3()*, **<re_comp.h>**, **<regex.h>**, **<varargs.h>**, *loc1*, *__loc1*, *loc2*,
locs

In addition, legacy features within non-legacy reference pages (for example, headers) are not carried forward.

- From ISO/IEC 9945-1:1996 (POSIX-1), the following obsolescent features are not carried forward in this issue:

Page 112, CLK_TCK
 Page 197, *tcgetattr()* rate returned option

- From XSH, Issue 5 text marked OH (Optional Header) has been reviewed on a case-by-case basis and removed where appropriate.
- A reorganization of the options in order to facilitate profiling has occurred, both for smaller profiles such as IEEE Std 1003.13-1998, and larger profiles such as the Single UNIX Specification.
- The **restrict** keyword from the ISO/IEC 9899:1999 standard has been added where applicable to function prototypes.

The rules for using **restrict** follow those of the ISO C standard. They are as follows:

- Only functions with two or more parameters need **restrict**.
- Additionally, at least two of the objects pointed to must have the same type; that is:

```
int foo (int *, double *)
```

 does not need a **restrict**.
- An exception to the above rule comes with pointers to structures. Since the Single UNIX Specification does not define the structures in all details, we must assume the worst case which is that it contains an element of the type described by one of the other parameters; that is:

```
int bar (int restrict *, struct baz restrict *)
```

 needs **restrict** since the structure **baz** can potentially contain an element of type **int**.

New Features in Issue 6

The functions first introduced in Issue 6 (over the XSH, Issue 5 and XNS, Issue 5 base documents) are listed in the table below:

New Functions in Issue 6		
<i>acosf()</i>	<i>catanhf()</i>	<i>cprojf()</i>
<i>acoshf()</i>	<i>catanl()</i>	<i>cprojl()</i>
<i>acoshl()</i>	<i>cbrtf()</i>	<i>creal()</i>
<i>acosl()</i>	<i>cbrtl()</i>	<i>crealf()</i>
<i>asinf()</i>	<i>ccos()</i>	<i>creall()</i>
<i>asinhf()</i>	<i>ccosf()</i>	<i>csin()</i>
<i>asinhl()</i>	<i>ccosh()</i>	<i>csinf()</i>
<i>asinl()</i>	<i>ccoshf()</i>	<i>csinh()</i>
<i>atan2f()</i>	<i>ccoshl()</i>	<i>csinhf()</i>
<i>atan2l()</i>	<i>ccosl()</i>	<i>csinhl()</i>
<i>atanf()</i>	<i>ceilf()</i>	<i>csinl()</i>
<i>atanhf()</i>	<i>ceill()</i>	<i>csqrt()</i>
<i>atanhl()</i>	<i>cexp()</i>	<i>csqrtf()</i>
<i>atanl()</i>	<i>cexpf()</i>	<i>csqrtl()</i>
<i>atoll()</i>	<i>cexpl()</i>	<i>ctan()</i>
<i>cabs()</i>	<i>cimag()</i>	<i>ctanf()</i>
<i>cabsf()</i>	<i>cimagf()</i>	<i>ctanh()</i>
<i>cabsl()</i>	<i>cimagl()</i>	<i>ctanhf()</i>
<i>cacos()</i>	<i>clock_getcpuclockid()</i>	<i>ctanhl()</i>
<i>cacosf()</i>	<i>clock_nanosleep()</i>	<i>ctanl()</i>
<i>cacosh()</i>	<i>clog()</i>	<i>erfcf()</i>
<i>cacoshf()</i>	<i>clogf()</i>	<i>erfcl()</i>
<i>cacoshl()</i>	<i>clogl()</i>	<i>erff()</i>
<i>cacosl()</i>	<i>conj()</i>	<i>erfl()</i>
<i>carg()</i>	<i>conjf()</i>	<i>exp2()</i>
<i>cargf()</i>	<i>conjl()</i>	<i>exp2f()</i>
<i>cargl()</i>	<i>copysign()</i>	<i>exp2l()</i>
<i>casin()</i>	<i>copysignf()</i>	<i>expf()</i>
<i>casinf()</i>	<i>copysignl()</i>	<i>expl()</i>
<i>casinh()</i>	<i>cosf()</i>	<i>expm1f()</i>
<i>casinhf()</i>	<i>coshf()</i>	<i>expm1l()</i>
<i>casinhl()</i>	<i>coshl()</i>	<i>fabsf()</i>
<i>casinl()</i>	<i>cosl()</i>	<i>fabsl()</i>
<i>catan()</i>	<i>cpow()</i>	<i>fdim()</i>
<i>catanf()</i>	<i>cpowf()</i>	<i>fdimf()</i>
<i>catanh()</i>	<i>cpowl()</i>	<i>fdiml()</i>
<i>catanhf()</i>	<i>cproj()</i>	<i>feclearexcept()</i>

New Functions in Issue 6		
<i>fegetenv()</i>	<i>ldexpl()</i>	<i>posix_fallocate()</i>
<i>fegetexceptflag()</i>	<i>lgammaf()</i>	<i>posix_madvise()</i>
<i>fegetround()</i>	<i>lgammal()</i>	<i>posix_mem_offset()</i>
<i>feholdexcept()</i>	<i>llabs()</i>	<i>posix_memalign()</i>
<i>feraiseexcept()</i>	<i>lldiv()</i>	<i>posix_openpt()</i>
<i>fesetenv()</i>	<i>llrint()</i>	<i>posix_spawn()</i>
<i>fesetexceptflag()</i>	<i>llrintf()</i>	<i>posix_spawn_file_actions_addclose()</i>
<i>fesetround()</i>	<i>llrintl()</i>	<i>posix_spawn_file_actions_adddup2()</i>
<i>fetestexcept()</i>	<i>llround()</i>	<i>posix_spawn_file_actions_addopen()</i>
<i>feupdateenv()</i>	<i>llroundf()</i>	<i>posix_spawn_file_actions_destroy()</i>
<i>floorf()</i>	<i>llroundl()</i>	<i>posix_spawn_file_actions_init()</i>
<i>floorl()</i>	<i>log10f()</i>	<i>posix_spawnattr_destroy()</i>
<i>fma()</i>	<i>log10l()</i>	<i>posix_spawnattr_getflags()</i>
<i>fmaf()</i>	<i>log1pf()</i>	<i>posix_spawnattr_getpgroup()</i>
<i>fmal()</i>	<i>log1pl()</i>	<i>posix_spawnattr_getschedparam()</i>
<i>fmax()</i>	<i>log2()</i>	<i>posix_spawnattr_getschedpolicy()</i>
<i>fmaxf()</i>	<i>log2f()</i>	<i>posix_spawnattr_getsigdefault()</i>
<i>fmaxl()</i>	<i>log2l()</i>	<i>posix_spawnattr_getsigmask()</i>
<i>fmin()</i>	<i>logbf()</i>	<i>posix_spawnattr_init()</i>
<i>fminf()</i>	<i>logbl()</i>	<i>posix_spawnattr_setflags()</i>
<i>fminl()</i>	<i>logf()</i>	<i>posix_spawnattr_setpgroup()</i>
<i>fmodf()</i>	<i>logl()</i>	<i>posix_spawnattr_setschedparam()</i>
<i>fmodl()</i>	<i>lrint()</i>	<i>posix_spawnattr_setschedpolicy()</i>
<i>fpclassify()</i>	<i>lrintf()</i>	<i>posix_spawnattr_setsigdefault()</i>
<i>frexpf()</i>	<i>lrintl()</i>	<i>posix_spawnattr_setsigmask()</i>
<i>frexpl()</i>	<i>lround()</i>	<i>posix_spawnnp()</i>
<i>hypotf()</i>	<i>lroundf()</i>	<i>posix_trace_attr_destroy()</i>
<i>hypotl()</i>	<i>lroundl()</i>	<i>posix_trace_attr_getclockres()</i>
<i>ilogbf()</i>	<i>modff()</i>	<i>posix_trace_attr_getcreatetime()</i>
<i>ilogbl()</i>	<i>modfl()</i>	<i>posix_trace_attr_getgenversion()</i>
<i>imaxabs()</i>	<i>mq_timedreceive()</i>	<i>posix_trace_attr_getinherited()</i>
<i>imaxdiv()</i>	<i>mq_timedsend()</i>	<i>posix_trace_attr_getlogfullpolicy()</i>
<i>isblank()</i>	<i>nan()</i>	<i>posix_trace_attr_getlogsize()</i>
<i>isfinite()</i>	<i>nanf()</i>	<i>posix_trace_attr_getmaxdatasize()</i>
<i>isgreater()</i>	<i>nanl()</i>	<i>posix_trace_attr_getmaxsystemeventsizesize()</i>
<i>isgreaterequal()</i>	<i>nearbyint()</i>	<i>posix_trace_attr_getmaxuserereventsizesize()</i>
<i>isinf()</i>	<i>nearbyintf()</i>	<i>posix_trace_attr_getname()</i>
<i>isless()</i>	<i>nearbyintl()</i>	<i>posix_trace_attr_getstreamfullpolicy()</i>
<i>islessequal()</i>	<i>nextafterf()</i>	<i>posix_trace_attr_getstreamsize()</i>
<i>islessgreater()</i>	<i>nextafterl()</i>	<i>posix_trace_attr_init()</i>
<i>isnormal()</i>	<i>nexttoward()</i>	<i>posix_trace_attr_setinherited()</i>
<i>isunordered()</i>	<i>nexttowardf()</i>	<i>posix_trace_attr_setlogfullpolicy()</i>
<i>iswblank()</i>	<i>nexttowardl()</i>	<i>posix_trace_attr_setlogsize()</i>
<i>ldexpf()</i>	<i>posix_fadvise()</i>	<i>posix_trace_create()</i>

New Functions in Issue 6		
<i>posix_trace_attr_setmaxdatasize()</i>	<i>pthread_barrier_destroy()</i>	<i>signbit()</i>
<i>posix_trace_attr_setname()</i>	<i>pthread_barrier_init()</i>	<i>sinf()</i>
<i>posix_trace_attr_setstreamfullpolicy()</i>	<i>pthread_barrier_wait()</i>	<i>sinhf()</i>
<i>posix_trace_attr_setstreamsizemax()</i>	<i>pthread_barrierattr_destroy()</i>	<i>sinhl()</i>
<i>posix_trace_clear()</i>	<i>pthread_barrierattr_getpshared()</i>	<i>sinl()</i>
<i>posix_trace_close()</i>	<i>pthread_barrierattr_init()</i>	<i>socketatmark()</i>
<i>posix_trace_create_withlog()</i>	<i>pthread_barrierattr_setpshared()</i>	<i>sqrtf()</i>
<i>posix_trace_event()</i>	<i>pthread_condattr_getclock()</i>	<i>sqrtl()</i>
<i>posix_trace_eventid_equal()</i>	<i>pthread_condattr_setclock()</i>	<i>strerror_r()</i>
<i>posix_trace_eventid_get_name()</i>	<i>pthread_getcpuclockid()</i>	<i>strtoimax()</i>
<i>posix_trace_eventid_open()</i>	<i>pthread_mutex_timedlock()</i>	<i>strtoll()</i>
<i>posix_trace_eventset_add()</i>	<i>pthread_rwlock_timedrdlock()</i>	<i>strtoull()</i>
<i>posix_trace_eventset_del()</i>	<i>pthread_rwlock_timedwrlock()</i>	<i>strtoumax()</i>
<i>posix_trace_eventset_empty()</i>	<i>pthread_setschedprio()</i>	<i>tanf()</i>
<i>posix_trace_eventset_fill()</i>	<i>pthread_spin_destroy()</i>	<i>tanhf()</i>
<i>posix_trace_eventset_ismember()</i>	<i>pthread_spin_init()</i>	<i>tanhl()</i>
<i>posix_trace_eventtypelist_getnext_id()</i>	<i>pthread_spin_lock()</i>	<i>tanl()</i>
<i>posix_trace_eventtypelist_rewind()</i>	<i>pthread_spin_trylock()</i>	<i>tgamma()</i>
<i>posix_trace_flush()</i>	<i>pthread_spin_unlock()</i>	<i>tgammaf()</i>
<i>posix_trace_get_attr()</i>	<i>remainderf()</i>	<i>tgammal()</i>
<i>posix_trace_get_filter()</i>	<i>remainderl()</i>	<i>trunc()</i>
<i>posix_trace_get_status()</i>	<i>remquo()</i>	<i>truncf()</i>
<i>posix_trace_getnext_event()</i>	<i>remquof()</i>	<i>truncl()</i>
<i>posix_trace_open()</i>	<i>remquol()</i>	<i>unsetenv()</i>
<i>posix_trace_rewind()</i>	<i>rintf()</i>	<i>vfprintf()</i>
<i>posix_trace_set_filter()</i>	<i>rintl()</i>	<i>vfscanf()</i>
<i>posix_trace_shutdown()</i>	<i>round()</i>	<i>vwscanf()</i>
<i>posix_trace_start()</i>	<i>roundf()</i>	<i>vprintf()</i>
<i>posix_trace_stop()</i>	<i>roundl()</i>	<i>vscanf()</i>
<i>posix_trace_timedgetnext_event()</i>	<i>scalbn()</i>	<i>vsprintf()</i>
<i>posix_trace_trid_eventid_open()</i>	<i>scalbnf()</i>	<i>vsscanf()</i>
<i>posix_trace_trygetnext_event()</i>	<i>scalbnl()</i>	<i>vswscanf()</i>
<i>posix_typed_mem_get_info()</i>	<i>scalbn()</i>	<i>vwscanf()</i>
<i>posix_typed_mem_open()</i>	<i>scalbnf()</i>	<i>wcstoimax()</i>
<i>powf()</i>	<i>scalbnl()</i>	<i>wcstoll()</i>
<i>powl()</i>	<i>sem_timedwait()</i>	<i>wcstoull()</i>
<i>pselect()</i>	<i>setegid()</i>	<i>wcstoumax()</i>
<i>pthread_attr_getstack()</i>	<i>setenv()</i>	
<i>pthread_attr_setstack()</i>	<i>seteuid()</i>	

The following table lists the functions and symbols from the XSI extension. These are new since ISO/IEC 9945-1:1996 (POSIX-1).

New XSI Functions and Symbols in Issue 6			
<i>_longjmp()</i>	<i>getcontext()</i>	<i>msgget()</i>	<i>setstate()</i>
<i>_setjmp()</i>	<i>getdate()</i>	<i>msgrcv()</i>	<i>setutxent()</i>
<i>_tolower()</i>	<i>getgrent()</i>	<i>msgsnd()</i>	<i>shmat()</i>
<i>_toupper()</i>	<i>gethostid()</i>	<i>nftw()</i>	<i>shmctl()</i>
<i>a64l()</i>	<i>getitimer()</i>	<i>nice()</i>	<i>shmdt()</i>
<i>basename()</i>	<i>getpgid()</i>	<i>nl_langinfo()</i>	<i>shmget()</i>
<i>bcmp()</i>	<i>getpmsg()</i>	<i>nrand48()</i>	<i>sigaltstack()</i>
<i>bcopy()</i>	<i>getpriority()</i>	<i>openlog()</i>	<i>sighold()</i>
<i>bzero()</i>	<i>getpwent()</i>	<i>poll()</i>	<i>sigignore()</i>
<i>catclose()</i>	<i>getrlimit()</i>	<i>posix_openpt()</i>	<i>siginterrupt()</i>
<i>catgets()</i>	<i>getrusage()</i>	<i>pread()</i>	<i>sigpause()</i>
<i>catopen()</i>	<i>getsid()</i>	<i>pthread_attr_getguardsize()</i>	<i>sigrelse()</i>
<i>closelog()</i>	<i>getsubopt()</i>	<i>pthread_attr_setguardsize()</i>	<i>sigset()</i>
<i>crypt()</i>	<i>gettimeofday()</i>	<i>pthread_attr_setstack()</i>	<i>srand48()</i>
<i>daylight</i>	<i>getutxent()</i>	<i>pthread_getconcurrency()</i>	<i>srandom()</i>
<i>dbm_clearerr()</i>	<i>getutxid()</i>	<i>pthread_mutexattr_gettype()</i>	<i>statvfs()</i>
<i>dbm_close()</i>	<i>getutxline()</i>	<i>pthread_mutexattr_settype()</i>	<i>strcasecmp()</i>
<i>dbm_delete()</i>	<i>getwd()</i>	<i>pthread_rwlockattr_init()</i>	<i>strdup()</i>
<i>dbm_error()</i>	<i>grantpt()</i>	<i>pthread_rwlockattr_setpshared()</i>	<i>strfmon()</i>
<i>dbm_fetch()</i>	<i>hcreate()</i>	<i>pthread_setconcurrency()</i>	<i>strncasecmp()</i>
<i>dbm_firstkey()</i>	<i>hdestroy()</i>	<i>ptsname()</i>	<i>strptime()</i>
<i>dbm_nextkey()</i>	<i>hsearch()</i>	<i>putenv()</i>	<i>swab()</i>
<i>dbm_open()</i>	<i>iconv()</i>	<i>pututxline()</i>	<i>swapcontext()</i>
<i>dbm_store()</i>	<i>iconv_close()</i>	<i>pwrite()</i>	<i>sync()</i>
<i>dirname()</i>	<i>iconv_open()</i>	<i>random()</i>	<i>syslog()</i>
<i>dlclose()</i>	<i>index()</i>	<i>readv()</i>	<i>tcgetsid()</i>
<i>dlderror()</i>	<i>initstate()</i>	<i>realpath()</i>	<i>tdelete()</i>
<i>dlopen()</i>	<i>insque()</i>	<i>remque()</i>	<i>telldir()</i>
<i>dlsym()</i>	<i>isascii()</i>	<i>rindex()</i>	<i>tempnam()</i>
<i>drand48()</i>	<i>jrand48()</i>	<i>seed48()</i>	<i>tfind()</i>
<i>ecvt()</i>	<i>killpg()</i>	<i>seekdir()</i>	<i>timezone</i>
<i>encrypt()</i>	<i>l64a()</i>	<i>semctl()</i>	<i>toascii()</i>
<i>endgrent()</i>	<i>lchown()</i>	<i>semget()</i>	<i>truncate()</i>
<i>endpwent()</i>	<i>lcong48()</i>	<i>semop()</i>	<i>tsearch()</i>
<i>endutxent()</i>	<i>lfind()</i>	<i>setcontext()</i>	<i>twalk()</i>
<i>erand48()</i>	<i>lockf()</i>	<i>setgrent()</i>	<i>ulimit()</i>
<i>fchdir()</i>	<i>lrand48()</i>	<i>setitimer()</i>	<i>unlockpt()</i>
<i>fcvt()</i>	<i>lsearch()</i>	<i>setkey()</i>	<i>utimes()</i>
<i>ffs()</i>	<i>makecontext()</i>	<i>setlogmask()</i>	<i>waitid()</i>
<i>fmtmsg()</i>	<i>memccpy()</i>	<i>setpggrp()</i>	<i>wcswcs()</i>
<i>fstatvfs()</i>	<i>mknod()</i>	<i>setpriority()</i>	<i>wcswidth()</i>
<i>ftime()</i>	<i>mkstemp()</i>	<i>setpwent()</i>	<i>wcwidth()</i>
<i>ftok()</i>	<i>mktemp()</i>	<i>setregid()</i>	<i>writev()</i>
<i>ftw()</i>	<i>mrand48()</i>	<i>setreuid()</i>	
<i>gcvt()</i>	<i>msgctl()</i>	<i>setrlimit()</i>	

5.3 System Interfaces by Category

The Single UNIX Specification, Version 3 supports and is aligned with key formal standards, of which IEEE Std 1003.1-2001 forms the core. This section gives lists of interfaces categorized by functional grouping:

Jump Interfaces

longjmp(), *setjmp()*

Maths Library Interfaces

acos(), *acosf()*, *acosh()*, *acoshf()*, *acoshl()*, *acosl()*, *asin()*, *asinf()*, *asinh()*, *asinhf()*, *asinhhl()*, *asinhl()*, *atan()*, *atan2()*, *atan2f()*, *atan2l()*, *atanf()*, *atanh()*, *atanhf()*, *atanhl()*, *atanl()*, *cabs()*, *cabsf()*, *cabsl()*, *cacos()*, *cacosf()*, *cacosh()*, *cacoshf()*, *cacoshl()*, *cacosl()*, *carg()*, *cargf()*, *cargl()*, *casin()*, *casinf()*, *casinh()*, *casinhf()*, *casinhl()*, *casinl()*, *catan()*, *catanf()*, *catanh()*, *catanhf()*, *catanhl()*, *catanl()*, *cbrt()*, *cbrtf()*, *cbrtl()*, *ccos()*, *ccosf()*, *ccosh()*, *ccoshf()*, *ccoshl()*, *ccosl()*, *ceil()*, *ceilf()*, *ceill()*, *cexp()*, *cexpf()*, *cexpl()*, *cimag()*, *cimagf()*, *cimagl()*, *clog()*, *clogf()*, *clogl()*, *conj()*, *conjf()*, *conjl()*, *copysign()*, *copysignf()*, *copysignl()*, *cos()*, *cosf()*, *cosh()*, *coshf()*, *coshl()*, *cosl()*, *cpow()*, *cpowf()*, *cpowl()*, *cproj()*, *cprojf()*, *cprojl()*, *creal()*, *crealf()*, *creall()*, *csin()*, *csinf()*, *csinh()*, *csinhf()*, *csinhl()*, *csinl()*, *csqrt()*, *csqrtf()*, *csqrtl()*, *ctan()*, *ctanf()*, *ctanh()*, *ctanhf()*, *ctanhl()*, *ctanl()*, *erf()*, *erfc()*, *erfcf()*, *erfcl()*, *erff()*, *erfl()*, *exp()*, *exp2()*, *exp2f()*, *exp2l()*, *expf()*, *expl()*, *expm1()*, *expm1f()*, *expm1l()*, *fabs()*, *fabsf()*, *fabsl()*, *fdim()*, *fdimf()*, *fdiml()*, *floor()*, *floorf()*, *floorl()*, *fma()*, *fmaf()*, *fmal()*, *fmax()*, *fmaxf()*, *fmaxl()*, *fmin()*, *fminf()*, *fminl()*, *fmod()*, *fmodf()*, *fmodl()*, *fpclassify()*, *frexp()*, *frexpf()*, *frexpl()*, *hypot()*, *hypotf()*, *hypotl()*, *ilogb()*, *ilogbf()*, *ilogbl()*, *isfinite()*, *isgreater()*, *isgreaterequal()*, *isinf()*, *isless()*, *islessequal()*, *islessgreater()*, *isnan()*, *isnormal()*, *isunordered()*, *ldexp()*, *ldexpf()*, *ldexpl()*, *lgamma()*, *lgammaf()*, *lgammal()*, *llrint()*, *llrintf()*, *llrintl()*, *llround()*, *llroundf()*, *llroundl()*, *log()*, *log10()*, *log10f()*, *log10l()*, *log1p()*, *log1pf()*, *log1pl()*, *log2()*, *log2f()*, *log2l()*, *logb()*, *logbf()*, *logbl()*, *logf()*, *logl()*, *lrint()*, *lrintf()*, *lrintl()*, *lround()*, *lroundf()*, *lroundl()*, *modf()*, *modff()*, *modfl()*, *nan()*, *nanf()*, *nanl()*, *nearbyint()*, *nearbyintf()*, *nearbyintl()*, *nextafter()*, *nextafterf()*, *nextafterl()*, *nexttoward()*, *nexttowardf()*, *nexttowardl()*, *pow()*, *powf()*, *powl()*, *remainder()*, *remainderf()*, *remainderl()*, *remquo()*, *remquof()*, *remquol()*, *rint()*, *rintf()*, *rintl()*, *round()*, *roundf()*, *roundl()*, *scalbln()*, *scalblnf()*, *scalblnl()*, *scalbn()*, *scalbnf()*, *scalbnl()*, *signbit()*, *sin()*, *sinf()*, *sinh()*, *sinhf()*, *sinhl()*, *sinl()*, *sqrt()*, *sqrtf()*, *sqrtl()*, *tan()*, *tanf()*, *tanh()*, *tanhf()*, *tanhl()*, *tanl()*, *tgamma()*, *tgammaf()*, *tgammaal()*, *trunc()*, *truncf()*, *truncl()*

General ISO C Library Interfaces

abs(), *asctime()*, *atoi()*, *atol()*, *atoll()*, *bsearch()*, *calloc()*, *ctime()*, *difftime()*, *div()*, *feclearexcept()*, *fegetenv()*, *fegetexceptflag()*, *fegetround()*, *feholdexcept()*, *feraiseexcept()*, *fesetenv()*, *fesetexceptflag()*, *fesetround()*, *fetestexcept()*, *feupdateenv()*, *free()*, *gmtime()*, *imaxabs()*, *imaxdiv()*, *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *isxdigit()*, *labs()*, *ldiv()*, *llabs()*, *lldiv()*, *localeconv()*, *localtime()*, *malloc()*, *memchr()*, *memcmp()*, *memcpy()*, *memmove()*, *memset()*, *mktime()*, *qsort()*, *rand()*, *realloc()*, *setlocale()*, *snprintf()*, *sprintf()*, *srand()*, *sscanf()*, *strcat()*, *strchr()*, *strcmp()*, *strcoll()*, *strcpy()*, *strcspn()*, *strerror()*, *strptime()*, *strlen()*, *strncat()*, *strncmp()*, *strncpy()*, *strpbrk()*, *strrchr()*, *strspn()*, *strstr()*, *strtod()*, *strtof()*, *strtoimax()*, *strtok()*, *strtol()*, *strtold()*, *strtoll()*, *strtoul()*, *strtoull()*, *strtoumax()*, *strxfrm()*, *time()*, *tolower()*, *toupper()*, *tzname*, *tzset()*, *va_arg()*, *va_copy()*, *va_end()*, *va_start()*, *vsprintf()*, *vsscanf()*

Thread-Safe General ISO C Library Interfaces

asctime_r(), *ctime_r()*, *gmtime_r()*, *localtime_r()*, *rand_r()*, *strerror_r()*, *strtok_r()*

Wide-Character ISO C Library Interfaces

btowc(), *iswalnum()*, *iswalpha()*, *iswblank()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *mblen()*, *mbrlen()*, *mbrtowc()*, *mbsinit()*, *mbsrtowcs()*, *mbstowcs()*, *mbtowc()*, *swprintf()*, *swscanf()*, *towctrans()*, *towlower()*, *toupper()*, *vswprintf()*, *vswscanf()*, *wcrtomb()*, *wcscat()*, *wcschr()*, *wcscmp()*, *wcscoll()*, *wcscopy()*, *wcscspn()*, *wcsftime()*, *wcslen()*, *wcsncat()*, *wcsncmp()*, *wcsncpy()*, *wcspbrk()*, *wcsrchr()*, *wcsrtombs()*, *wcsspn()*, *wcsstr()*, *wcstod()*, *wcstof()*, *wcstoimax()*, *wcstok()*, *wcstol()*, *wcstold()*, *wcstoll()*, *wcstombs()*, *wcstoul()*, *wcstoull()*, *wcstoumax()*, *wcsxfrm()*, *wctob()*, *wctomb()*, *wctrans()*, *wctype()*, *wmemchr()*, *wmemcmp()*, *wmemcpy()*, *wmemmove()*, *wmemset()*

General C Library Extension Interfaces

fnmatch(), *getopt()*, *optarg*, *opterr*, *optind*, *optopt*

Device Input and Output Interfaces

FD_CLR(), *FD_ISSET()*, *FD_SET()*, *FD_ZERO()*, *clearerr()*, *close()*, *fclose()*, *fdopen()*, *feof()*, *ferror()*, *fflush()*, *fgetc()*, *fgets()*, *fileno()*, *fopen()*, *fprintf()*, *fputc()*, *fputs()*, *fread()*, *freopen()*, *fscanf()*, *fwrite()*, *getc()*, *getchar()*, *gets()*, *open()*, *perror()*, *printf()*, *pselect()*, *putc()*, *putchar()*, *puts()*, *read()*, *scanf()*, *select()*, *setbuf()*, *setvbuf()*, *stderr*, *stdin*, *stdout*, *ungetc()*, *vfprintf()*, *vscanf()*, *vprintf()*, *vscanf()*, *write()*

General Terminal Interfaces

cfgetispeed(), *cfgetospeed()*, *cfsetispeed()*, *cfsetospeed()*, *ctermid()*, *isatty()*, *tcdrain()*, *tcflow()*, *tcflush()*, *tcgetattr()*, *tcsendbreak()*, *tcsetattr()*, *ttyname()*

Thread-Safe General Terminal Interfaces

ttyname_r()

File Descriptor Management Interfaces

dup(), *dup2()*, *fcntl()*, *fgetpos()*, *fseek()*, *fseeko()*, *fsetpos()*, *ftell()*, *ftello()*, *ftruncate()*, *lseek()*, *rewind()*

FIFO Interfaces

mkfifo()

File Attributes Interfaces

chmod(), *chown()*, *fchmod()*, *fchown()*, *umask()*

Thread-Safe Stdio Locking Interfaces

flockfile(), *ftrylockfile()*, *funlockfile()*, *getc_unlocked()*, *getchar_unlocked()*, *putc_unlocked()*, *putchar_unlocked()*

File System Interfaces

access(), *chdir()*, *closedir()*, *creat()*, *fpathconf()*, *fstat()*, *getcwd()*, *link()*, *mkdir()*, *opendir()*, *pathconf()*, *readdir()*, *remove()*, *rename()*, *rewinddir()*, *rmdir()*, *stat()*, *tmpfile()*, *tmpnam()*, *unlink()*, *utime()*

File System Extensions Interfaces

glob(), *globfree()*

Thread-Safe File System Interfaces

readdir_r()

Job Control Interfaces

setpgid(), *tcgetpgrp()*, *tcsetpgrp()*

Multiple Processes Interfaces

_Exit(), *_exit()*, *assert()*, *atexit()*, *clock()*, *execl()*, *execle()*, *execlp()*, *execv()*, *execve()*, *execvp()*, *exit()*, *fork()*, *getpgrp()*, *getpid()*, *getppid()*, *setsid()*, *sleep()*, *times()*, *wait()*, *waitpid()*

Networking Interfaces

accept(), *bind()*, *connect()*, *endhostent()*, *endnetent()*, *endprotoent()*, *endservent()*, *freeaddrinfo()*, *gai_strerror()*, *getaddrinfo()*, *gethostbyaddr()*, *gethostbyname()*, *gethostent()*, *gethostname()*, *getnameinfo()*, *getnetbyaddr()*, *getnetbyname()*, *getnetent()*, *getpeername()*, *getprotobyname()*, *getprotobynumber()*, *getprotoent()*, *getservbyname()*, *getservbyport()*, *getservent()*, *getsockname()*, *getsockopt()*, *h_errno*, *htonl()*, *htons()*, *if_freenameindex()*, *if_indextoname()*, *if_nameindex()*, *if_nametoindex()*, *inet_addr()*, *inet_ntoa()*, *inet_ntop()*, *inet_pton()*, *listen()*, *ntohl()*, *ntohs()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *sethostent()*, *setnetent()*, *setprotoent()*, *setservent()*, *setsockopt()*, *shutdown()*, *socket()*, *socketatmark()*, *socketpair()*

Pipe Interfaces

pipe()

Regular Expressions Interfaces

regcomp(), *regerror()*, *regexexec()*, *regfree()*

Shell and Utilities Interfaces

pclose(), *popen()*, *system()*, *wordexp()*, *wordfree()*

Signal Interfaces

abort(), *alarm()*, *kill()*, *pause()*, *raise()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *signal()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigwait()*

Signal Jump Functions Interfaces

siglongjmp(), *sigsetjmp()*

Single Process Interfaces

confstr(), *environ*, *errno*, *getenv()*, *setenv()*, *sysconf()*, *uname()*, *unsetenv()*

Symbolic Links Interfaces

lstat(), *readlink()*, *symlink()*

System Database Interfaces

getgrgid(), *getgrnam()*, *getpwnam()*, *getpwuid()*

Thread-Safe System Database Interfaces

getgrgid_r(), *getgrnam_r()*, *getpwnam_r()*, *getpwuid_r()*

Threads Interfaces

pthread_addr_setstacksize(), *pthread_atfork()*, *pthread_attr_destroy()*, *pthread_attr_getdetachstate()*, *pthread_attr_getstackaddr()*, *pthread_attr_getstacksize()*, *pthread_attr_init()*, *pthread_attr_setdetachstate()*, *pthread_attr_setschedparam()*, *pthread_attr_setstackaddr()*, *pthread_barrierattr_destroy()*, *pthread_barrierattr_getpshared()*, *pthread_barrierattr_init()*, *pthread_barrierattr_setpshared()*, *pthread_barrier_destroy()*, *pthread_barrier_init()*, *pthread_barrier_wait()*, *pthread_cancel()*, *pthread_cleanup_pop()*, *pthread_cleanup_push()*, *pthread_cond_broadcast()*, *pthread_cond_destroy()*, *pthread_cond_init()*, *pthread_cond_signal()*, *pthread_cond_timedwait()*, *pthread_cond_wait()*, *pthread_condattr_destroy()*, *pthread_condattr_getpshared()*, *pthread_condattr_init()*, *pthread_condattr_setpshared()*, *pthread_create()*,

pthread_detach(), *pthread_equal()*, *pthread_exit()*, *pthread_getschedparam()*,
pthread_getspecific(), *pthread_join()*, *pthread_key_create()*, *pthread_key_delete()*,
pthread_kill(), *pthread_mutex_destroy()*, *pthread_mutex_init()*, *pthread_mutex_lock()*,
pthread_mutex_timedlock(), *pthread_mutex_trylock()*, *pthread_mutex_unlock()*,
pthread_mutexattr_destroy(), *pthread_mutexattr_getpshared()*, *pthread_mutexattr_init()*,
pthread_mutexattr_setpshared(), *pthread_once()*, *pthread_rwlockattr_destroy()*,
pthread_rwlockattr_getpshared(), *pthread_rwlockattr_init()*,
pthread_rwlockattr_setpshared(), *pthread_rwlock_destroy()*, *pthread_rwlock_init()*,
pthread_rwlock_rdlock(), *pthread_rwlock_timedrdlock()*, *pthread_rwlock_timedwrlock()*,
pthread_rwlock_tryrdlock(), *pthread_rwlock_trywrlock()*, *pthread_rwlock_unlock()*,
pthread_rwlock_wrlock(), *pthread_self()*, *pthread_setcancelstate()*,
pthread_setcanceltype(), *pthread_setspecific()*, *pthread_sigmask()*,
pthread_spin_destroy(), *pthread_spin_init()*, *pthread_spin_lock()*, *pthread_spin_trylock()*,
pthread_spin_unlock(), *pthread_testcancel()*

Realtime Threads Interfaces

pthread_attr_getinheritsched(), *pthread_attr_getschedpolicy()*, *pthread_attr_getscope()*,
pthread_attr_setinheritsched(), *pthread_attr_setschedpolicy()*, *pthread_attr_setscope()*,
pthread_getschedparam(), *pthread_mutex_getprioceiling()*,
pthread_mutex_setprioceiling(), *pthread_mutexattr_getprioceiling()*,
pthread_mutexattr_getprotocol(), *pthread_mutexattr_setprioceiling()*,
pthread_mutexattr_setprotocol(), *pthread_setschedparam()*

Realtime Interfaces

aio_cancel(), *aio_error()*, *aio_fsync()*, *aio_read()*, *aio_return()*, *aio_suspend()*,
aio_write(), *clock_getres()*, *clock_gettime()*, *clock_settime()*, *fdatasync()*, *lio_listio()*,
mlock(), *mlockall()*, *mq_close()*, *mq_getattr()*, *mq_notify()*, *mq_open()*, *mq_receive()*,
mq_send(), *mq_setattr()*, *mq_timedreceive()*, *mq_timedsend()*, *mq_unlink()*, *munlock()*,
munlockall(), *nanosleep()*, *sched_get_priority_max()*, *sched_get_priority_min()*,
sched_getparam(), *sched_getscheduler()*, *sched_rr_get_interval()*, *sched_setparam()*,
sched_setscheduler(), *sched_yield()*, *sem_close()*, *sem_destroy()*, *sem_getvalue()*,
sem_init(), *sem_open()*, *sem_post()*, *sem_timedwait()*, *sem_trywait()*, *sem_unlink()*,
sem_wait(), *shm_open()*, *shm_unlink()*, *sigqueue()*, *sigtimedwait()*, *sigwaitinfo()*,
timer_create(), *timer_delete()*, *timer_getoverrun()*, *timer_gettime()*, *timer_settime()*

Tracing Interfaces

posix_trace_attr_destroy(), *posix_trace_attr_getclockres()*,
posix_trace_attr_getcreatetime(), *posix_trace_attr_getgenversion()*,
posix_trace_attr_getinherited(), *posix_trace_attr_getlogfullpolicy()*,
posix_trace_attr_getlogssize(), *posix_trace_attr_getmaxdatasize()*,
posix_trace_attr_getmaxsystemeventsizesize(), *posix_trace_attr_getmaxusereventsizesize()*,
posix_trace_attr_getname(), *posix_trace_attr_getstreamfullpolicy()*,
posix_trace_attr_getstreamsize(), *posix_trace_attr_init()*, *posix_trace_attr_setinherited()*,
posix_trace_attr_setlogfullpolicy(), *posix_trace_attr_setlogssize()*,
posix_trace_attr_setmaxdatasize(), *posix_trace_attr_setname()*,
posix_trace_attr_setstreamfullpolicy(), *posix_trace_attr_setstreamsize()*,
posix_trace_clear(), *posix_trace_close()*, *posix_trace_create()*,
posix_trace_create_withlog(), *posix_trace_event()*, *posix_trace_eventid_equal()*,
posix_trace_eventid_get_name(), *posix_trace_eventid_open()*,
posix_trace_eventset_add(), *posix_trace_eventset_del()*, *posix_trace_eventset_empty()*,
posix_trace_eventset_fill(), *posix_trace_eventset_ismember()*,
posix_trace_eventtypelist_getnext_id(), *posix_trace_eventtypelist_rewind()*,
posix_trace_flush(), *posix_trace_get_attr()*, *posix_trace_get_filter()*,
posix_trace_getnext_event(), *posix_trace_get_status()*, *posix_trace_open()*,

posix_trace_rewind(), *posix_trace_set_filter()*, *posix_trace_shutdown()*,
posix_trace_start(), *posix_trace_stop()*, *posix_trace_timedgetnext_event()*,
posix_trace_trid_eventid_open(), *posix_trace_trygetnext_event()*

Advisory Interfaces Interfaces

posix_fadvise(), *posix_fallocate()*, *posix_madvise()*

Typed Memory Interfaces Interfaces

posix_mem_offset(), *posix_typed_mem_get_info()*, *posix_typed_mem_open()*

User and Group Interfaces

getgid(), *geteuid()*, *getgid()*, *getgroups()*, *getlogin()*, *getuid()*, *setegid()*, *seteuid()*,
setgid(), *setuid()*

Thread-Safe User and Group Interfaces

getlogin_r()

Wide Character Device Input and Output Interfaces

fgetwc(), *fgetws()*, *fputwc()*, *fputws()*, *fwide()*, *fwprintf()*, *fwscanf()*, *getwc()*, *getwchar()*,
putwc(), *putwchar()*, *ungetwc()*, *vfwprintf()*, *vfwscanf()*, *vwprintf()*, *vwscanf()*, *wprintf()*,
wscanf()

XSI General C Library Interfaces

_tolower(), *_toupper()*, *a64l()*, *daylight()*, *drand48()*, *erand48()*, *ffs()*, *getcontext()*,
getdate(), *getsubopt()*, *hcreate()*, *hdestroy()*, *hsearch()*, *iconv()*, *iconv_close()*,
iconv_open(), *initstate()*, *insque()*, *isascii()*, *jrand48()*, *l64a()*, *lcong48()*, *lfind()*,
lrand48(), *lsearch()*, *makecontext()*, *memccpy()*, *mrnd48()*, *nrnd48()*, *random()*,
remque(), *seed48()*, *setcontext()*, *setstate()*, *siggam*, *srnd48()*, *srandom()*,
strcasecmp(), *strdup()*, *strfmon()*, *strncasecmp()*, *strptime()*, *swab()*, *swapcontext()*,
tdelete(), *tfind()*, *timezone()*, *toascii()*, *tsearch()*, *twalk()*

XSI Encryption Interfaces

crypt(), *encrypt()*, *setkey()*

XSI Database Management Interfaces

dbm_clearerr(), *dbm_close()*, *dbm_delete()*, *dbm_error()*, *dbm_fetch()*, *dbm_firstkey()*,
dbm_nextkey(), *dbm_open()*, *dbm_store()*

XSI Device Input and Output Interfaces

fmsg(), *poll()*, *pread()*, *pwrite()*, *readv()*, *writev()*

XSI General Terminal Interfaces

grantpt(), *posix_openpt()*, *ptsname()*, *unlockpt()*

XSI Dynamic Linking Interfaces

dlclose(), *dlderror()*, *dlopen()*, *dlsym()*

XSI File Descriptor Management Interfaces

truncate()

XSI File System Interfaces

basename(), *dirname()*, *fchdir()*, *fstatvfs()*, *ftw()*, *lchown()*, *lockf()*, *mknod()*, *mkstemp()*,
nftw(), *realpath()*, *seekdir()*, *statvfs()*, *sync()*, *telldir()*, *tempnam()*

XSI Internationalization Interfaces

catclose(), *catgets()*, *catopen()*, *nl_langinfo()*

XSI Interprocess Communication Interfaces

flok(), *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*, *semctl()*, *semget()*, *semop()*, *shmat()*,
shmctl(), *shmdt()*, *shmget()*

XSI Job Control Interfaces

tcgetsid()

XSI Jump Functions Interfaces

_longjmp(), *_setjmp()*

XSI Maths Library Interfaces

j0(), *j1()*, *jn()*, *scalb()*, *y0()*, *y1()*, *yn()*

XSI Multiple Process Interfaces

getpgid(), *getpriority()*, *getrlimit()*, *getrusage()*, *getsid()*, *nice()*, *setpgrp()*, *setpriority()*, *setrlimit()*, *ulimit()*, *usleep()*, *vfork()*, *waitid()*

XSI Signal Interfaces

bsd_signal(), *killpg()*, *sigaltstack()*, *sighold()*, *sigignore()*, *siginterrupt()*, *sigpause()*, *sigrelse()*, *sigset()*, *ualarm()*

XSI Single Process Interfaces

gethostid(), *gettimeofday()*, *putenv()*

XSI System Database Interfaces

endpwent(), *getpwent()*, *setpwent()*

XSI System Logging Interfaces

closelog(), *openlog()*, *setlogmask()*, *syslog()*

XSI Thread Mutex Extensions Interfaces

pthread_mutexattr_gettype(), *pthread_mutexattr_settype()*

XSI Threads Extensions Interfaces

pthread_attr_getguardsize(), *pthread_attr_setguardsize()*, *pthread_getconcurrency()*, *pthread_setconcurrency()*

XSI Timers Interfaces

getitimer(), *setitimer()*

XSI User and Group Interfaces

endgrent(), *endutxent()*, *getgrent()*, *getutxent()*, *getutxid()*, *getutxline()*, *pututxline()*, *setgrent()*, *setregid()*, *setreuid()*, *setutxent()*

XSI Wide-Character Library Interfaces

wcswidth(), *wcwidth()*

XSI Legacy Interfaces

bcmp(), *bcopy()*, *bzero()*, *ecvt()*, *fcvt()*, *ftime()*, *gcvt()*, *getwd()*, *index()*, *mktemp()*, *rindex()*, *utimes()*, *wcswcs()*

5.4 XSH Option Groups

The Single UNIX Specification includes a set of profiling options, allowing larger profiles of the options of the Base standard. In earlier versions of the Single UNIX Specification these were known as Feature Groups. The Option Groups within the Single UNIX Specification are defined within XBD, Section 2.1.5.2, XSI Option Groups.

The Single UNIX Specification contains the following Option Groups relating to XSH:

- **Encryption**, covering the functions *crypt()*, *encrypt()*, and *setkey()*
- **Realtime**, covering the functions from the IEEE Std 1003.1b-1993 Realtime extension

This Option Group consists of the set of the following options from within IEEE Std 1003.1-2001:

```
_POSIX_ASYNCHRONOUS_IO
_POSIX_FSYNC
_POSIX_MAPPED_FILES
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MEMORY_PROTECTION
_POSIX_MESSAGE_PASSING
_POSIX_PRIORITIZED_IO
_POSIX_PRIORITY_SCHEDULING
_POSIX_REALTIME_SIGNALS
_POSIX_SEMAPHORES
_POSIX_SHARED_MEMORY_OBJECTS
_POSIX_SYNCHRONIZED_IO
_POSIX_TIMERS
```

Where entire functions are included in the Option Group, the NAME section is marked with REALTIME. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within IEEE Std 1003.1-2001.

- **Realtime Threads**, covering the functions from the IEEE Std 1003.1c-1995 Threads extension that are related to realtime functionality

This Option Group consists of the set of the following options from within IEEE Std 1003.1-2001:

```
_POSIX_THREAD_PRIO_INHERIT
_POSIX_THREAD_PRIO_PROTECT
_POSIX_THREAD_PRIORITY_SCHEDULING
```

Where entire functions are included in the Option Group, the NAME section is marked with REALTIME THREADS. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within IEEE Std 1003.1-2001.

- **Advanced Realtime**, covering some of the non-threads-related functions from IEEE Std 1003.1d-1999 and IEEE Std 1003.1j-2000

This Option Group consists of the set of the following options from within IEEE Std 1003.1-2001:

```
_POSIX_ADVISORY_INFO
_POSIX_CLOCK_SELECTION
_POSIX_CPUTIME
_POSIX_MONOTONIC_CLOCK
_POSIX_SPAWN
_POSIX_SPORADIC_SERVER
_POSIX_TIMEOUTS
_POSIX_TYPED_MEMORY_OBJECTS
```

Where entire functions are included in the Option Group, the NAME section is marked with ADVANCED REALTIME. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within IEEE Std 1003.1-2001.

- **Advanced Realtime Threads**, covering some of the threads-related functions from IEEE Std 1003.1d-1999 and IEEE Std 1003.1j-2000

This Option Group consists of the set of the following options from within IEEE Std 1003.1-2001:

```
_POSIX_BARRIERS
_POSIX_SPIN_LOCKS
_POSIX_THREAD_CPUTIME
_POSIX_THREAD_SPORADIC_SERVER
```

Where entire functions are included in the Option Group, the NAME section is marked with ADVANCED REALTIME THREADS. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within IEEE Std 1003.1-2001.

- **Tracing**, covering the functionality from IEEE Std 1003.1q-2000

This Option Group consists of the set of the following options from within IEEE Std 1003.1-2001:

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_LOG
_POSIX_TRACE_INHERIT
```

Where entire functions are included in the Option Group, the NAME section is marked with TRACING. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within IEEE Std 1003.1-2001.

- **XSI STREAMS**, covering the functionality and interfaces related to STREAMS, a uniform mechanism for implementing networking services and other character-based I/O as described in XSH, Section 2.6, STREAMS

This was mandatory in previous versions of the Single UNIX Specification, but is optional in this version. It includes the following functions:

```
fattach(), fdetach(), getmsg(), getpmsg(), ioctl(), isastream(), putmsg(), putpmsg()
```

and the **<stropts.h>** header.

Where applicable, whole pages are marked STREAMS, together with the appropriate option margin legend for the SYNOPSIS section. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within IEEE Std 1003.1-2001.

- **Legacy**, covering the functionality and interfaces which were mandatory in previous versions of the Single UNIX Specification, but are optional in this version

Various factors may have contributed to the decision to mark a function or header LEGACY. In all cases, the specific reasons for the withdrawal of a function or header are documented on the relevant pages.

Once a function or header is marked LEGACY, no modifications are made to the specifications of such functions or headers other than to the APPLICATION USAGE sections of the relevant pages.

The functions and headers which form this Option Group are as follows:

bcmp(), *bcopy()*, *bzero()*, *ecvt()*, *fcvt()*, *ftime()*, *gcvt()*, *getwd()*, *index()*, *mktemp()*, *rindex()*, *utimes()*, *wcswcs()*

5.5 Options Policy

XSH contains a new policy for options handling. In previous versions of XSH, conforming systems had to provide headers, and stub functions returning the [ENOSYS] error condition for unsupported options and functions, if present, had to be conforming even if the option was not claimed to be supported.

The policy in this version of the specification provides flexibility for implementations in how they support options. It also specifies how conforming applications can adapt to different implementations that support different sets of options. It allows the following:

1. If an implementation has no interest in supporting an option, it does not have to provide anything associated with that option beyond the announcement that it does not support it.
2. An implementation can support a partial or incompatible version of an option (as a non-standard extension) as long as it does not claim to support the option.
3. An application can determine whether the option is supported. A strictly conforming application must check this announcement mechanism before first using anything associated with the option.

There is an important implication of this policy. XSH does not dictate the behavior of interfaces associated with an option when the implementation does not claim to support the option. In particular, it does not require that a function associated with an unsupported option will fail if it does not perform as specified.

The policy is enabled through the symbolic constants for Options and Option Groups defined in **<unistd.h>**. If a symbolic constant is undefined, the *fpathconf()*, *pathconf()*, or *sysconf()* functions can be used to determine whether the option is provided for a particular invocation of the application.

If a symbolic constant is defined with the value -1, the option is not supported. Headers, data types, and function interfaces required only for the option need not be supplied. An application that attempts to use anything associated only with the option is considered to be requiring an extension.

If a symbolic constant is defined with a value greater than zero, the option shall always be supported when the application is executed. All headers, data types, and functions shall be present and shall operate as specified.

If a symbolic constant is defined with the value zero, all headers, data types, and functions shall be present. The application can check at runtime to see whether the option is supported by calling *fpathconf()*, *pathconf()*, or *sysconf()* with the indicated name parameter.

Unless explicitly specified otherwise, the behavior of functions associated with an unsupported option is unspecified, and an application that uses such functions without first checking *fpathconf()*, *pathconf()*, or *sysconf()* is considered to be requiring an extension.

Any application must consider the following three cases for each option:

1. Option never supported.

The implementation advertises at compile time that the option will never be supported. In this case, it is not necessary for the implementation to supply any of the data types or function interfaces that are provided only as part of the option. The implementation might provide data types and functions that are similar to those defined by XSH, but there is no guarantee for any particular behavior.

2. Option always supported.

The implementation advertises at compile time that the option will always be supported. In this case, all data types and function interfaces shall be available and shall operate as specified.

3. Option might or might not be supported.

Some implementations might not provide a mechanism to specify support of options at compile time. In addition, the implementation might be unable or unwilling to specify support or non-support at compile time. In either case, any application that might use the option at runtime must be able to compile and execute. The implementation must provide, at compile time, all data types and function interfaces that are necessary to allow this. In this situation, there must be a mechanism that allows the application to query, at runtime, whether the option is supported. If the application attempts to use the option when it is not supported, the result is unspecified unless explicitly specified otherwise in XSH.

Shell and Utilities

XCU, Issue 6 describes all of the utilities required in the Single UNIX Specification environment. This chapter covers the contents of XCU, describing options, functional overview, the scope of the interfaces, an overview of the changes in this version, and lists of the new features. It also contains rationale of why certain utilities are excluded.

6.1 Options in XCU

Some of the utilities need not be present on a system conforming to the Single UNIX Specification. The optional utilities are as follows:

- DEVELOPMENT utilities are those required in a software development environment. The reference pages for these utilities are marked DEVELOPMENT.
- FORTRAN utilities are required in a FORTRAN-77 development environment, and essentially consist of the compiler, *fort77*. The reference page for this utility is marked FORTRAN.
- The Batch Environment Services and Utilities are optional. These are marked by the BE notation in their SYNOPSIS.

6.2 Functional Overview

The Single UNIX Specification supports a robust tool environment of 174 utilities, described in XCU. This is 160 external utilities and 14 required built-in utilities. The Single UNIX Specification utilities have been quite stable since XPG4 Base, and provide a standards-based tool environment based on UNIX systems practice.

The XSI extensions over the base POSIX functionality in XCU include utilities such as:

- The *sccs* suite of tools for source code management:
admin, delta, get, prs, rmdel, sact, sccs, unget, val, what
- The *uucp* suite of tools for UNIX-to-UNIX file copying:
uucp, uustat, uuto, uux
- The compression utilities:
compress, pack, pcat, uncompress, unpack, zcat
- Additional C-language development tools:
cflow, cxref, lint, m4
- Additional utilities:
fuser, ipcrm, ipcs, link, unlink

- Other miscellaneous tools and utilities. (These tools can be found in the table in Chapter 10 (on page 99), without a mark in the POSIX base column.)

The Single UNIX Specification utilities include:

<i>alias</i>	<i>diff</i>	<i>localedef</i>	<i>qhold</i>	<i>tr</i>
<i>ar</i>	<i>dirname</i>	<i>logger</i>	<i>qmove</i>	<i>true</i>
<i>asa</i>	<i>du</i>	<i>logname</i>	<i>qmsg</i>	<i>tsort</i>
<i>at</i>	<i>echo</i>	<i>lp</i>	<i>qrerun</i>	<i>tty</i>
<i>awk</i>	<i>ed</i>	<i>ls</i>	<i>qrls</i>	<i>type</i>
<i>basename</i>	<i>env</i>	<i>mailx</i>	<i>qselect</i>	<i>ulimit</i>
<i>batch</i>	<i>ex</i>	<i>man</i>	<i>qsig</i>	<i>umask</i>
<i>bc</i>	<i>expand</i>	<i>mesg</i>	<i>qstat</i>	<i>unalias</i>
<i>bg</i>	<i>expr</i>	<i>mkdir</i>	<i>qsub</i>	<i>uname</i>
<i>c99</i>	<i>false</i>	<i>mkfifo</i>	<i>read</i>	<i>uncompress</i>
<i>cal</i>	<i>fc</i>	<i>more</i>	<i>renice</i>	<i>unexpand</i>
<i>cat</i>	<i>fg</i>	<i>mv</i>	<i>rm</i>	<i>uniq</i>
<i>cd</i>	<i>file</i>	<i>newgrp</i>	<i>rmdir</i>	<i>uucp</i>
<i>chgrp</i>	<i>find</i>	<i>nice</i>	<i>sed</i>	<i>uudecode</i>
<i>chmod</i>	<i>fold</i>	<i>nl</i>	<i>sh</i>	<i>uuencode</i>
<i>chown</i>	<i>gencat</i>	<i>nohup</i>	<i>sleep</i>	<i>uupack</i>
<i>cksum</i>	<i>getconf</i>	<i>od</i>	<i>sort</i>	<i>uustat</i>
<i>cmp</i>	<i>getopts</i>	<i>paste</i>	<i>split</i>	<i>uux</i>
<i>comm</i>	<i>grep</i>	<i>patch</i>	<i>strings</i>	<i>vi</i>
<i>command</i>	<i>hash</i>	<i>pathchk</i>	<i>stty</i>	<i>wait</i>
<i>compress</i>	<i>head</i>	<i>pax</i>	<i>tabs</i>	<i>wc</i>
<i>cp</i>	<i>iconv</i>	<i>pcat</i>	<i>tail</i>	<i>who</i>
<i>crontab</i>	<i>id</i>	<i>pr</i>	<i>talk</i>	<i>write</i>
<i>csplit</i>	<i>jobs</i>	<i>printf</i>	<i>tee</i>	<i>xargs</i>
<i>cut</i>	<i>join</i>	<i>ps</i>	<i>test</i>	<i>zcat</i>
<i>date</i>	<i>kill</i>	<i>pwd</i>	<i>time</i>	
<i>dd</i>	<i>ln</i>	<i>qalter</i>	<i>touch</i>	
<i>df</i>	<i>locale</i>	<i>qdel</i>	<i>tput</i>	

There are two additional groups of utilities in XCU. The software development tools, marked with DEVELOPMENT in the NAME section of their reference page, consist of:

admin, cflow, ctags, cxref, delta, dis, get, lex, lint, m4, make, nm, prs, rmdel, sact, sccs, strip, unget, val, what, yacc

It should be noted that the compiler front-end, *c99*, is not considered part of the DEVELOPMENT group. All certified UNIX systems must provide a way of compiling C-language programs. The rest of the tools are development tools.

The only tool that is part of the FORTRAN development environment is the compiler itself, *fort77*, marked with FORTRAN in the NAME section of the reference page.

A certified UNIX system will provide all the tools in XCU, with the following provisions:

- The DEVELOPMENT utilities need not be provided.
- The FORTRAN compiler need not be provided.
- The Batch Environment Services and Utilities need not be provided.

If the implementation claims to provide the DEVELOPMENT option, then all the tools in the group must be provided.

6.3 Base Documents

The Shell and Utilities provided in XCU, Issue 6 are drawn from the following base documents:

- IEEE Std 1003.2-1992 (POSIX.2) (including IEEE Std 1003.2a-1992)
- The following amendments to ISO/IEC 9945-2: 1993 (POSIX-2):
 - The IEEE P1003.2b draft standard (Additional Utilities)
 - IEEE Std 1003.2d-1994 (Batch Environment)
- System Interface Definitions, Issue 5 (XBD5)
- Commands and Utilities, Issue 5 (XCU5)
- ISO/IEC 9899: 1999, Programming Languages — C (ISO C)

6.4 Overview of Changes

The following sections give a high-level overview of changes made to XCU since Issue 5. The CHANGE HISTORY section for each utility describes technical changes made to that utility from Issue 5. Changes between earlier issues of XCU and Issue 5 are not included.

The change history between Issue 5 and Issue 6 also lists the changes since ISO/IEC 9945-2: 1993 (POSIX-2).

Changes from Issue 5 to Issue 6

The following list summarizes the major changes that were made in XCU from Issue 5 to Issue 6:

- This volume is extensively revised so that it can be both an IEEE POSIX Standard and an Open Group Technical Standard.
- The terminology has been reworked to meet the style requirements.
- Shading notation and margin codes are introduced for identification of options within the volume.
- This volume is updated to mandate support of FIPS 151-2. The following changes were made:
 - Support is mandated for the capabilities associated with the following symbolic constants:
 - _POSIX_CHOWN_RESTRICTED
 - _POSIX_JOB_CONTROL
 - _POSIX_SAVED_IDS
 - In the environment for the login shell, the environment variables *LOGNAME* and *HOME* shall be defined and have the properties described in XBD, Chapter 7, Locale.
- A new section on Utility Limits is added.
- A section on the Relationships to Other Documents is added.
- Concepts and definitions have been moved to XBD.
- A RATIONALE section is added to each reference page.
- The *c89* utility (which specified a compiler for the C Language specified by the ISO/IEC 9899: 1990 standard) has been replaced by a *c99* utility (which specifies a compiler

for the C Language specified by the ISO/IEC 9899:1999 standard). The *c89* utility is withdrawn in this issue.

- IEEE Std 1003.2d-1994 is incorporated, adding the *qalter*, *qdel*, *qhold*, *qmove*, *qmsg*, *qrerun*, *qrls*, *qselect*, *qsig*, *qstat*, and *qsub* utilities.
- The IEEE P1003.2b draft standard is incorporated, making extensive updates and adding the *iconv* utility. The IEEE P1003.2b draft standard was withdrawn by the IEEE Standards Board upon the approval of the revision to IEEE Std 1003.1.
- Deficiencies have been corrected as a result of issues raised by IEEE PASC or ISO/IEC Interpretations against IEEE Std 1003.2-1992 (POSIX.2) (ISO/IEC 9945-2: 1993 (POSIX-2)).
- The Open Group's Corrigenda and Resolutions are applied.
- The following features, marked legacy or obsolescent in the base documents, are not carried forward in this issue. Other features from the base documents marked legacy or obsolescent are carried forward unless otherwise noted.

From XCU, Issue 5, the following legacy utilities are not carried forward:

calendar, *cancel*, *cc*, *col*, *cpio*, *cu*, *dircmp*, *dis*, *egrep*, *fgrep*, *line*, *lint*, *lpstat*, *mail*, *pack*, *pcat*, *pg*, *spell*, *sum*, *tar*, *unpack*, *uulog*, *uname*, *uupick*, *uuto*

In addition, legacy features within non-legacy reference pages (for example, headers) are not carried forward.

- From ISO/IEC 9945-2: 1993 (POSIX-2), obsolescent features within the following pages are not carried forward in this issue:

Page No.	Obsolescent Feature
75	Zero-length prefix within <i>PATH</i>
156, 159	<i>set</i>
178	<i>awk</i> ; use of no argument and no parentheses with length
259	<i>ed</i>
272	<i>env</i>
282	<i>find -perm[-]onum</i>
295-296	<i>egrep</i>
299-300	<i>head</i>
305-306	<i>join</i>
309-310	<i>kill</i>
431-433, 435-436	<i>sort</i>
444-445	<i>tail</i>
453, 455-456	<i>touch</i>
464-465	<i>tty</i>
472	<i>uniq</i>
515-516	<i>ex</i>
542-543	<i>expand</i>
563-565	<i>more</i>
574-576	<i>newgrp</i>
578	<i>nice</i>
594-596	<i>renice</i>

Page No.	Obsolescent Feature
597-598	<i>split</i>
600-601	<i>strings</i>
624-625	<i>vi</i>
693	<i>lex</i>

New Features in Issue 6

The following table lists the new utilities introduced since ISO/IEC 9945-2: 1993 (POSIX-2) (as modified by IEEE Std 1003.2d-1994). Apart from the *c99* and *iconv* utilities, these are all part of the XSI extension.

New Utilities in Issue 6							
<i>admin</i>	<i>compress</i>	<i>gencat</i>	<i>ipcrm</i>	<i>nl</i>	<i>tsort</i>	<i>unlink</i>	<i>val</i>
<i>c99</i>	<i>cxref</i>	<i>get</i>	<i>ipcs</i>	<i>pr</i>	<i>ulimit</i>	<i>uucp</i>	<i>what</i>
<i>cal</i>	<i>delta</i>	<i>hash</i>	<i>link</i>	<i>sact</i>	<i>uncompress</i>	<i>uustat</i>	<i>zcat</i>
<i>cflow</i>	<i>fuser</i>	<i>iconv</i>	<i>m4</i>	<i>sccs</i>	<i>unget</i>	<i>uux</i>	

6.5 Exclusion of Utilities

The set of utilities contained in the Single UNIX Specification is drawn from the base documents, with one addition: the *c99* utility. This section contains information on some of the deliberations that led to this set of utilities, and why certain utilities were excluded.

Many utilities were evaluated by the standard developers; more historical utilities were excluded from the base documents than included. The following list contains many common UNIX system utilities that were not included.

This section is limited to a discussion of only those utilities actively or indirectly evaluated by the standard developers of the base documents, rather than the list of all known UNIX utilities from all its variants.

- adb* The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool. Furthermore, many useful aspects of *adb* are very hardware-specific.
- as* Assemblers are hardware-specific and are included implicitly as part of the compilers in the Single UNIX Specification.
- banner* The only known use of this command is as part of the *lp* printer header pages. It was decided that the format of the header is implementation-defined, so this utility is superfluous to applications portability.
- calendar* This reminder service program is not useful to conforming applications.
- cancel* The *lp* (line printer spooling) system specified is the most basic possible and did not need this level of application control.
- chroot* This is primarily of administrative use, requiring superuser privileges.

<i>col</i>	No utilities defined in the Single UNIX Specification produce output requiring such a filter. The <i>nroff</i> text formatter is present on many historical systems and will continue to remain as an extension; <i>col</i> is expected to be shipped by all the systems that ship <i>nroff</i> .
<i>cpio</i>	This has been replaced by <i>pax</i> , for reasons explained in the rationale for that utility.
<i>cpp</i>	This is subsumed by <i>c99</i> .
<i>cu</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs.
<i>dc</i>	The functionality of this utility can be provided by the <i>bc</i> utility; <i>bc</i> was selected because it was easier to use and had superior functionality. Although the historical versions of <i>bc</i> are implemented using <i>dc</i> as a base, the Single UNIX Specification prescribes the interface and not the underlying mechanism used to implement it.
<i>dircmp</i>	Although a useful concept, the historical output of this directory comparison program is not suitable for processing in application programs. Also, the <i>diff -r</i> command gives equivalent functionality.
<i>dis</i>	Disassemblers are hardware-specific.
<i>emacs</i>	The community of <i>emacs</i> editing enthusiasts was adamant that the full <i>emacs</i> editor not be included in the base documents because they were concerned that an attempt to standardize this very powerful environment would encourage vendors to ship versions conforming strictly to the standard, but lacking the extensibility required by the community. The author of the original <i>emacs</i> program also expressed his desire to omit the program. Furthermore, there were a number of historical UNIX systems that did not include <i>emacs</i> , or included it without supporting it, but there were very few that did not include and support <i>vi</i> .
<i>ld</i>	This is subsumed by <i>c99</i> .
<i>line</i>	The functionality of <i>line</i> can be provided with <i>read</i> .
<i>lint</i>	<p>This technology is partially subsumed by <i>c99</i>. It is also hard to specify the degree of checking for possible error conditions in programs in any compiler, and specifying what <i>lint</i> would do in these cases is equally difficult.</p> <p>It is fairly easy to specify what a compiler does. It requires specifying the language, what it does with that language, and stating that the interpretation of any incorrect program is unspecified. Unfortunately, any description of <i>lint</i> is required to specify what to do with erroneous programs. Since the number of possible errors and questionable programming practices is infinite, one cannot require <i>lint</i> to detect all errors of any given class.</p> <p>Additionally, some vendors complained that since many compilers are distributed in a binary form without a <i>lint</i> facility (because the ISO C standard does not require one), implementing the standard as a stand-alone product will be much harder. Rather than being able to build upon a standard compiler component (simply by providing <i>c99</i> as an interface), source to that compiler would most likely need to be modified to provide the <i>lint</i> functionality. This was considered a major burden on system providers for a very small gain to developers (users).</p>
<i>login</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs.

<i>lorder</i>	This utility is an aid in creating an implementation-defined detail of object libraries that the standard developers did not feel required standardization.
<i>lpstat</i>	The <i>lp</i> system specified is the most basic possible and did not need this level of application control.
<i>mail</i>	This utility was omitted in favor of <i>mailx</i> because there was a considerable functionality overlap between the two.
<i>mknod</i>	This was omitted in favor of <i>mkfifo</i> , as <i>mknod</i> has too many implementation-defined functions.
<i>news</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs.
<i>pack</i>	This compression program was considered inferior to <i>compress</i> .
<i>passwd</i>	This utility was proposed in a historical draft of the base documents but met with too many objections to be included. There were various reasons: <ul style="list-style-type: none"> • Changing a password should not be viewed as a command, but as part of the login sequence. Changing a password should only be done while a trusted path is in effect. • Even though the text in early drafts was intended to allow a variety of implementations to conform, the security policy for one site may differ from another site running with identical hardware and software. One site might use password authentication while the other did not. Vendors could not supply a <i>passwd</i> utility that would conform to the Single UNIX Specification for all sites using their system. • This is really a subject for a system administration working group or a security working group.
<i>pcat</i>	This compression program was considered inferior to <i>zcat</i> .
<i>pg</i>	This duplicated many of the features of the <i>more</i> pager, which was preferred by the standard developers.
<i>prof</i>	The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool.
RCS	RCS was originally considered as part of a version control utilities portion of the scope. However, this aspect was abandoned by the standard developers. SCCS is included as part of the Development option within the Single UNIX Specification.
<i>red</i>	Restricted editor. This was not considered by the standard developers because it never provided the level of security restriction required.
<i>rsh</i>	Restricted shell. This was not considered by the standard developers because it does not provide the level of security restriction that is implied by historical documentation.
<i>sdb</i>	The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool. Furthermore, some useful aspects of <i>sdb</i> are very hardware-specific.

<i>sdiff</i>	The “side-by-side <i>diff</i> ” utility from System V was omitted because it is used infrequently, and even less so by conforming applications. Despite being in System V, it is not in the SVID or XPG.
<i>shar</i>	Any of the numerous “shell archivers” were excluded because they did not meet the requirement of existing practice.
<i>shl</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs. The job control aspects of the shell command language are generally more useful.
<i>size</i>	The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool.
<i>spell</i>	This utility is not useful from shell scripts or typical application programs. The <i>spell</i> utility was considered, but was omitted because there is no known technology that can be used to make it recognize general language for user-specified input without providing a complete dictionary along with the input file.
<i>su</i>	This utility is not useful from shell scripts or typical application programs. (There was also sentiment to avoid security-related utilities.)
<i>sum</i>	This utility was renamed <i>cksum</i> .
<i>tar</i>	This utility has been replaced by <i>pax</i> , for reasons explained in the rationale for that utility.
<i>unpack</i>	This compression program was considered inferior to <i>uncompress</i> .
<i>wall</i>	This utility is terminal-oriented and is not useful in shell scripts or typical applications. It is generally used only by system administrators.

Headers

The headers in this version of the Single UNIX Specification are now included in XBD, Issue 6, Chapter 13. This is a change over previous versions of the Single UNIX Specification where they had been included in XSH. This chapter covers the header and name space rules, the scope of the headers, an overview of the changes to them in this version of the specification, and lists of the new headers.

7.1 Header and Name Space Rules

The headers in the Single UNIX Specification are built upon the headers from the ISO C standard.

7.1.1 ISO C Headers

The ISO C standard describes general rules for implementations *versus* applications with respect to headers and name space.

Headers

1. Headers need not be regular text files and the characters between the '`<`' and '`>`' need not name a source file.
2. Headers should be self-sufficient such that any standard header does not need any other standard header to be included using **#include** before or after it (and thus headers can be included in any order).
3. Headers should be idempotent, such that any standard header can be included any number of times without causing problems.
4. Headers can provide macro versions of functions which should behave in a semantically identical fashion. By undefining such a macro, a regular declaration of the function is made visible.
5. Headers can assume that they are included outside of any file scope declaration or definition.
6. Headers can assume that they are included prior to the first use of its functions, objects, or macros.

With the exception of **<assert.h>**, the ISO C standard headers are both self-sufficient and idempotent.

The ISO C standard specifies the rules for its 24 standard headers. Items 2. through 6. listed above for headers are requirements for these.

<assert.h>	<float.h>	<math.h>	<stddef.h>	<time.h>
<complex.h>	<inttypes.h>	<setjmp.h>	<stdio.h>	<tgmath.h>
<ctype.h>	<iso646.h>	<signal.h>	<stdint.h>	<wchar.h>
<errno.h>	<limits.h>	<stdarg.h>	<stdlib.h>	<wctype.h>
<fenv.h>	<locale.h>	<stdbool.h>	<string.h>	

A compilation system can provide more headers, but a strictly conforming ISO C standard program can only use these.

Name Space

1. `[_A-Z][0-9a-z_A-Z]*` are reserved for implementation use anywhere.
2. `[_0-9a-z_A-Z]*` are reserved for implementation use as identifiers with external linkage and in any header as a tag or an ordinary identifier with file scope.

7.1.2 POSIX.1-2001 Base Headers

The requirements for the POSIX.1 base standard are a subset of the requirements for the Single UNIX Specification. For the purposes of this description, the POSIX.1 base standard is the set of requirements excluding the XSI extension.

The POSIX.1 base standard requires that the macro `_POSIX_C_SOURCE` be defined using **#define** before a standard header is included. The POSIX-specific names are then declared and no others are permitted by POSIX.

POSIX.1 specifies general rules for name space control through *feature test macros*.

1. Feature test macros have names that match `[_A-Z][0-9a-z_A-Z]*`. (Thus, the use of such names takes advantage of the loophole left in the ISO C standard.) They almost always match `[_A-Z]_SOURCE`. The original feature test macro for the 1990 revision of POSIX.1 is `_POSIX_SOURCE`. The 2001 revision of POSIX.1 use the more powerful feature test macro called `_POSIX_C_SOURCE`, which allows you to specify the revision of POSIX to which your application conforms. The value of `_POSIX_C_SOURCE` which corresponds to the functionality contained in the Single UNIX Specification, Version 3 is 200112L.
2. If the feature test macro associated with a particular header is defined prior to its inclusion, the header must obey the name space rules of the matching specification.
3. If such a header is included without the feature test macro being defined, it has no restrictions (at least with respect to the associated specification).

POSIX.1 specifies general rules for its headers:

1. `[a-z_A-Z][0-9a-z_A-Z]*_t` are reserved for all POSIX.1 headers. It is unclear whether this reservation is intended to be in effect for those headers that are covered by the ISO C standard. This makes the assumption that it does apply.
2. Each function must be declared with a prototype in at least one header (ISO C standard implementations).
3. The default location for this declaration is `<unistd.h>`.

POSIX.1 specifies the rules for the 27 standard POSIX base headers and subsumes the headers from the ISO C standard, giving additional rules to six of them.

<aioh.h>	<net/if.h>	<sys/mman.h>	<tar.h>
<dirent.h>	<pthread.h>	<sys/stat.h>	<termios.h>
<fcntl.h>	<regex.h>	<sys/times.h>	<trace.h>
<fnmatch.h>	<pwd.h>	<sys/types.h>	<unistd.h>
<glob.h>	<sched.h>	<sys/un.h>	<utime.h>
<grp.h>	<semaphore.h>	<sys/utsname.h>	<wordexp.h>
<queue.h>	<spawn.h>	<sys/wait.h>	

The six modified ISO C standard headers are:

```
<errno.h>
<limits.h>
<setjmp.h>
<signal.h>
<stdio.h>
<time.h>
```

7.1.3 XSI Headers

The Single UNIX Specification mandates the XSI extension within the joint revision and subsumes the ISO C standard and POSIX.1 base, adding the following headers:

<cpio.h>	<monetary.h>	<sys/ipc.h>	<sys/timeb.h>
<dlfcn.h>	<ndbm.h>	<sys/msg.h>	<sys/uio.h>
<fmtmsg.h>	<nl_types.h>	<sys/resource.h>	<syslog.h>
<ftw.h>	<poll.h>	<sys/sem.h>	<ucontext.h>
<iconv.h>	<search.h>	<sys/shm.h>	<ulimit.h>
<langinfo.h>	<strings.h>	<sys/statvfs.h>	<utmpx.h>
<libgen.h>	<stropts.h>	<sys/time.h>	

A feature test macro named `_XOPEN_SOURCE` provides for all the XSI extensions (which include the POSIX.1 base) to the ISO C standard, in a similar way to the `_POSIX_C_SOURCE` macro. For conformance to the Single UNIX Specification, Version 3, `_XOPEN_SOURCE` should be set to the value 600. The definition of `_XOPEN_SOURCE` subsumes the use of `_POSIX_C_SOURCE`, ensuring that the appropriate POSIX name space is also exposed.

7.2 Names Safe to Use

Reserved name rules are complicated and set out below in the following sections. There are, however, four fairly simple rules which, if followed, avoid collisions with any reserved names:

1. Use **#include** to include all system headers at the top of source files.
2. Do not define or declare any names that begin with an underscore.
3. Use an underscore or a capital letter somewhere within the first few characters of all file scope tags and regular names.

Note: Beware of the `va_` prefix found in `<stdarg.h>`.

4. Use a digit or a non-capital letter somewhere within the first few characters of all macro names.

Note: Almost all names beginning with an 'E' are reserved if `<errno.h>` is included.

Most implementations continue to add names to the standard headers. The `_POSIX_C_SOURCE` and `_XOPEN_SOURCE` macros can be used to remove the added

declarations, so that a predictable standard name space is visible.

7.2.1 Reserved Names

Consult XSH, Section 2.2 for full information on the name space reservation. The section gives tables of the symbols reserved for implementations and future standards

7.3 Base Documents

The headers provided in XBD, Issue 6 are drawn from the following base documents:

- IEEE Std 1003.1-1996 (POSIX.1) (incorporating IEEE Std 1003.1-1990 (POSIX.1), IEEE Std 1003.1b-1993, IEEE Std 1003.1c-1995, and IEEE Std 1003.1i-1995)
- The following amendments to IEEE Std 1003.1-1990 (POSIX.1):
 - The IEEE P1003.1a draft standard (Additional System Services)
 - IEEE Std 1003.1d-1999 (Additional Realtime Extensions)
 - IEEE Std 1003.1g-2000 (Protocol-Independent Interfaces (PII))
 - IEEE Std 1003.1j-2000 (Advanced Realtime Extensions)
 - IEEE Std 1003.1q-2000 (Tracing)
- System Interfaces and Headers, Issue 5 (XSH5) (in 2 Volumes)
- Networking Services, Issue 5.2 (XNS5.2)
- ISO/IEC 9899: 1999, Programming Languages — C (ISO C)

7.4 Overview of Changes

The following sections describe changes made to the headers since Issue 5. The CHANGE HISTORY section for each entry details the technical changes that have been made to that entry from Issue 5. Changes between earlier issues and Issue 5 are not included.

The change history between Issue 5 and Issue 6 also lists the changes since ISO/IEC 9945-1: 1996 (POSIX-1).

Changes from Issue 5 to Issue 6

The following list summarizes the major changes that were made in the headers from Issue 5 to Issue 6:

- The headers are now included in XBD; previously in Issue 5 they were in XSH.
- From XSH, Issue 5, the following legacy headers are not carried forward:
 - `<re_comp.h>`, `<regexp.h>`, `<varargs.h>`
- A RATIONALE section is added to each reference page.
- Networking interfaces from the XNS, Issue 5.2 specification are incorporated.
- IEEE Std 1003.1d-1999 is incorporated.
- IEEE Std 1003.1j-2000 is incorporated.

- IEEE Std 1003.1q-2000 is incorporated.
- The IEEE P1003.1a draft standard is incorporated.
- Existing functionality is aligned with the ISO/IEC 9899:1999 standard.
- New functionality from the ISO/IEC 9899:1999 standard is incorporated.
- IEEE PASC Interpretations are applied. The final text for IEEE Interpretations can be located by following the links from www.pasc.org/interps.
- The Open Group Corrigenda and Resolutions are applied.

New Features in Issue 6

The following new headers are introduced in Issue 6:

New Headers in Issue 6		
<complex.h>	<spawn.h>	<tgmath.h>
<fenv.h>	<stdbool.h>	<trace.h>
<net/if.h>	<stdint.h>	

The following table lists the headers from the XSI extension. These are new since ISO/IEC 9945-1:1996 (POSIX-1).

New XSI Headers in Issue 6		
<cpio.h>	<poll.h>	<sys/statvfs.h>
<dlfcn.h>	<search.h>	<sys/time.h>
<fmtmsg.h>	<strings.h>	<sys/timeb.h>
<ftw.h>	<stropts.h>	<sys/uio.h>
<iconv.h>	<sys/ipc.h>	<syslog.h>
<langinfo.h>	<sys/mman.h>	<ucontext.h>
<libgen.h>	<sys/msg.h>	<ulimit.h>
<monetary.h>	<sys/resource.h>	<utmpx.h>
<ndbm.h>	<sys/sem.h>	
<nl_types.h>	<sys/shm.h>	

Terminal Interfaces

The Single UNIX Specification contains X/Open Curses, Issue 4, Version 2. These interfaces support a terminal-independent character screen update method. X/Open Curses, Issue 4, Version 2 is an upwardly-compatible enhancement of X/Open Curses, Issue 3, adding considerable functionality supporting multi-byte and wide characters, color terminal devices, support for wide-spacing and non-spacing characters, color support, and modem and printer control via *terminfo*.

New interfaces over Issue 3 are marked with ENHANCED CURSES in their reference page headers, and a portability code (EC) (similar to those found in XSH) is used with shading to indicate the extended features throughout the specification.

A number of interfaces have been marked TO BE WITHDRAWN, as comments from the industry have clarified certain interfaces in the earlier X/Open Curses, Issue 3 model. These interfaces are all still available on certified UNIX systems, supporting existing applications portability, and the APPLICATION USAGE section of the appropriate reference pages gives guidance to developers for how best to plan for future portability needs.

8.1 Functional Overview

X/Open Curses, Issue 4, Version 2, Chapters 1 to 3, provide all the introductory material needed to work with the X/Open Curses interfaces. These interfaces include:

<i>add_wch()</i>	<i>delwin()</i>	<i>initscr()</i>	<i>pnoutrefresh()</i>	<i>standend()</i>
<i>add_wchnstr()</i>	<i>derwin()</i>	<i>innstr()</i>	<i>printw()</i>	<i>start_color()</i>
<i>addch()</i>	<i>douupdate()</i>	<i>innwstr()</i>	<i>putp()</i>	<i>stdscr()</i>
<i>addchstr()</i>	<i>dupwin()</i>	<i>ins_nwstr()</i>	<i>putwin()</i>	<i>subpad()</i>
<i>addnstr()</i>	<i>echo()</i>	<i>ins_wch()</i>	<i>qiflush()</i>	<i>subwin()</i>
<i>addnwstr()</i>	<i>echo_wchar()</i>	<i>ins_wstr()</i>	<i>raw()</i>	<i>syncok()</i>
<i>attr_get()</i>	<i>echochar()</i>	<i>insch()</i>	<i>redrawwin()</i>	<i>term_attrs()</i>
<i>attr_get()</i>	<i>endwin()</i>	<i>insdelln()</i>	<i>refresh()</i>	<i>termattrs()</i>
<i>attr_off()</i>	<i>erase()</i>	<i>insertln()</i>	<i>reset_prog_mode()</i>	<i>termname()</i>
<i>attr_on()</i>	<i>erasechar()</i>	<i>insnstr()</i>	<i>resetty()</i>	<i>tgetent()</i>
<i>attr_set()</i>	<i>filter()</i>	<i>insstr()</i>	<i>restartterm()</i>	<i>tigetflag()</i>
<i>attroff()</i>	<i>flash()</i>	<i>instr()</i>	<i>ripline()</i>	<i>timeout()</i>
<i>baudrate()</i>	<i>flushinp()</i>	<i>intrflush()</i>	<i>savetty()</i>	<i>touchline()</i>
<i>beep()</i>	<i>get_wch()</i>	<i>inwstr()</i>	<i>scanw()</i>	<i>tparm()</i>
<i>bkgd()</i>	<i>get_wstr()</i>	<i>is_linetouched()</i>	<i>scr_dump()</i>	<i>tputs()</i>
<i>bkgnd()</i>	<i>getbegyx()</i>	<i>isendwin()</i>	<i>sclr()</i>	<i>typeahead()</i>
<i>border()</i>	<i>getbkgd()</i>	<i>keyname()</i>	<i>scrollok()</i>	<i>unctrl()</i>
<i>border_set()</i>	<i>getbkgnd()</i>	<i>keypad()</i>	<i>set_curterm()</i>	<i>ungetch()</i>
<i>box()</i>	<i>getcchar()</i>	<i>killchar()</i>	<i>set_term()</i>	<i>untouchwin()</i>

<i>box_set()</i>	<i>getch()</i>	<i>leaveok()</i>	<i>setcchar()</i>	<i>use_env()</i>
<i>can_change_color()</i>	<i>getmaxyx()</i>	<i>longname()</i>	<i>setscreg()</i>	<i>vid_attr()</i>
<i>cbreak()</i>	<i>getn_wstr()</i>	<i>meta()</i>	<i>setupterm()</i>	<i>vid_puts()</i>
<i>chgat()</i>	<i>getnstr()</i>	<i>move()</i>	<i>slk_attr_off()</i>	<i>vidattr()</i>
<i>clear()</i>	<i>getparyx()</i>	<i>mvcur()</i>	<i>slk_attr_on()</i>	<i>vline()</i>
<i>clearok()</i>	<i>getstr()</i>	<i>mvderwin()</i>	<i>slk_attr_set()</i>	<i>vline_set()</i>
<i>clrtoeol()</i>	<i>getwin()</i>	<i>mvprintw()</i>	<i>slk_atroff()</i>	<i>vw_printw()</i>
<i>clrtoeol()</i>	<i>getyx()</i>	<i>mvscanw()</i>	<i>slk_atroff()</i>	<i>vw_scanw()</i>
<i>color_content()</i>	<i>halfdelay()</i>	<i>mvwin()</i>	<i>slk_attron()</i>	<i>vwprintw()</i>
<i>color_set()</i>	<i>has_colors()</i>	<i>napms()</i>	<i>slk_attrset()</i>	<i>vwscanw()</i>
<i>copywin()</i>	<i>has_ic()</i>	<i>newpad()</i>	<i>slk_clear()</i>	<i>wattr_get()</i>
<i>cur_term()</i>	<i>hline()</i>	<i>newterm()</i>	<i>slk_color()</i>	<i>wattr_off()</i>
<i>cursor_set()</i>	<i>hline_set()</i>	<i>newwin()</i>	<i>slk_init()</i>	<i>wattr_on()</i>
<i>curscr()</i>	<i>idcok()</i>	<i>nl()</i>	<i>slk_label()</i>	<i>wattr_set()</i>
<i>def_prog_mode()</i>	<i>immedok()</i>	<i>nodelay()</i>	<i>slk_nooutrefresh()</i>	<i>wcolor_set()</i>
<i>del_curterm()</i>	<i>in_wch()</i>	<i>noqiflush()</i>	<i>slk_refresh()</i>	<i>wunctrl()</i>
<i>delay_output()</i>	<i>in_wchnstr()</i>	<i>notimeout()</i>	<i>slk_restore()</i>	
<i>delch()</i>	<i>inch()</i>	<i>overlay()</i>	<i>slk_set()</i>	
<i>deleteln()</i>	<i>inchstr()</i>	<i>pair_content()</i>	<i>slk_touch()</i>	
<i>delscreen()</i>	<i>init_color()</i>	<i>pechochar()</i>	<i>slk_wset()</i>	

Note that there are functions listed in the table that represent a set of related functions. For example, *addch()* represents the set of functions *addch()*, *mvaddch()*, *mvwaddch()*, and *waddch()*. A table in X/Open Curses, Issue 4, Version 2, Section 3.6.2, Function Families Provided, summarizes these families of related calls.

8.2 Curses in an Issue 6 Environment

Corrigendum U056 (dated November 2001) describes the changes required to X/Open Curses, Issue 4, Version 2 to function in a Base Specifications, Issue 6 environment.

This includes the following changes.

Change Number: U056/1

On environments supporting System Interfaces, Issue 6, replace Section 2.2, The Compilation Environment, with the following text:

Applications should ensure that the feature test macro `_XOPEN_SOURCE` is defined before inclusion of any header. This is needed to enable the functionality described in this document, and possibly to enable functionality defined elsewhere in the Common Applications Environment.

The `_XOPEN_SOURCE` macro may be defined automatically by the compilation process, but to ensure maximum portability, applications should make sure that `_XOPEN_SOURCE` is defined by using either compiler options or `#define` directives in the source files, before any `#include` directives. Identifiers in this document may only be undefined using the `#undef` directive as described in Chapter 2 or Section 2.2.1. These `#undef` directives must follow all `#include` directives of any XSI headers.

Most strictly conforming POSIX and ISO C applications will compile on systems conformant with this specification. However, an application

which uses any of the items marked as an extension to POSIX and ISO C, for any purpose other than that shown here, may not compile. In such cases, it may be necessary to alter those applications to use alternative identifiers.

Since this document is aligned with the ISO C standard, and since all functionality enabled by `_POSIX_C_SOURCE` set equal to 200112L is enabled by `_XOPEN_SOURCE` set equal to 600, there should be no need to define `_POSIX_C_SOURCE` if `_XOPEN_SOURCE` is so defined. Therefore, if `_XOPEN_SOURCE` is set equal to 600 and `_POSIX_C_SOURCE` is set equal to 200112L, the behavior is the same as if only `_XOPEN_SOURCE` is defined and set equal to 600. However, should `_POSIX_C_SOURCE` be set to a value greater than 200112L, the behavior is unspecified.

The `c99` utility shall recognize the additional `-l` operand for standard libraries:

`-l curses` This operand makes visible all library functions referenced in this specification (except for those labelled ENHANCED CURSES and except for portions marked with the EC margin legend).

EC If the implementation defines `_XOPEN_CURSES` and if the application defines the `_XOPEN_SOURCE` feature test macro with the value 600 before including any header, then `-l curses` also makes visible all library functions referenced in this specification and labelled ENHANCED CURSES and portions marked with the EC margin legend.

It is unspecified whether the library `libcurses.a` exists as a regular file.

EC An application that uses any API specified as ENHANCED CURSES or relies on any portion of this specification marked with the EC margin legend must define `_XOPEN_SOURCE=600` in each source file or as part of its compilation environment.

Base Environment: System Interfaces, Issue 6

Change Number: U056/2

On environments supporting System Interfaces, Issue 6, change the following text within the reference page:

These attribute flags need not be distinct

EC except when `_XOPEN_CURSES` is defined and the application sets `_XOPEN_SOURCE_EXTENDED` to 1.

to:

These attribute flags need not be distinct

EC except when `_XOPEN_CURSES` is defined and the application sets `_XOPEN_SOURCE=600`.

Base Environment: System Interfaces, Issue 6

Change Number: U056/3

On environments supporting System Interfaces, Issue 6, change the following text within the reference page:

The following data types are defined through typedef:

bool Boolean data type.

to:

The following data types are defined through typedef:

bool As described in <stdbool.h>.

Base Environment: System Interfaces, Issue 6

System Interface Table

This chapter contains a table of all the interfaces defined in XSH, Issue 6, complete with an indication of their status for the XSI extension, the POSIX Base, and their availability in UNIX 98 (denoted by U98), UNIX 95 (denoted by U95), IEEE Std 1003.1-1996 (POSIX.1) (denoted P96), IEEE Std 1003.2-1992 (POSIX.2) (denoted P92), the ISO C standard (denoted C99), C 89, SVID, Issue 3 (denoted by SVID3) and BSD 4.3 (denoted BSD). The following conventions are used in columns 2 through 10:

- m** Indicates that the interface is defined as mandatory.
- o** Indicates that the interface is part of an Option or Feature Group.
- ob** Indicates that the interface is Obsolescent, and although mandatory for the implementation, applications are discouraged from its use.
- l** In the XSI column, indicates that the interface is part of the Legacy Option Group and need not be available on all implementations.
- opt* In the POSIX Base column, two or three letter option codes are used as described in Section 3.3 (on page 16), denoting the option to which the interface belongs.
- r** In the P96 column, this indicates that the interface is part of the POSIX Realtime Extension.
- t** In the P96 column, this indicates that the interface is part of the POSIX Threads Extension.
- 1** In the C89 column this indicates that the interface is part of the ISO/IEC 9899:1990 standard.
- .** Indicates that the interface is not specified.

The table is intended as a quick reference guide for programmers migrating to or developing applications for the Single UNIX Specification, Version 3. Products that brand to a profile may not provide all of the interfaces listed, depending on which Option or Feature Groups are supported.

There are 1123 interfaces listed.

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>FD_CLR()</i>	m	m	m	m
<i>FD_ISSET()</i>	m	m	m	m
<i>FD_SET()</i>	m	m	m	m
<i>FD_ZERO()</i>	m	m	m	m
<i>_Exit()</i>	m	m	.	.	.	m	.	.	.
<i>_exit()</i>	m	m	m	m	m	.	.	m	m
<i>_longjmp()</i>	m	xsi	m	m	m
<i>_setjmp()</i>	m	xsi	m	m	m
<i>_tolower()</i>	m	xsi	m	m	.	.	.	m	.
<i>_toupper()</i>	m	xsi	m	m	.	.	.	m	.
<i>a64l()</i>	m	xsi	m	m	.	.	.	m	.
<i>abort()</i>	m	m	m	m	m	m	m	m	m
<i>abs()</i>	m	m	m	m	m	m	m	m	m
<i>accept()</i>	m	m	m	m	m
<i>access()</i>	m	m	m	m	m	.	.	m	m
<i>acos()</i>	m	m	m	m	m	m	m	m	m
<i>acosf()</i>	m	m	.	.	.	m	.	.	.
<i>acosh()</i>	m	m	m	m	.	m	.	m	m
<i>acoshf()</i>	m	m	.	.	.	m	.	.	.
<i>acoshl()</i>	m	m	.	.	.	m	.	.	.
<i>acosl()</i>	m	m	.	.	.	m	.	.	.
<i>aio_cancel()</i>	o	aio	o	.	r
<i>aio_error()</i>	o	aio	o	.	r
<i>aio_fsync()</i>	o	aio	o	.	r
<i>aio_read()</i>	o	aio	o	.	r
<i>aio_return()</i>	o	aio	o	.	r
<i>aio_suspend()</i>	o	aio	o	.	r
<i>aio_write()</i>	o	aio	o	.	r
<i>alarm()</i>	m	m	m	m	m	.	.	m	m
<i>asctime()</i>	m	m	m	m	m	m	m	m	m
<i>asctime_r()</i>	m	tsf	m	.	t
<i>asin()</i>	m	m	m	m	m	m	m	m	m
<i>asinf()</i>	m	m	.	.	.	m	.	.	.
<i>asinh()</i>	m	m	m	m	.	m	.	m	m
<i>asinhf()</i>	m	m	.	.	.	m	.	.	.
<i>asinhf()</i>	m	m	.	.	.	m	.	.	.
<i>asinl()</i>	m	m	.	.	.	m	.	.	.
<i>assert()</i>	m	m	m	m	m	m	m	m	m
<i>atan()</i>	m	m	m	m	m	m	m	m	m
<i>atan2()</i>	m	m	m	m	m	m	m	m	m
<i>atan2f()</i>	m	m	.	.	.	m	.	.	.
<i>atan2l()</i>	m	m	.	.	.	m	.	.	.
<i>atanf()</i>	m	m	.	.	.	m	.	.	.
<i>atanh()</i>	m	m	m	m	.	m	.	m	m
<i>atanhf()</i>	m	m	.	.	.	m	.	.	.
<i>atanhl()</i>	m	m	.	.	.	m	.	.	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>atanl()</i>	m	m	.	.	.	m	.	.	.
<i>atexit()</i>	m	m	m	m	.	m	m	m	.
<i>atof()</i>	m	m	m	m	m	m	m	m	m
<i>atoi()</i>	m	m	m	m	m	m	m	m	m
<i>atol()</i>	m	m	m	m	m	m	m	m	m
<i>atoll()</i>	m	m	.	.	.	m	.	.	.
<i>basename()</i>	m	xsi	m	m
<i>bcmp()</i>	l	xsi	m	m	m
<i>bcopy()</i>	l	xsi	m	m	m
<i>bind()</i>	m	m	m	m	m
<i>bsd_signal()</i>	ob	xsi	m	m
<i>bsearch()</i>	m	m	m	m	m	m	m	m	.
<i>btowc()</i>	m	m	m	.	.	m	1	.	.
<i>bzero()</i>	l	xsi	m	m	m
<i>cabs()</i>	m	m	.	.	.	m	.	.	.
<i>cabsf()</i>	m	m	.	.	.	m	.	.	.
<i>cabsl()</i>	m	m	.	.	.	m	.	.	.
<i>cacos()</i>	m	m	.	.	.	m	.	.	.
<i>cacosf()</i>	m	m	.	.	.	m	.	.	.
<i>cacosh()</i>	m	m	.	.	.	m	.	.	.
<i>cacoshf()</i>	m	m	.	.	.	m	.	.	.
<i>cacoshl()</i>	m	m	.	.	.	m	.	.	.
<i>cacosl()</i>	m	m	.	.	.	m	.	.	.
<i>calloc()</i>	m	m	m	m	m	m	m	m	m
<i>carg()</i>	m	m	.	.	.	m	.	.	.
<i>cargf()</i>	m	m	.	.	.	m	.	.	.
<i>cargl()</i>	m	m	.	.	.	m	.	.	.
<i>casin()</i>	m	m	.	.	.	m	.	.	.
<i>casinf()</i>	m	m	.	.	.	m	.	.	.
<i>casinh()</i>	m	m	.	.	.	m	.	.	.
<i>casinhf()</i>	m	m	.	.	.	m	.	.	.
<i>casinhl()</i>	m	m	.	.	.	m	.	.	.
<i>casinl()</i>	m	m	.	.	.	m	.	.	.
<i>catan()</i>	m	m	.	.	.	m	.	.	.
<i>catanf()</i>	m	m	.	.	.	m	.	.	.
<i>catanh()</i>	m	m	.	.	.	m	.	.	.
<i>catanhf()</i>	m	m	.	.	.	m	.	.	.
<i>catanhl()</i>	m	m	.	.	.	m	.	.	.
<i>catanl()</i>	m	m	.	.	.	m	.	.	.
<i>catclose()</i>	m	xsi	m	m	.	.	.	m	.
<i>catgets()</i>	m	xsi	m	m	.	.	.	m	.
<i>catopen()</i>	m	xsi	m	m	.	.	.	m	.
<i>cbrt()</i>	m	m	m	m	.	m	.	m	m
<i>cbrtf()</i>	m	m	.	.	.	m	.	.	.
<i>cbrtl()</i>	m	m	.	.	.	m	.	.	.
<i>ccos()</i>	m	m	.	.	.	m	.	.	.

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>ccosf()</i>	m	m	.	.	.	m	.	.	.
<i>ccosh()</i>	m	m	.	.	.	m	.	.	.
<i>ccoshf()</i>	m	m	.	.	.	m	.	.	.
<i>ccoshl()</i>	m	m	.	.	.	m	.	.	.
<i>ccosl()</i>	m	m	.	.	.	m	.	.	.
<i>ceil()</i>	m	m	m	m	m	m	m	m	m
<i>ceilf()</i>	m	m	.	.	.	m	.	.	.
<i>ceilll()</i>	m	m	.	.	.	m	.	.	.
<i>cexp()</i>	m	m	.	.	.	m	.	.	.
<i>cexpf()</i>	m	m	.	.	.	m	.	.	.
<i>cexpl()</i>	m	m	.	.	.	m	.	.	.
<i>cfgetispeed()</i>	m	m	m	m	m	.	.	m	.
<i>cfgetospeed()</i>	m	m	m	m	m	.	.	m	.
<i>cfsetispeed()</i>	m	m	m	m	m	.	.	m	.
<i>cfsetospeed()</i>	m	m	m	m	m	.	.	m	.
<i>chdir()</i>	m	m	m	m	m	.	.	m	m
<i>chmod()</i>	m	m	m	m	m	.	.	m	m
<i>chown()</i>	m	m	m	m	m	.	.	m	m
<i>cimag()</i>	m	m	.	.	.	m	.	.	.
<i>cimagf()</i>	m	m	.	.	.	m	.	.	.
<i>cimagl()</i>	m	m	.	.	.	m	.	.	.
<i>clearerr()</i>	m	m	m	m	m	m	m	m	m
<i>clock()</i>	m	m	m	m	.	m	m	m	.
<i>clock_getcpuclockid()</i>	o	cpt
<i>clock_getres()</i>	o	tmr	o	.	r
<i>clock_gettime()</i>	o	tmr	o	.	r
<i>clock_nanosleep()</i>	o	cs
<i>clock_settime()</i>	o	tmr	o	.	r
<i>clog()</i>	m	m	.	.	.	m	.	.	.
<i>clogf()</i>	m	m	.	.	.	m	.	.	.
<i>clogl()</i>	m	m	.	.	.	m	.	.	.
<i>close()</i>	m	m	m	m	m	.	.	m	m
<i>closedir()</i>	m	m	m	m	m	.	.	m	m
<i>closelog()</i>	m	xsi	m	m	m
<i>confstr()</i>	m	m	m	m	m
<i>conj()</i>	m	m	.	.	.	m	.	.	.
<i>conjf()</i>	m	m	.	.	.	m	.	.	.
<i>conjl()</i>	m	m	.	.	.	m	.	.	.
<i>connect()</i>	m	m	m	m	m
<i>copysign()</i>	m	m	.	.	.	m	.	.	.
<i>copysignf()</i>	m	m	.	.	.	m	.	.	.
<i>copysignl()</i>	m	m	.	.	.	m	.	.	.
<i>cos()</i>	m	m	m	m	m	m	m	m	m
<i>cosf()</i>	m	m	.	.	.	m	.	.	.
<i>cosh()</i>	m	m	m	m	m	m	m	m	m
<i>coshf()</i>	m	m	.	.	.	m	.	.	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96 P92	C99	C89	SVID3	BSD
<i>coshl()</i>	m	m	.	.	.	m	.	.	.
<i>cosl()</i>	m	m	.	.	.	m	.	.	.
<i>cpow()</i>	m	m	.	.	.	m	.	.	.
<i>cpowf()</i>	m	m	.	.	.	m	.	.	.
<i>cpowl()</i>	m	m	.	.	.	m	.	.	.
<i>cproj()</i>	m	m	.	.	.	m	.	.	.
<i>cprojf()</i>	m	m	.	.	.	m	.	.	.
<i>cprojl()</i>	m	m	.	.	.	m	.	.	.
<i>creal()</i>	m	m	.	.	.	m	.	.	.
<i>crealf()</i>	m	m	.	.	.	m	.	.	.
<i>creall()</i>	m	m	.	.	.	m	.	.	.
<i>creat()</i>	m	m	m	m	m	.	.	m	m
<i>crypt()</i>	o	xsi	o	o	.	.	.	m	m
<i>csin()</i>	m	m	.	.	.	m	.	.	.
<i>csinf()</i>	m	m	.	.	.	m	.	.	.
<i>csinh()</i>	m	m	.	.	.	m	.	.	.
<i>csinhf()</i>	m	m	.	.	.	m	.	.	.
<i>csinhl()</i>	m	m	.	.	.	m	.	.	.
<i>csinl()</i>	m	m	.	.	.	m	.	.	.
<i>csqrt()</i>	m	m	.	.	.	m	.	.	.
<i>csqrtf()</i>	m	m	.	.	.	m	.	.	.
<i>csqrtl()</i>	m	m	.	.	.	m	.	.	.
<i>ctan()</i>	m	m	.	.	.	m	.	.	.
<i>ctanf()</i>	m	m	.	.	.	m	.	.	.
<i>ctanh()</i>	m	m	.	.	.	m	.	.	.
<i>ctanhf()</i>	m	m	.	.	.	m	.	.	.
<i>ctanhl()</i>	m	m	.	.	.	m	.	.	.
<i>ctanl()</i>	m	m	.	.	.	m	.	.	.
<i>ctermid()</i>	m	m	m	m	m	.	.	m	.
<i>ctime()</i>	m	m	m	m	m	m	m	m	m
<i>ctime_r()</i>	m	tsf	m	.	t
<i>daylight</i>	m	xsi	m	m	.	.	.	m	.
<i>dbm_clearerr()</i>	m	xsi	m	m
<i>dbm_close()</i>	m	xsi	m	m
<i>dbm_delete()</i>	m	xsi	m	m
<i>dbm_error()</i>	m	xsi	m	m
<i>dbm_fetch()</i>	m	xsi	m	m
<i>dbm_firstkey()</i>	m	xsi	m	m
<i>dbm_nextkey()</i>	m	xsi	m	m
<i>dbm_open()</i>	m	xsi	m	m
<i>dbm_store()</i>	m	xsi	m	m
<i>difftime()</i>	m	m	m	m	.	m	m	m	.
<i>dirname()</i>	m	xsi	m	m
<i>div()</i>	m	m	m	m	.	m	m	m	.
<i>dlclose()</i>	m	xsi	m
<i>dLError()</i>	m	xsi	m

Interface	XSI	POSIX Base	U98	U95	P96 P92	C99	C89	SVID3	BSD
<i>dlopen()</i>	m	xsi	m
<i>dlsym()</i>	m	xsi	m
<i>drand48()</i>	m	xsi	m	m	.	.	.	m	.
<i>dup()</i>	m	m	m	m	m	.	.	m	m
<i>dup2()</i>	m	m	m	m	m	.	.	m	m
<i>ecvt()</i>	l	xsi	m	m	m
<i>encrypt()</i>	o	xsi	o	o	.	.	.	m	m
<i>endgrent()</i>	m	xsi	m	m	.	.	.	m	m
<i>endhostent()</i>	m	m	m	m	m
<i>endnetent()</i>	m	m	m	m	m
<i>endprotoent()</i>	m	m	m	m	m
<i>endpwent()</i>	m	xsi	m	m	.	.	.	m	m
<i>endservent()</i>	m	m	m	m	m
<i>endutxent()</i>	m	xsi	m	m
<i>environ</i>	m	m	m	m	m	.	.	m	m
<i>erand48()</i>	m	xsi	m	m	.	.	.	m	.
<i>erf()</i>	m	m	m	m	.	m	.	m	m
<i>erfc()</i>	m	m	m	m	.	m	.	m	m
<i>erfcf()</i>	m	m	.	.	.	m	.	.	.
<i>erfcl()</i>	m	m	.	.	.	m	.	.	.
<i>erff()</i>	m	m	.	.	.	m	.	.	.
<i>erfl()</i>	m	m	.	.	.	m	.	.	.
<i>errno</i>	m	m	m	m	m	m	m	m	m
<i>execl()</i>	m	m	m	m	m	.	.	m	m
<i>execle()</i>	m	m	m	m	m	.	.	m	m
<i>execlp()</i>	m	m	m	m	m	.	.	m	m
<i>execv()</i>	m	m	m	m	m	.	.	m	m
<i>execve()</i>	m	m	m	m	m	.	.	m	m
<i>execvp()</i>	m	m	m	m	m	.	.	m	m
<i>exit()</i>	m	m	m	m	m	m	m	m	m
<i>exp()</i>	m	m	m	m	m	m	m	m	m
<i>exp2()</i>	m	m	.	.	.	m	.	.	.
<i>exp2f()</i>	m	m	.	.	.	m	.	.	.
<i>exp2l()</i>	m	m	.	.	.	m	.	.	.
<i>expf()</i>	m	m	.	.	.	m	.	.	.
<i>expl()</i>	m	m	.	.	.	m	.	.	.
<i>expm1()</i>	m	m	m	m	.	m	.	.	m
<i>expm1f()</i>	m	m	.	.	.	m	.	.	.
<i>expm1l()</i>	m	m	.	.	.	m	.	.	.
<i>fabs()</i>	m	m	m	m	m	m	m	m	m
<i>fabsf()</i>	m	m	.	.	.	m	.	.	.
<i>fabsl()</i>	m	m	.	.	.	m	.	.	.
<i>fattach()</i>	o	xsr	m	m	.	.	.	m	.
<i>fchdir()</i>	m	xsi	m	m	.	.	.	m	.
<i>fchmod()</i>	m	m	m	m	.	.	.	m	m
<i>fchown()</i>	m	m	m	m	.	.	.	m	m

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>fclose()</i>	m	m	m	m	m	m	m	m	m
<i>fcntl()</i>	m	m	m	m	m	.	.	m	m
<i>fcvt()</i>	l	xsi	m	m	m
<i>fdatasync()</i>	o	sio	o	.	r
<i>fdetach()</i>	o	xsr	m	m	.	.	.	m	.
<i>fdim()</i>	m	m	.	.	.	m	.	.	.
<i>fdimf()</i>	m	m	.	.	.	m	.	.	.
<i>fdiml()</i>	m	m	.	.	.	m	.	.	.
<i>fdopen()</i>	m	m	m	m	m	.	.	m	m
<i>feclearexcept()</i>	m	m	.	.	.	m	.	.	.
<i>fegetenv()</i>	m	m	.	.	.	m	.	.	.
<i>fegetexceptflag()</i>	m	m	.	.	.	m	.	.	.
<i>fegetround()</i>	m	m	.	.	.	m	.	.	.
<i>feholdexcept()</i>	m	m	.	.	.	m	.	.	.
<i>feof()</i>	m	m	m	m	m	m	m	m	m
<i>feraiseexcept()</i>	m	m	.	.	.	m	.	.	.
<i>ferror()</i>	m	m	m	m	m	m	m	m	m
<i>fesetenv()</i>	m	m	.	.	.	m	.	.	.
<i>fesetexceptflag()</i>	m	m	.	.	.	m	.	.	.
<i>fesetround()</i>	m	m	.	.	.	m	.	.	.
<i>fetestexcept()</i>	m	m	.	.	.	m	.	.	.
<i>feupdateenv()</i>	m	m	.	.	.	m	.	.	.
<i>fflush()</i>	m	m	m	m	m	m	m	m	m
<i>ffs()</i>	m	xsi	m	m	m
<i>fgetc()</i>	m	m	m	m	m	m	m	m	m
<i>fgetpos()</i>	m	m	m	m	.	m	m	m	.
<i>fgets()</i>	m	m	m	m	m	m	m	m	m
<i>fgetwc()</i>	m	m	m	m	.	m	1	.	.
<i>fgetws()</i>	m	m	m	m	.	m	1	.	.
<i>fileno()</i>	m	m	m	m	m	.	.	m	m
<i>flockfile()</i>	m	tsf	m	.	t
<i>floor()</i>	m	m	m	m	m	m	m	m	m
<i>floorf()</i>	m	m	.	.	.	m	.	.	.
<i>floorl()</i>	m	m	.	.	.	m	.	.	.
<i>fma()</i>	m	m	.	.	.	m	.	.	.
<i>fmaf()</i>	m	m	.	.	.	m	.	.	.
<i>fmal()</i>	m	m	.	.	.	m	.	.	.
<i>fmax()</i>	m	m	.	.	.	m	.	.	.
<i>fmaxf()</i>	m	m	.	.	.	m	.	.	.
<i>fmaxl()</i>	m	m	.	.	.	m	.	.	.
<i>fmin()</i>	m	m	.	.	.	m	.	.	.
<i>fminf()</i>	m	m	.	.	.	m	.	.	.
<i>fminl()</i>	m	m	.	.	.	m	.	.	.
<i>fmod()</i>	m	m	m	m	m	m	m	m	.
<i>fmodf()</i>	m	m	.	.	.	m	.	.	.
<i>fmodl()</i>	m	m	.	.	.	m	.	.	.

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>fmtmsg()</i>	m	xsi	m	m	.	.	.	m	.
<i>fnmatch()</i>	m	m	m	m	m
<i>fopen()</i>	m	m	m	m	m	m	m	m	m
<i>fork()</i>	m	m	m	m	m	.	.	m	m
<i>fpathconf()</i>	m	m	m	m	m	.	.	m	.
<i>fpclassify()</i>	m	m	.	.	.	m	.	.	.
<i>fprintf()</i>	m	m	m	m	m	m	m	m	m
<i>fputc()</i>	m	m	m	m	m	m	m	m	m
<i>fputs()</i>	m	m	m	m	m	m	m	m	m
<i>fputwc()</i>	m	m	m	m	.	m	1	.	.
<i>fputws()</i>	m	m	m	m	.	m	1	.	.
<i>fread()</i>	m	m	m	m	m	m	m	m	m
<i>free()</i>	m	m	m	m	m	m	m	m	m
<i>freeaddrinfo()</i>	m	m
<i>freopen()</i>	m	m	m	m	m	m	m	m	m
<i>frexp()</i>	m	m	m	m	m	m	m	m	m
<i>frexpf()</i>	m	m	.	.	.	m	.	.	.
<i>frexpl()</i>	m	m	.	.	.	m	.	.	.
<i>fscanf()</i>	m	m	m	m	m	m	m	m	m
<i>fseek()</i>	m	m	m	m	m	m	m	m	m
<i>fseeko()</i>	m	m	m
<i>fsetpos()</i>	m	m	m	m	.	m	m	m	.
<i>fstat()</i>	m	m	m	m	m	.	.	m	m
<i>fstatvfs()</i>	m	xsi	m	m	.	.	.	m	.
<i>fsync()</i>	m	fsc	m	m	.	.	.	m	m
<i>ftell()</i>	m	m	m	m	m	m	m	m	m
<i>ftello()</i>	m	m	m
<i>ftime()</i>	l	xsi	m	m	m
<i>ftok()</i>	m	xsi	m	m
<i>ftruncate()</i>	m	m	m	m
<i>ftwlockfile()</i>	m	tsf	m	.	t
<i>ftw()</i>	m	xsi	m	m	.	.	.	m	.
<i>funlockfile()</i>	m	tsf	m	.	t
<i>fwide()</i>	m	m	m	.	.	m	1	.	.
<i>fwprintf()</i>	m	m	m	.	.	m	1	.	.
<i>fwrite()</i>	m	m	m	m	m	m	m	m	m
<i>fwscanf()</i>	m	m	m	.	.	m	1	.	.
<i>gai_strerror()</i>	m	m
<i>gcvt()</i>	l	xsi	m	m	m
<i>getaddrinfo()</i>	m	m
<i>getc()</i>	m	m	m	m	m	m	m	m	m
<i>getc_unlocked()</i>	m	tsf	m	.	t
<i>getchar()</i>	m	m	m	m	m	m	m	m	m
<i>getchar_unlocked()</i>	m	tsf	m	.	t
<i>getcontext()</i>	m	xsi	m	m	.	.	.	m	.
<i>getcwd()</i>	m	m	m	m	m	.	.	m	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>getdate()</i>	m	xsi	m	m	.	.	.	m	.
<i>getdate_err</i>	m	xsi	m	m	.	.	.	m	.
<i>getegid()</i>	m	m	m	m	m	.	.	m	m
<i>getenv()</i>	m	m	m	m	m	m	m	m	m
<i>geteuid()</i>	m	m	m	m	m	.	.	m	m
<i>getgid()</i>	m	m	m	m	m	.	.	m	m
<i>getgrent()</i>	m	xsi	m	m	.	.	.	m	m
<i>getgrgid()</i>	m	m	m	m	m	.	.	m	m
<i>getgrgid_r()</i>	m	tsf	m	.	t
<i>getgrnam()</i>	m	m	m	m	m	.	.	m	.
<i>getgrnam_r()</i>	m	tsf	m	.	t
<i>getgroups()</i>	m	m	m	m	m	.	.	m	m
<i>gethostbyaddr()</i>	ob	ob	m	m	m
<i>gethostbyname()</i>	ob	ob	m	m	m
<i>gethostent()</i>	m	m	m	m	m
<i>gethostid()</i>	m	xsi	m	m	m
<i>gethostname()</i>	m	m	m	m	m
<i>getitimer()</i>	m	xsi	m	m	.	.	.	m	m
<i>getlogin()</i>	m	m	m	m	m	.	.	m	m
<i>getlogin_r()</i>	m	tsf	m	.	t
<i>getmsg()</i>	o	xsr	m	m	.	.	.	m	.
<i>getnameinfo()</i>	m	m
<i>getnetbyaddr()</i>	m	m	m	m	m
<i>getnetbyname()</i>	m	m	m	m	m
<i>getnetent()</i>	m	m	m	m	m
<i>getopt()</i>	m	m	m	m	m	.	.	m	m
<i>getpeername()</i>	m	m	m	m	m
<i>getpgid()</i>	m	xsi	m	m	.	.	.	m	.
<i>getpgrp()</i>	m	m	m	m	m	.	.	m	m
<i>getpid()</i>	m	m	m	m	m	.	.	m	m
<i>getpmsg()</i>	o	xsr	m	m	.	.	.	m	.
<i>getppid()</i>	m	m	m	m	m	.	.	m	m
<i>getpriority()</i>	m	xsi	m	m	m
<i>getprotobyname()</i>	m	m	m	m	m
<i>getprotobynumber()</i>	m	m	m	m	m
<i>getprotoent()</i>	m	m	m	m	m
<i>getpwent()</i>	m	xsi	m	m	.	.	.	m	m
<i>getpwnam()</i>	m	m	m	m	m	.	.	m	m
<i>getpwnam_r()</i>	m	tsf	m	.	t
<i>getpwuid()</i>	m	m	m	m	m	.	.	m	m
<i>getpwuid_r()</i>	m	tsf	m	.	t
<i>getrlimit()</i>	m	xsi	m	m	.	.	.	m	m
<i>getrusage()</i>	m	xsi	m	m	m
<i>gets()</i>	m	m	m	m	m	m	m	m	m
<i>getservbyname()</i>	m	m	m	m	m
<i>getservbyport()</i>	m	m	m	m	m

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>getservent()</i>	m	m	m	m	m
<i>getsid()</i>	m	xsi	m	m	.	.	.	m	.
<i>getsockname()</i>	m	m	m	m	m
<i>getsockopt()</i>	m	m	m	m	m
<i>getsubopt()</i>	m	xsi	m	m	.	.	.	m	.
<i>gettimeofday()</i>	m	xsi	m	m	.	.	.	m	m
<i>getuid()</i>	m	m	m	m	m	.	.	m	m
<i>getutxent()</i>	m	xsi	m	m
<i>getutxid()</i>	m	xsi	m	m
<i>getutxline()</i>	m	xsi	m	m
<i>getwc()</i>	m	m	m	m	.	m	1	.	.
<i>getwchar()</i>	m	m	m	m	.	m	1	.	.
<i>getwd()</i>	l	xsi	m	m	m
<i>glob()</i>	m	m	m	o	m
<i>globfree()</i>	m	m	m	o	m
<i>gmtime()</i>	m	m	m	m	m	m	m	m	m
<i>gmtime_r()</i>	m	tsf	m	.	t
<i>grantpt()</i>	m	xsi	m	m	.	.	.	m	.
<i>h_errno</i>	ob	ob
<i>hcreate()</i>	m	xsi	m	m	.	.	.	m	.
<i>hdestroy()</i>	m	xsi	m	m	.	.	.	m	.
<i>hsearch()</i>	m	xsi	m	m	.	.	.	m	.
<i>htonl()</i>	m	m	m	m	m
<i>htons()</i>	m	m	m	m	m
<i>hypot()</i>	m	m	m	m	.	m	.	m	m
<i>hypotf()</i>	m	m	.	.	.	m	.	.	.
<i>hypotl()</i>	m	m	.	.	.	m	.	.	.
<i>iconv()</i>	m	xsi	m	m
<i>iconv_close()</i>	m	xsi	m	m
<i>iconv_open()</i>	m	xsi	m	m
<i>if_freenameindex()</i>	m	m
<i>if_indextoname()</i>	m	m
<i>if_nameindex()</i>	m	m
<i>if_nametoindex()</i>	m	m
<i>ilogb()</i>	m	m	m	m	.	m	.	.	.
<i>ilogbf()</i>	m	m	.	.	.	m	.	.	.
<i>ilogbl()</i>	m	m	.	.	.	m	.	.	.
<i>imaxabs()</i>	m	m	.	.	.	m	.	.	.
<i>imaxdiv()</i>	m	m	.	.	.	m	.	.	.
<i>index()</i>	l	xsi	m	m	m
<i>inet_addr()</i>	m	m	m	m	m
<i>inet_ntoa()</i>	m	m	m	m	m
<i>inet_ntop()</i>	m	m
<i>inet_pton()</i>	m	m
<i>initstate()</i>	m	xsi	m	m	m
<i>insque()</i>	m	xsi	m	m	m

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>ioctl()</i>	o	xsr	m	m	.	.	.	m	m
<i>isalnum()</i>	m	m	m	m	m	m	m	m	m
<i>isalpha()</i>	m	m	m	m	m	m	m	m	m
<i>isascii()</i>	m	xsi	m	m	.	.	.	m	m
<i>isastream()</i>	o	xsr	m	m	.	.	.	m	.
<i>isatty()</i>	m	m	m	m	m	.	.	m	m
<i>isblank()</i>	m	m	.	.	.	m	.	.	.
<i>iscntrl()</i>	m	m	m	m	m	m	m	m	m
<i>isdigit()</i>	m	m	m	m	m	m	m	m	m
<i>isfinite()</i>	m	m	.	.	.	m	.	.	.
<i>isgraph()</i>	m	m	m	m	m	m	m	m	m
<i>isgreater()</i>	m	m	.	.	.	m	.	.	.
<i>isgreaterequal()</i>	m	m	.	.	.	m	.	.	.
<i>isinf()</i>	m	m	.	.	.	m	.	.	.
<i>isless()</i>	m	m	.	.	.	m	.	.	.
<i>islessequal()</i>	m	m	.	.	.	m	.	.	.
<i>islessgreater()</i>	m	m	.	.	.	m	.	.	.
<i>islower()</i>	m	m	m	m	m	m	m	m	m
<i>isnan()</i>	m	m	m	m	.	.	.	m	.
<i>isnormal()</i>	m	m	.	.	.	m	.	.	.
<i>isprint()</i>	m	m	m	m	m	m	m	m	m
<i>ispunct()</i>	m	m	m	m	m	m	m	m	m
<i>isspace()</i>	m	m	m	m	m	m	m	m	m
<i>isunordered()</i>	m	m	.	.	.	m	.	.	.
<i>isupper()</i>	m	m	m	m	m	m	m	m	m
<i>iswalnum()</i>	m	m	m	m	.	m	1	.	.
<i>iswalpha()</i>	m	m	m	m	.	m	1	.	.
<i>iswblank()</i>	m	m	.	.	.	m	.	.	.
<i>iswcntrl()</i>	m	m	m	m	.	m	1	.	.
<i>iswctype()</i>	m	m	m	m	.	m	1	.	.
<i>iswdigit()</i>	m	m	m	m	.	m	1	.	.
<i>iswgraph()</i>	m	m	m	m	.	m	1	.	.
<i>iswlower()</i>	m	m	m	m	.	m	1	.	.
<i>iswprint()</i>	m	m	m	m	.	m	1	.	.
<i>iswpunct()</i>	m	m	m	m	.	m	1	.	.
<i>iswspace()</i>	m	m	m	m	.	m	1	.	.
<i>iswupper()</i>	m	m	m	m	.	m	1	.	.
<i>iswxdigit()</i>	m	m	m	m	.	m	1	.	.
<i>isxdigit()</i>	m	m	m	m	m	m	m	m	m
<i>j0()</i>	m	xsi	m	m	.	.	.	m	m
<i>j1()</i>	m	xsi	m	m	.	.	.	m	m
<i>jn()</i>	m	xsi	m	m	.	.	.	m	m
<i>jrands48()</i>	m	xsi	m	m	.	.	.	m	.
<i>kill()</i>	m	m	m	m	m	.	.	m	m
<i>killpg()</i>	m	xsi	m	m	m
<i>l64a()</i>	m	xsi	m	m	.	.	.	m	.

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>labs()</i>	m	m	m	m	.	m	m	m	.
<i>lchown()</i>	m	xsi	m	m	.	.	.	m	.
<i>lcong48()</i>	m	xsi	m	m	.	.	.	m	.
<i>ldexp()</i>	m	m	m	m	m	m	m	m	m
<i>ldexpf()</i>	m	m	.	.	.	m	.	.	.
<i>ldexpl()</i>	m	m	.	.	.	m	.	.	.
<i>ldiv()</i>	m	m	m	m	.	m	m	m	.
<i>lfind()</i>	m	xsi	m	m	.	.	.	m	.
<i>lgamma()</i>	m	m	m	m	.	m	.	m	m
<i>lgammaf()</i>	m	m	.	.	.	m	.	.	.
<i>lgammal()</i>	m	m	.	.	.	m	.	.	.
<i>link()</i>	m	m	m	m	m	.	.	m	m
<i>lio_listio()</i>	o	aio	o	.	r
<i>listen()</i>	m	m	m	m	m
<i>llabs()</i>	m	m	.	.	.	m	.	.	.
<i>lldiv()</i>	m	m	.	.	.	m	.	.	.
<i>llrint()</i>	m	m	.	.	.	m	.	.	.
<i>llrintf()</i>	m	m	.	.	.	m	.	.	.
<i>llrintl()</i>	m	m	.	.	.	m	.	.	.
<i>llround()</i>	m	m	.	.	.	m	.	.	.
<i>llroundf()</i>	m	m	.	.	.	m	.	.	.
<i>llroundl()</i>	m	m	.	.	.	m	.	.	.
<i>localeconv()</i>	m	m	m	m	.	m	m	m	.
<i>localtime()</i>	m	m	m	m	m	m	m	m	m
<i>localtime_r()</i>	m	tsf	m	.	t
<i>lockf()</i>	m	xsi	m	m	.	.	.	m	.
<i>log()</i>	m	m	m	m	m	m	m	m	m
<i>log10()</i>	m	m	m	m	m	m	m	m	m
<i>log10f()</i>	m	m	.	.	.	m	.	.	.
<i>log10l()</i>	m	m	.	.	.	m	.	.	.
<i>log1p()</i>	m	m	m	m	.	m	.	.	m
<i>log1pf()</i>	m	m	.	.	.	m	.	.	.
<i>log1pl()</i>	m	m	.	.	.	m	.	.	.
<i>log2()</i>	m	m	.	.	.	m	.	.	.
<i>log2f()</i>	m	m	.	.	.	m	.	.	.
<i>log2l()</i>	m	m	.	.	.	m	.	.	.
<i>logb()</i>	m	m	m	m	.	m	.	m	m
<i>logbf()</i>	m	m	.	.	.	m	.	.	.
<i>logbl()</i>	m	m	.	.	.	m	.	.	.
<i>logf()</i>	m	m	.	.	.	m	.	.	.
<i>logl()</i>	m	m	.	.	.	m	.	.	.
<i>longjmp()</i>	m	m	m	m	m	m	m	m	m
<i>lrand48()</i>	m	xsi	m	m	.	.	.	m	.
<i>lrint()</i>	m	m	.	.	.	m	.	.	.
<i>lrintf()</i>	m	m	.	.	.	m	.	.	.
<i>lrintl()</i>	m	m	.	.	.	m	.	.	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>lround()</i>	m	m	.	.	.	m	.	.	.
<i>lroundf()</i>	m	m	.	.	.	m	.	.	.
<i>lroundl()</i>	m	m	.	.	.	m	.	.	.
<i>lsearch()</i>	m	xsi	m	m	.	.	.	m	.
<i>lseek()</i>	m	m	m	m	m	.	.	m	m
<i>lstat()</i>	m	m	m	m	.	.	.	m	m
<i>makecontext()</i>	m	xsi	m	m
<i>malloc()</i>	m	m	m	m	m	m	m	m	m
<i>mblen()</i>	m	m	m	m	.	m	m	m	.
<i>mbrlen()</i>	m	m	m	.	.	m	1	.	.
<i>mbrtowc()</i>	m	m	m	.	.	m	1	.	.
<i>mbsinit()</i>	m	m	m	.	.	m	1	.	.
<i>mbsrtowcs()</i>	m	m	m	.	.	m	1	.	.
<i>mbstowcs()</i>	m	m	m	m	.	m	m	m	.
<i>mbtowc()</i>	m	m	m	m	.	m	m	m	.
<i>memccpy()</i>	m	xsi	m	m	.	.	.	m	.
<i>memchr()</i>	m	m	m	m	.	m	m	m	.
<i>memcmp()</i>	m	m	m	m	.	m	m	m	.
<i>memcpy()</i>	m	m	m	m	.	m	m	m	.
<i>memmove()</i>	m	m	m	m	.	m	m	m	.
<i>memset()</i>	m	m	m	m	.	m	m	m	.
<i>mkdir()</i>	m	m	m	m	m	.	.	m	m
<i>mkfifo()</i>	m	m	m	m	m	.	.	m	.
<i>mknod()</i>	m	xsi	m	m	.	.	.	m	m
<i>mkstemp()</i>	m	xsi	m	m	m
<i>mktemp()</i>	l	xsi	m	m	.	.	.	m	m
<i>mtime()</i>	m	m	m	m	m	m	m	m	.
<i>mlock()</i>	o	mr	o	.	r
<i>mlockall()</i>	o	ml	o	.	r
<i>mmap()</i>	m	mflshmltym	m	m	.	.	.	m	.
<i>modf()</i>	m	m	m	m	m	m	m	m	m
<i>modff()</i>	m	m	.	.	.	m	.	.	.
<i>modfl()</i>	m	m	.	.	.	m	.	.	.
<i>mprotect()</i>	m	mpr	m	m	.	.	.	m	.
<i>mq_close()</i>	o	msg	o	.	r
<i>mq_getattr()</i>	o	msg	o	.	r
<i>mq_notify()</i>	o	msg	o	.	r
<i>mq_open()</i>	o	msg	o	.	r
<i>mq_receive()</i>	o	msg	o	.	r
<i>mq_send()</i>	o	msg	o	.	r
<i>mq_setattr()</i>	o	msg	o	.	r
<i>mq_timedreceive()</i>	o	msg
<i>mq_timedsend()</i>	o	msg
<i>mq_unlink()</i>	o	msg	o	.	r
<i>rand48()</i>	m	xsi	m	m	.	.	.	m	.
<i>msgctl()</i>	m	xsi	m	m	.	.	.	m	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>msgget()</i>	m	xsi	m	m	.	.	.	m	.
<i>msgrcv()</i>	m	xsi	m	m	.	.	.	m	.
<i>msgsnd()</i>	m	xsi	m	m	.	.	.	m	.
<i>msync()</i>	m	mf sio	m	m	.	.	.	m	.
<i>munlock()</i>	o	mr	o	.	r
<i>munlockall()</i>	o	ml	o	.	r
<i>munmap()</i>	m	mflshmltym	m	m	.	.	.	m	.
<i>nan()</i>	m	m	.	.	.	m	.	.	.
<i>nanf()</i>	m	m	.	.	.	m	.	.	.
<i>nanl()</i>	m	m	.	.	.	m	.	.	.
<i>nanosleep()</i>	o	tmr	o	.	r
<i>nearbyint()</i>	m	m	.	.	.	m	.	.	.
<i>nearbyintf()</i>	m	m	.	.	.	m	.	.	.
<i>nearbyintl()</i>	m	m	.	.	.	m	.	.	.
<i>nextafter()</i>	m	m	m	m	.	m	.	m	.
<i>nextafterf()</i>	m	m	.	.	.	m	.	.	.
<i>nextafterl()</i>	m	m	.	.	.	m	.	.	.
<i>nexttoward()</i>	m	m	.	.	.	m	.	.	.
<i>nexttowardf()</i>	m	m	.	.	.	m	.	.	.
<i>nexttowardl()</i>	m	m	.	.	.	m	.	.	.
<i>nftw()</i>	m	xsi	m	m	.	.	.	m	.
<i>nice()</i>	m	xsi	m	m	.	.	.	m	m
<i>nl_langinfo()</i>	m	xsi	m	m	.	.	.	m	.
<i>nrnd48()</i>	m	xsi	m	m	.	.	.	m	.
<i>ntohl()</i>	m	m	m	m	m
<i>ntohs()</i>	m	m	m	m	m
<i>open()</i>	m	m	m	m	m	.	.	m	m
<i>opendir()</i>	m	m	m	m	m	.	.	m	m
<i>openlog()</i>	m	xsi	m	m	m
<i>optarg</i>	m	m	m	m	m	.	.	m	m
<i>opterr</i>	m	m	m	m	m	.	.	m	m
<i>optind</i>	m	m	m	m	m	.	.	m	m
<i>optopt</i>	m	m	m	m	m	.	.	m	m
<i>pathconf()</i>	m	m	m	m	m	.	.	m	.
<i>pause()</i>	m	m	m	m	m	.	.	m	m
<i>pclose()</i>	m	m	m	m	m	.	.	m	m
<i>perror()</i>	m	m	m	m	m	m	m	m	m
<i>pipe()</i>	m	m	m	m	m	.	.	m	m
<i>poll()</i>	m	xsi	m	m	.	.	.	m	.
<i>popen()</i>	m	m	m	m	m	.	.	m	m
<i>posix_fadvise()</i>	o	adv
<i>posix_fallocate()</i>	o	adv
<i>posix_madvise()</i>	o	adv
<i>posix_mem_offset()</i>	o	tym
<i>posix_memalign()</i>	o	adv
<i>posix_openpt()</i>	m	xsi

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>posix_spawn()</i>	0	spn
<i>posix_spawn_file_actions_addclose()</i>	0	spn
<i>posix_spawn_file_actions_adddup2()</i>	0	spn
<i>posix_spawn_file_actions_addopen()</i>	0	spn
<i>posix_spawn_file_actions_destroy()</i>	0	spn
<i>posix_spawn_file_actions_init()</i>	0	spn
<i>posix_spawnattr_destroy()</i>	0	spn
<i>posix_spawnattr_getflags()</i>	0	spn
<i>posix_spawnattr_getpgroup()</i>	0	spn
<i>posix_spawnattr_getschedparam()</i>	0	spn ps
<i>posix_spawnattr_getschedpolicy()</i>	0	spn ps
<i>posix_spawnattr_getsigdefault()</i>	0	spn
<i>posix_spawnattr_getsigmask()</i>	0	spn
<i>posix_spawnattr_init()</i>	0	spn
<i>posix_spawnattr_setflags()</i>	0	spn
<i>posix_spawnattr_setpgroup()</i>	0	spn
<i>posix_spawnattr_setschedparam()</i>	0	spn ps
<i>posix_spawnattr_setschedpolicy()</i>	0	spn ps
<i>posix_spawnattr_setsigdefault()</i>	0	spn
<i>posix_spawnattr_setsigmask()</i>	0	spn
<i>posix_spawnp()</i>	0	spn
<i>posix_trace_attr_destroy()</i>	0	trc
<i>posix_trace_attr_getclockres()</i>	0	trc
<i>posix_trace_attr_getcreatetime()</i>	0	trc
<i>posix_trace_attr_getgenversion()</i>	0	trc
<i>posix_trace_attr_getinherited()</i>	0	trc tri
<i>posix_trace_attr_getlogfullpolicy()</i>	0	trc trl
<i>posix_trace_attr_getlogsize()</i>	0	trc trl
<i>posix_trace_attr_getmaxdatasize()</i>	0	trc
<i>posix_trace_attr_getmaxsystemeventsz()</i>	0	trc
<i>posix_trace_attr_getmaxusereventsz()</i>	0	trc
<i>posix_trace_attr_getname()</i>	0	trc
<i>posix_trace_attr_getstreamfullpolicy()</i>	0	trc
<i>posix_trace_attr_getstreamsize()</i>	0	trc
<i>posix_trace_attr_init()</i>	0	trc
<i>posix_trace_attr_setinherited()</i>	0	trc tri
<i>posix_trace_attr_setlogfullpolicy()</i>	0	trc trl
<i>posix_trace_attr_setlogsize()</i>	0	trc trl
<i>posix_trace_attr_setmaxdatasize()</i>	0	trc
<i>posix_trace_attr_setname()</i>	0	trc
<i>posix_trace_attr_setstreamfullpolicy()</i>	0	trc
<i>posix_trace_attr_setstreamsize()</i>	0	trc
<i>posix_trace_clear()</i>	0	trc
<i>posix_trace_close()</i>	0	trc trl
<i>posix_trace_create()</i>	0	trc
<i>posix_trace_create_withlog()</i>	0	trc trl

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>posix_trace_event()</i>	o	trc
<i>posix_trace_eventid_equal()</i>	o	trc
<i>posix_trace_eventid_get_name()</i>	o	trc
<i>posix_trace_eventid_open()</i>	o	trc
<i>posix_trace_eventset_add()</i>	o	trc tef
<i>posix_trace_eventset_del()</i>	o	trc tef
<i>posix_trace_eventset_empty()</i>	o	trc tef
<i>posix_trace_eventset_fill()</i>	o	trc tef
<i>posix_trace_eventset_ismember()</i>	o	trc tef
<i>posix_trace_eventtypelist_getnext_id()</i>	o	trc
<i>posix_trace_eventtypelist_rewind()</i>	o	trc
<i>posix_trace_flush()</i>	o	trc trl
<i>posix_trace_get_attr()</i>	o	trc
<i>posix_trace_get_filter()</i>	o	trc tef
<i>posix_trace_get_status()</i>	o	trc
<i>posix_trace_getnext_event()</i>	o	trc
<i>posix_trace_open()</i>	o	trc trl
<i>posix_trace_rewind()</i>	o	trc trl
<i>posix_trace_set_filter()</i>	o	trc tef
<i>posix_trace_shutdown()</i>	o	trc
<i>posix_trace_start()</i>	o	trc
<i>posix_trace_stop()</i>	o	trc
<i>posix_trace_timedgetnext_event()</i>	o	trc tmo
<i>posix_trace_trid_eventid_open()</i>	o	trc tef
<i>posix_trace_trygetnext_event()</i>	o	trc
<i>posix_typed_mem_get_info()</i>	o	tym
<i>posix_typed_mem_open()</i>	o	tym
<i>pow()</i>	m	m	m	m	m	m	m	m	m
<i>powf()</i>	m	m	.	.	.	m	.	.	.
<i>powl()</i>	m	m	.	.	.	m	.	.	.
<i>pread()</i>	m	xsi	m
<i>printf()</i>	m	m	m	m	m	m	m	m	m
<i>pselect()</i>	m	m
<i>pthread_atfork()</i>	m	thr	m	.	t
<i>pthread_attr_destroy()</i>	m	thr	m	.	t
<i>pthread_attr_getdetachstate()</i>	m	thr	m	.	t
<i>pthread_attr_getguardsize()</i>	m	xsi	m
<i>pthread_attr_getinheritsched()</i>	o	thr tps	o	.	t
<i>pthread_attr_getschedparam()</i>	m	thr	m	.	t
<i>pthread_attr_getschedpolicy()</i>	o	thr tps	o	.	t
<i>pthread_attr_getscope()</i>	o	thr tps	o	.	t
<i>pthread_attr_getstack()</i>	m	thr tsa tss
<i>pthread_attr_getstackaddr()</i>	m	thr tsa	m	.	t
<i>pthread_attr_getstacksize()</i>	m	thr tss	m	.	t
<i>pthread_attr_init()</i>	m	thr	m	.	t
<i>pthread_attr_setdetachstate()</i>	m	thr	m	.	t

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>pthread_attr_setguardsize()</i>	m	xsi	m
<i>pthread_attr_setinheritsched()</i>	o	thr tps	o	.	t
<i>pthread_attr_setschedparam()</i>	m	thr	m	.	t
<i>pthread_attr_setschedpolicy()</i>	o	thr tps	o	.	t
<i>pthread_attr_setscope()</i>	o	thr tps	o	.	t
<i>pthread_attr_setstack()</i>	m	thr tsa tss
<i>pthread_attr_setstackaddr()</i>	m	thr tsa	m	.	t
<i>pthread_attr_setstacksize()</i>	m	thr tss	m	.	t
<i>pthread_barrier_destroy()</i>	o	thr bar
<i>pthread_barrier_init()</i>	o	thr bar
<i>pthread_barrier_wait()</i>	o	thr bar
<i>pthread_barrierattr_destroy()</i>	o	thr bar
<i>pthread_barrierattr_getpshared()</i>	o	thr bar tsh
<i>pthread_barrierattr_init()</i>	o	thr bar
<i>pthread_barrierattr_setpshared()</i>	o	thr bar tsh
<i>pthread_cancel()</i>	m	thr	m	.	t
<i>pthread_cleanup_pop()</i>	m	thr	m	.	t
<i>pthread_cleanup_push()</i>	m	thr	m	.	t
<i>pthread_cond_broadcast()</i>	m	thr	m	.	t
<i>pthread_cond_destroy()</i>	m	thr	m	.	t
<i>pthread_cond_init()</i>	m	thr	m	.	t
<i>pthread_cond_signal()</i>	m	thr	m	.	t
<i>pthread_cond_timedwait()</i>	m	thr	m	.	t
<i>pthread_cond_wait()</i>	m	thr	m	.	t
<i>pthread_condattr_destroy()</i>	m	thr	m	.	t
<i>pthread_condattr_getclock()</i>	o	thr cs
<i>pthread_condattr_getpshared()</i>	m	thr tsh	m	.	t
<i>pthread_condattr_init()</i>	m	thr	m	.	t
<i>pthread_condattr_setclock()</i>	o	thr cs
<i>pthread_condattr_setpshared()</i>	m	thr tsh	m	.	t
<i>pthread_create()</i>	m	thr	m	.	t
<i>pthread_detach()</i>	m	thr	m	.	t
<i>pthread_equal()</i>	m	thr	m	.	t
<i>pthread_exit()</i>	m	thr	m	.	t
<i>pthread_getconcurrency()</i>	m	xsi	m
<i>pthread_getcpuclockid()</i>	o	thr tct
<i>pthread_getschedparam()</i>	o	thr tps	o	.	t
<i>pthread_getspecific()</i>	m	thr	m	.	t
<i>pthread_join()</i>	m	thr	m	.	t
<i>pthread_key_create()</i>	m	thr	m	.	t
<i>pthread_key_delete()</i>	m	thr	m	.	t
<i>pthread_kill()</i>	m	thr	m	.	t
<i>pthread_mutex_destroy()</i>	m	thr	m	.	t
<i>pthread_mutex_getprioceiling()</i>	o	thr tpp	o	.	t
<i>pthread_mutex_init()</i>	m	thr	m	.	t
<i>pthread_mutex_lock()</i>	m	thr	m	.	t

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>pthread_mutex_setprioceiling()</i>	o	thr tpp	o	.	t
<i>pthread_mutex_timedlock()</i>	o	thr tmo
<i>pthread_mutex_trylock()</i>	m	thr	m	.	t
<i>pthread_mutex_unlock()</i>	m	thr	m	.	t
<i>pthread_mutexattr_destroy()</i>	m	thr	m	.	t
<i>pthread_mutexattr_getprioceiling()</i>	o	thr tpp	o	.	t
<i>pthread_mutexattr_getprotocol()</i>	o	thr tpp tpi	o	.	t
<i>pthread_mutexattr_getpshared()</i>	m	thr tsh	m	.	t
<i>pthread_mutexattr_gettype()</i>	m	xsi	m
<i>pthread_mutexattr_init()</i>	m	thr	m	.	t
<i>pthread_mutexattr_setprioceiling()</i>	o	thr tpp	o	.	t
<i>pthread_mutexattr_setprotocol()</i>	o	thr tpp tpi	o	.	t
<i>pthread_mutexattr_setpshared()</i>	m	thr tsh	m	.	t
<i>pthread_mutexattr_settype()</i>	m	xsi	m
<i>pthread_once()</i>	m	thr	m	.	t
<i>pthread_rwlock_destroy()</i>	m	thr	m
<i>pthread_rwlock_init()</i>	m	thr	m
<i>pthread_rwlock_rdlock()</i>	m	thr	m
<i>pthread_rwlock_timedrdlock()</i>	o	thr tmo
<i>pthread_rwlock_timedwrlock()</i>	o	thr tmo
<i>pthread_rwlock_tryrdlock()</i>	m	thr	m
<i>pthread_rwlock_trywrlock()</i>	m	thr	m
<i>pthread_rwlock_unlock()</i>	m	thr	m
<i>pthread_rwlock_wrlock()</i>	m	thr	m
<i>pthread_rwlockattr_destroy()</i>	m	thr	m
<i>pthread_rwlockattr_getpshared()</i>	m	thr tsh	m
<i>pthread_rwlockattr_init()</i>	m	thr	m
<i>pthread_rwlockattr_setpshared()</i>	m	thr tsh	m
<i>pthread_self()</i>	m	thr	m	.	t
<i>pthread_setcancelstate()</i>	m	thr	m	.	t
<i>pthread_setcanceltype()</i>	m	thr	m	.	t
<i>pthread_setconcurrency()</i>	m	xsi	m
<i>pthread_setschedparam()</i>	o	thr tps	o	.	t
<i>pthread_setschedprio()</i>	o	thr tps
<i>pthread_setspecific()</i>	m	thr	m	.	t
<i>pthread_sigmask()</i>	m	thr	m	.	t
<i>pthread_spin_destroy()</i>	o	thr spi
<i>pthread_spin_init()</i>	o	thr spi
<i>pthread_spin_lock()</i>	o	thr spi
<i>pthread_spin_trylock()</i>	o	thr spi
<i>pthread_spin_unlock()</i>	o	thr spi
<i>pthread_testcancel()</i>	m	thr	m	.	t
<i>ptsname()</i>	m	xsi	m	m	.	.	.	m	.
<i>putc()</i>	m	m	m	m	m	m	m	m	m
<i>putc_unlocked()</i>	m	tsf	m	.	t
<i>putchar()</i>	m	m	m	m	m	m	m	m	m

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>putchar_unlocked()</i>	m	tsf	m	.	t
<i>putenv()</i>	m	xsi	m	m	.	.	.	m	.
<i>putmsg()</i>	o	xsr	m	m	.	.	.	m	.
<i>putpmsg()</i>	o	xsr	m	m	.	.	.	m	.
<i>puts()</i>	m	m	m	m	m	m	m	m	m
<i>pututxline()</i>	m	xsi	m	m
<i>putwc()</i>	m	m	m	m	.	m	1	.	.
<i>putwchar()</i>	m	m	m	m	.	m	1	.	.
<i>pwrite()</i>	m	xsi	m
<i>qsort()</i>	m	m	m	m	m	m	m	m	m
<i>raise()</i>	m	m	m	m	.	m	m	m	.
<i>rand()</i>	m	m	m	m	m	m	m	m	m
<i>rand_r()</i>	m	tsf	m	.	t
<i>random()</i>	m	xsi	m	m	m
<i>read()</i>	m	xsi	m	m	m	.	.	m	m
<i>readdir()</i>	m	m	m	m	m	.	.	m	m
<i>readdir_r()</i>	m	tsf	m	.	t
<i>readlink()</i>	m	m	m	m	.	.	.	m	m
<i>readv()</i>	m	xsi	m	m	.	.	.	m	m
<i>realloc()</i>	m	m	m	m	m	m	m	m	m
<i>realpath()</i>	m	xsi	m	m
<i>recv()</i>	m	m	m	m	m
<i>recvfrom()</i>	m	m	m	m	m
<i>recvmsg()</i>	m	m	m	m	m
<i>regcomp()</i>	m	m	m	o	m
<i>regerror()</i>	m	m	m	o	m
<i>regexec()</i>	m	m	m	o	m
<i>regfree()</i>	m	m	m	o	m
<i>remainder()</i>	m	m	m	m	.	m	.	m	.
<i>remainderf()</i>	m	m	.	.	.	m	.	.	.
<i>remainderl()</i>	m	m	.	.	.	m	.	.	.
<i>remove()</i>	m	m	m	m	m	m	m	m	.
<i>remque()</i>	m	xsi	m	m	m
<i>remquo()</i>	m	m	.	.	.	m	.	.	.
<i>remquof()</i>	m	m	.	.	.	m	.	.	.
<i>remquol()</i>	m	m	.	.	.	m	.	.	.
<i>rename()</i>	m	m	m	m	m	m	m	m	m
<i>rewind()</i>	m	m	m	m	m	m	m	m	m
<i>rewinddir()</i>	m	m	m	m	m	.	.	m	m
<i>rindex()</i>	l	xsi	m	m	m
<i>rint()</i>	m	m	m	m	.	m	.	.	m
<i>rintf()</i>	m	m	.	.	.	m	.	.	.
<i>rintl()</i>	m	m	.	.	.	m	.	.	.
<i>rmdir()</i>	m	m	m	m	m	.	.	m	m
<i>round()</i>	m	m	.	.	.	m	.	.	.
<i>roundf()</i>	m	m	.	.	.	m	.	.	.

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>roundl()</i>	m	m	.	.	.	m	.	.	.
<i>scalb()</i>	ob	xsi	m	m	.	.	.	m	m
<i>scalbln()</i>	m	m	.	.	.	m	.	.	.
<i>scalblnf()</i>	m	m	.	.	.	m	.	.	.
<i>scalbln()</i>	m	m	.	.	.	m	.	.	.
<i>scalbn()</i>	m	m	.	.	.	m	.	.	.
<i>scalbnf()</i>	m	m	.	.	.	m	.	.	.
<i>scalbnl()</i>	m	m	.	.	.	m	.	.	.
<i>scanf()</i>	m	m	m	m	m	m	m	m	m
<i>sched_get_priority_max()</i>	o	ps	o	.	r
<i>sched_get_priority_min()</i>	o	ps	o	.	r
<i>sched_getparam()</i>	o	ps	o	.	r
<i>sched_getscheduler()</i>	o	ps	o	.	r
<i>sched_rr_get_interval()</i>	o	ps	o	.	r
<i>sched_setparam()</i>	o	ps	o	.	r
<i>sched_setscheduler()</i>	o	ps	o	.	r
<i>sched_yield()</i>	m	pslthr	m	.	r
<i>seed48()</i>	m	xsi	m	m	.	.	.	m	.
<i>seekdir()</i>	m	xsi	m	m	.	.	.	m	m
<i>select()</i>	m	m	m	m	m
<i>sem_close()</i>	o	sem	o	.	r
<i>sem_destroy()</i>	o	sem	o	.	r
<i>sem_getvalue()</i>	o	sem	o	.	r
<i>sem_init()</i>	o	sem	o	.	r
<i>sem_open()</i>	o	sem	o	.	r
<i>sem_post()</i>	o	sem	o	.	r
<i>sem_timedwait()</i>	o	sem tmo
<i>sem_trywait()</i>	o	sem	o	.	r
<i>sem_unlink()</i>	o	sem	o	.	r
<i>sem_wait()</i>	o	sem	o	.	r
<i>semctl()</i>	m	xsi	m	m	.	.	.	m	.
<i>semget()</i>	m	xsi	m	m	.	.	.	m	.
<i>semop()</i>	m	xsi	m	m	.	.	.	m	.
<i>send()</i>	m	m	m	m	m
<i>sendmsg()</i>	m	m	m	m	m
<i>sendto()</i>	m	m	m	m	m
<i>setbuf()</i>	m	m	m	m	m	m	m	m	m
<i>setcontext()</i>	m	xsi	m	m	.	.	.	m	.
<i>setegid()</i>	m	m
<i>setenv()</i>	m	m
<i>seteuid()</i>	m	m
<i>setgid()</i>	m	m	m	m	m	.	.	m	m
<i>setgrent()</i>	m	xsi	m	m	.	.	.	m	m
<i>sethostent()</i>	m	m	m	m	m
<i>setitimer()</i>	m	xsi	m	m	.	.	.	m	m
<i>setjmp()</i>	m	m	m	m	m	m	m	m	m

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>setkey()</i>	o	xsi	o	o	.	.	.	m	m
<i>setlocale()</i>	m	m	m	m	m	m	m	m	.
<i>setlogmask()</i>	m	xsi	m	m	m
<i>setnetent()</i>	m	m	m	m	m
<i>setpgid()</i>	m	m	m	m	m	.	.	m	.
<i>setpgrp()</i>	m	xsi	m	m	.	.	.	m	m
<i>setpriority()</i>	m	xsi	m	m	m
<i>setprotoent()</i>	m	m	m	m	m
<i>setpwent()</i>	m	xsi	m	m	.	.	.	m	m
<i>setregid()</i>	m	xsi	m	m	m
<i>setreuid()</i>	m	xsi	m	m	m
<i>setrlimit()</i>	m	xsi	m	m	.	.	.	m	m
<i>setservent()</i>	m	m	m	m	m
<i>setsid()</i>	m	m	m	m	m	.	.	m	.
<i>setsockopt()</i>	m	m	m	m	m
<i>setstate()</i>	m	xsi	m	m	m
<i>setuid()</i>	m	m	m	m	m	.	.	m	m
<i>setutxent()</i>	m	xsi	m	m
<i>setvbuf()</i>	m	m	m	m	.	m	m	m	.
<i>shm_open()</i>	o	shm	o	.	r
<i>shm_unlink()</i>	o	shm	o	.	r
<i>shmat()</i>	m	xsi	m	o	.	.	.	m	.
<i>shmctl()</i>	m	xsi	m	o	.	.	.	m	.
<i>shmdt()</i>	m	xsi	m	o	.	.	.	m	.
<i>shmget()</i>	m	xsi	m	o	.	.	.	m	.
<i>shutdown()</i>	m	m	m	m	m
<i>sigaction()</i>	m	m	m	m	m	.	.	m	.
<i>sigaddset()</i>	m	m	m	m	m	.	.	m	.
<i>sigaltstack()</i>	m	xsi	m	m	.	.	.	m	.
<i>sigdelset()</i>	m	m	m	m	m	.	.	m	.
<i>sigemptyset()</i>	m	m	m	m	m	.	.	m	.
<i>sigfillset()</i>	m	m	m	m	m	.	.	m	.
<i>sighold()</i>	m	xsi	m	m	.	.	.	m	.
<i>sigignore()</i>	m	xsi	m	m	.	.	.	m	.
<i>siginterrupt()</i>	m	xsi	m	m	m
<i>sigismember()</i>	m	m	m	m	m	.	.	m	.
<i>siglongjmp()</i>	m	m	m	m	m	.	.	m	.
<i>signal()</i>	m	m	m	m	.	m	m	m	m
<i>signbil()</i>	m	m	.	.	.	m	.	.	.
<i>signgam</i>	m	xsi	m	.
<i>sigpause()</i>	m	xsi	m	m	.	.	.	m	m
<i>sigpending()</i>	m	m	m	m	m	.	.	m	.
<i>sigprocmask()</i>	m	thr	m	m	m	.	.	m	.
<i>sigqueue()</i>	o	rts	o	.	r
<i>sigrelse()</i>	m	xsi	m	m	.	.	.	m	.
<i>sigset()</i>	m	xsi	m	m	.	.	.	m	.

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>sigsetjmp()</i>	m	m	m	m	m	.	.	m	.
<i>sigsuspend()</i>	m	m	m	m	m	.	.	m	.
<i>sigtimedwait()</i>	o	rts	o	.	r
<i>sigwait()</i>	m	m	m	.	t
<i>sigwaitinfo()</i>	o	rts	o	.	r
<i>sin()</i>	m	m	m	m	m	m	m	m	m
<i>sinf()</i>	m	m	.	.	.	m	.	.	.
<i>sinh()</i>	m	m	m	m	m	m	m	m	m
<i>sinhf()</i>	m	m	.	.	.	m	.	.	.
<i>sinhl()</i>	m	m	.	.	.	m	.	.	.
<i>sinl()</i>	m	m	.	.	.	m	.	.	.
<i>sleep()</i>	m	m	m	m	m	.	.	m	m
<i>snprintf()</i>	m	m	m	.	.	m	.	.	.
<i>socketatmark()</i>	m	m
<i>socket()</i>	m	m	m	m	m
<i>socketpair()</i>	m	m	m	m	m
<i>sprintf()</i>	m	m	m	m	m	m	m	m	m
<i>sqrt()</i>	m	m	m	m	m	m	m	m	m
<i>sqrtf()</i>	m	m	.	.	.	m	.	.	.
<i>sqrtl()</i>	m	m	.	.	.	m	.	.	.
<i>srand()</i>	m	m	m	m	m	m	m	m	m
<i>srand48()</i>	m	xsi	m	m	.	.	.	m	.
<i>srandom()</i>	m	xsi	m	m	m
<i>sscanf()</i>	m	m	m	m	m	m	m	m	m
<i>stat()</i>	m	m	m	m	m	.	.	m	m
<i>statvfs()</i>	m	xsi	m	m	.	.	.	m	.
<i>stderr</i>	m	m	m	m	m	m	m	m	m
<i>stdin</i>	m	m	m	m	m	m	m	m	m
<i>stdout</i>	m	m	m	m	m	m	m	m	m
<i>strcasecmp()</i>	m	xsi	m	m
<i>strcat()</i>	m	m	m	m	m	m	m	m	m
<i>strchr()</i>	m	m	m	m	m	m	m	m	.
<i>strcmp()</i>	m	m	m	m	m	m	m	m	m
<i>strcoll()</i>	m	m	m	m	.	m	m	m	.
<i>strcpy()</i>	m	m	m	m	m	m	m	m	m
<i>strcspn()</i>	m	m	m	m	m	m	m	m	.
<i>strdup()</i>	m	xsi	m	m	.	.	.	m	.
<i>strerror()</i>	m	m	m	m	.	m	m	m	.
<i>strerror_r()</i>	m	tsf
<i>strfmon()</i>	m	xsi	m	o
<i>strftime()</i>	m	m	m	m	m	m	m	m	.
<i>strlen()</i>	m	m	m	m	m	m	m	m	m
<i>strncasecmp()</i>	m	xsi	m	m
<i>strncat()</i>	m	m	m	m	m	m	m	m	m
<i>strncmp()</i>	m	m	m	m	m	m	m	m	m
<i>strncpy()</i>	m	m	m	m	m	m	m	m	m

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>strpbrk()</i>	m	m	m	m	m	m	m	m	.
<i>strptime()</i>	m	xsi	m	o
<i>strchr()</i>	m	m	m	m	m	m	m	m	.
<i>strspn()</i>	m	m	m	m	m	m	m	m	.
<i>strstr()</i>	m	m	m	m	m	m	m	m	.
<i>strtod()</i>	m	m	m	m	.	m	m	m	.
<i>strtof()</i>	m	m	.	.	.	m	.	.	.
<i>strtoimax()</i>	m	m	.	.	.	m	.	.	.
<i>strtok()</i>	m	m	m	m	m	m	m	m	.
<i>strtok_r()</i>	m	tsf	m	.	t
<i>strtol()</i>	m	m	m	m	.	m	m	m	.
<i>strtold()</i>	m	m	.	.	.	m	.	.	.
<i>strtoll()</i>	m	m	.	.	.	m	.	.	.
<i>strtoul()</i>	m	m	m	m	.	m	m	m	.
<i>strtoull()</i>	m	m	.	.	.	m	.	.	.
<i>strtoumax()</i>	m	m	.	.	.	m	.	.	.
<i>strxfrm()</i>	m	m	m	m	.	m	m	m	.
<i>swab()</i>	m	xsi	m	m	.	.	.	m	m
<i>swapcontext()</i>	m	xsi	m	m
<i>swprintf()</i>	m	m	m	.	.	m	1	.	.
<i>swscanf()</i>	m	m	m	.	.	m	1	.	.
<i>symlink()</i>	m	m	m	m	.	.	.	m	m
<i>sync()</i>	m	xsi	m	m	.	.	.	m	m
<i>sysconf()</i>	m	m	m	m	m	.	.	m	.
<i>syslog()</i>	m	xsi	m	m	m
<i>system()</i>	m	m	m	m	.	m	m	m	m
<i>tan()</i>	m	m	m	m	m	m	m	m	m
<i>tanf()</i>	m	m	.	.	.	m	.	.	.
<i>tanh()</i>	m	m	m	m	m	m	m	m	m
<i>tanhf()</i>	m	m	.	.	.	m	.	.	.
<i>tanhl()</i>	m	m	.	.	.	m	.	.	.
<i>tanl()</i>	m	m	.	.	.	m	.	.	.
<i>tcdrain()</i>	m	m	m	m	m	.	.	m	.
<i>tcfldw()</i>	m	m	m	m	m	.	.	m	.
<i>tcfldsh()</i>	m	m	m	m	m	.	.	m	.
<i>tcgetattr()</i>	m	m	m	m	m	.	.	m	.
<i>tcgetpgrp()</i>	m	m	m	m	m	.	.	m	.
<i>tcgetsid()</i>	m	xsi	m	m	.	.	.	m	.
<i>tcsendbreak()</i>	m	m	m	m	m	.	.	m	.
<i>tcsetattr()</i>	m	m	m	m	m	.	.	m	.
<i>tcsetpgrp()</i>	m	m	m	m	m	.	.	m	.
<i>tdelete()</i>	m	xsi	m	m	.	.	.	m	.
<i>telldir()</i>	m	xsi	m	m	.	.	.	m	m
<i>tempnam()</i>	m	xsi	m	m	.	.	.	m	.
<i>tfind()</i>	m	xsi	m	m	.	.	.	m	.
<i>tgamma()</i>	m	m	.	.	.	m	.	.	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96	C99	C89	SVID3	BSD
<i>tgammalf()</i>	m	m	.	.	.	m	.	.	.
<i>tgammal()</i>	m	m	.	.	.	m	.	.	.
<i>time()</i>	m	m	m	m	m	m	m	m	m
<i>timer_create()</i>	o	tmr	o	.	r
<i>timer_delete()</i>	o	tmr	o	.	r
<i>timer_getoverrun()</i>	o	tmr	o	.	r
<i>timer_gettime()</i>	o	tmr	o	.	r
<i>timer_settime()</i>	o	tmr	o	.	r
<i>times()</i>	m	m	m	m	m	.	.	m	m
<i>timezone</i>	m	xsi	m	m	.	.	.	m	.
<i>tmpfile()</i>	m	m	m	m	m	m	m	m	.
<i>tmpnam()</i>	m	m	m	m	m	m	m	m	.
<i>toascii()</i>	m	xsi	m	m	.	.	.	m	m
<i>tolower()</i>	m	m	m	m	m	m	m	m	m
<i>toupper()</i>	m	m	m	m	m	m	m	m	m
<i>towctrans()</i>	m	m	m	.	.	m	1	.	.
<i>towlower()</i>	m	m	m	m	.	m	1	.	.
<i>towupper()</i>	m	m	m	m	.	m	1	.	.
<i>trunc()</i>	m	m	.	.	.	m	.	.	.
<i>truncate()</i>	m	xsi	m	m	m
<i>truncf()</i>	m	m	.	.	.	m	.	.	.
<i>truncl()</i>	m	m	.	.	.	m	.	.	.
<i>tsearch()</i>	m	xsi	m	m	.	.	.	m	.
<i>ttyname()</i>	m	tsf	m	m	m	.	.	m	m
<i>ttyname_r()</i>	m	tsf	m	.	t
<i>twalk()</i>	m	xsi	m	m	.	.	.	m	.
<i>tzname</i>	m	xsi	m	m	m	.	.	m	.
<i>tzset()</i>	m	xsi	m	m	m	.	.	m	.
<i>ualarm()</i>	ob	xsi	m	m	m
<i>ulimit()</i>	m	xsi	m	m	.	.	.	m	.
<i>umask()</i>	m	m	m	m	m	.	.	m	m
<i>uname()</i>	m	m	m	m	m	.	.	m	.
<i>ungetc()</i>	m	m	m	m	m	m	m	m	m
<i>ungetwc()</i>	m	m	m	m	.	m	1	.	.
<i>unlink()</i>	m	m	m	m	m	.	.	m	m
<i>unlockpt()</i>	m	xsi	m	m	.	.	.	m	.
<i>unsetenv()</i>	m	m
<i>usleep()</i>	ob	xsi	m	m	m
<i>utime()</i>	m	m	m	m	m	.	.	m	m
<i>utimes()</i>	l	xsi	m	m	m
<i>va_arg()</i>	m	m
<i>va_copy()</i>	m	m
<i>va_end()</i>	m	m
<i>va_start()</i>	m	m
<i>vfork()</i>	ob	xsi	m	m	m
<i>vfprintf()</i>	m	m	m	m	.	.	.	m	.

System Interface Table

Interface	XSI	POSIX Base	U98	U95	P96 P92	C99	C89	SVID3	BSD
<i>vfscanf()</i>	m	m	.	.	.	m	.	.	.
<i>vwprintf()</i>	m	m	m	.	.	m	1	.	.
<i>vwscanf()</i>	m	m	.	.	.	m	.	.	.
<i>vprintf()</i>	m	m	m	m	.	m	m	m	.
<i>vscanf()</i>	m	m	.	.	.	m	.	.	.
<i>vsprintf()</i>	m	m	m	.	.	m	.	.	.
<i>vsprintf()</i>	m	m	m	m	.	m	m	m	.
<i>vsscanf()</i>	m	m	.	.	.	m	.	.	.
<i>vswprintf()</i>	m	m	m	.	.	m	1	.	.
<i>vswscanf()</i>	m	m	.	.	.	m	.	.	.
<i>vwprintf()</i>	m	m	m	.	.	m	1	.	.
<i>vwscanf()</i>	m	m	.	.	.	m	.	.	.
<i>wait()</i>	m	m	m	m	m	.	.	m	m
<i>waitid()</i>	m	xsi	m	m	.	.	.	m	.
<i>waitpid()</i>	m	m	m	m	m	.	.	m	m
<i>wcrtomb()</i>	m	m	m	.	.	m	1	.	.
<i>wcscat()</i>	m	m	m	m	.	m	1	.	.
<i>wcscchr()</i>	m	m	m	m	.	m	1	.	.
<i>wcscmp()</i>	m	m	m	m	.	m	1	.	.
<i>wcscoll()</i>	m	m	m	o	.	m	1	.	.
<i>wcscopy()</i>	m	m	m	m	.	m	1	.	.
<i>wcscspn()</i>	m	m	m	m	.	m	1	.	.
<i>wcsftime()</i>	m	m	m	o	.	m	1	.	.
<i>wcslen()</i>	m	m	m	m	.	m	1	.	.
<i>wcsncat()</i>	m	m	m	m	.	m	1	.	.
<i>wcsncmp()</i>	m	m	m	m	.	m	1	.	.
<i>wcsncpy()</i>	m	m	m	m	.	m	1	.	.
<i>wcspbrk()</i>	m	m	m	m	.	m	1	.	.
<i>wcsrchr()</i>	m	m	m	m	.	m	1	.	.
<i>wcsrtombs()</i>	m	m	m	.	.	m	1	.	.
<i>wcsspn()</i>	m	m	m	m	.	m	1	.	.
<i>wcsstr()</i>	m	m	m	.	.	m	1	.	.
<i>wcstod()</i>	m	m	m	m	.	m	1	.	.
<i>wcstof()</i>	m	m	.	.	.	m	.	.	.
<i>wcstoimax()</i>	m	m	.	.	.	m	.	.	.
<i>wcstok()</i>	m	m	m	m	.	m	1	.	.
<i>wcstol()</i>	m	m	m	m	.	m	1	.	.
<i>wcstold()</i>	m	m	.	.	.	m	.	.	.
<i>wcstoll()</i>	m	m	.	.	.	m	.	.	.
<i>wcstombs()</i>	m	m	m	m	.	m	m	m	.
<i>wcstoul()</i>	m	m	m	m	.	m	1	.	.
<i>wcstoull()</i>	m	m	.	.	.	m	.	.	.
<i>wcstoumax()</i>	m	m	.	.	.	m	.	.	.
<i>wcswcs()</i>	l	xsi	m	m
<i>wcswidth()</i>	m	xsi	m	m
<i>wcsxfrm()</i>	m	m	m	o	.	m	1	.	.

Interface	XSI	POSIX Base	U98	U95	P96 P92	C99	C89	SVID3	BSD
<i>wctob()</i>	m	m	m	.	.	m	1	.	.
<i>wctomb()</i>	m	m	m	m	.	m	m	m	.
<i>wctrans()</i>	m	m	m	.	.	m	1	.	.
<i>wctype()</i>	m	m	m	m	.	m	1	.	.
<i>wcwidth()</i>	m	xsi	m	m
<i>wmemchr()</i>	m	m	m	.	.	m	1	.	.
<i>wmemcmp()</i>	m	m	m	.	.	m	1	.	.
<i>wmemcpy()</i>	m	m	m	.	.	m	1	.	.
<i>wmemmove()</i>	m	m	m	.	.	m	1	.	.
<i>wmemset()</i>	m	m	m	.	.	m	1	.	.
<i>wordexp()</i>	m	m	m	o	m
<i>wordfree()</i>	m	m	m	o	m
<i>wprintf()</i>	m	m	m	.	.	m	1	.	.
<i>write()</i>	m	xsi	m	m	m	.	.	m	m
<i>writev()</i>	m	xsi	m	m	.	.	.	m	m
<i>wscanf()</i>	m	m	m	.	.	m	1	.	.
<i>y0()</i>	m	xsi	m	m	.	.	.	m	m
<i>y1()</i>	m	xsi	m	m	.	.	.	m	m
<i>yn()</i>	m	xsi	m	m	.	.	.	m	m

Utilities Interface Table

This chapter lists all the utilities described in XCU, Issue 6, and indicates what other specifications support the interfaces.

The following conventions are used in columns 2 through 8:

- m** Indicates that the interface is defined as mandatory.
- opt* In the POSIX Base column, two or three letter option codes are used as described in Section 3.3 (on page 16), denoting the option to which the interface belongs.
- d** Indicates that the interface is part of the DEVELOPMENT Option or Feature Group.
- f** Indicates that the interface is part of the FORTRAN Option or Feature Group.
- 2d** Indicates that the interface is part of IEEE Std 1003.2d-1994 (Batch Environment).
- o** Indicates that the interface is optional.
- .** Indicates that the interface is not specified.

It should be noted that while another specification may support the interface, some of the interface semantics may have changed with evolution and standardization. A developer should not assume that because the interface appears in other specifications, it will behave exactly as described in the Single UNIX Specification, Version 3.

There are 160 utilities listed.

Interface	XSI	POSIX Base	UNIX 98	UNIX 95	POSIX.2-92	SVID 3	4.3BSD
<i>admin</i>	d	xsi	d	d	.	m	.
<i>alias</i>	m	up	m	m	m	.	m
<i>ar</i>	m	sd	m	m	o	m	m
<i>asa</i>	m	FR	m	m	o	.	.
<i>at</i>	m	up	m	m	m	m	m
<i>awk</i>	m	m	m	m	m	m	m
<i>basename</i>	m	m	m	m	m	m	m
<i>batch</i>	m	up	m	m	m	m	.
<i>bc</i>	m	m	m	m	m	.	m
<i>bg</i>	m	up	m	m	m	.	m
<i>c99</i>	m	cd
<i>cal</i>	m	xsi	m	m	.	m	m
<i>cat</i>	m	m	m	m	m	m	m
<i>cd</i>	m	m	m	m	m	m	m
<i>cflow</i>	d	xsi	d	d	.	m	.
<i>chgrp</i>	m	m	m	m	m	m	m
<i>chmod</i>	m	m	m	m	m	m	m
<i>chown</i>	m	m	m	m	m	m	m
<i>cksum</i>	m	m	m	m	m	.	.
<i>cmp</i>	m	m	m	m	m	m	m
<i>comm</i>	m	m	m	m	m	m	m
<i>command</i>	m	up	m	m	m	.	.
<i>compress</i>	m	xsi	m	m	.	m	m
<i>cp</i>	m	m	m	m	m	m	m
<i>crontab</i>	m	up	m	m	m	m	.
<i>csplit</i>	m	up	m	m	m	m	.
<i>ctags</i>	m	up	d	d	m	m	m
<i>cut</i>	m	m	m	m	m	m	.
<i>cxref</i>	d	xsi	d	d	.	m	.
<i>date</i>	m	m	m	m	m	m	m
<i>dd</i>	m	m	m	m	m	m	m
<i>delta</i>	d	xsi	d	d	.	m	.
<i>df</i>	m	up	m	m	m	m	m
<i>diff</i>	m	m	m	m	m	m	m
<i>dirname</i>	m	m	m	m	m	m	.
<i>du</i>	m	up	l	m	m	m	m
<i>echo</i>	m	m	m	m	m	m	m
<i>ed</i>	m	m	m	m	m	m	m
<i>env</i>	m	m	m	m	m	m	.
<i>ex</i>	m	up	m	m	m	m	m
<i>expand</i>	m	up	m	m	m	.	m
<i>expr</i>	m	m	m	m	m	m	m
<i>false</i>	m	m	m	m	m	m	m
<i>fc</i>	m	up	m	m	m	.	m
<i>fg</i>	m	up	m	m	m	.	m
<i>file</i>	m	up	m	m	m	m	m

Utilities Interface Table

Interface	XSI	POSIX Base	UNIX 98	UNIX 95	POSIX.2-92	SVID 3	4.3BSD
<i>find</i>	m	m	m	m	m	m	m
<i>fold</i>	m	m	m	m	m	.	m
<i>fort77</i>	o	fd	o	f	o	.	.
<i>fuser</i>	m	xsi	m
<i>genscat</i>	m	xsi	m	m	.	.	.
<i>get</i>	d	xsi	d	d	.	m	.
<i>getconf</i>	m	m	m	m	.	.	.
<i>getopts</i>	m	m	m	m	.	.	.
<i>grep</i>	m	m	m	m	m	m	m
<i>hash</i>	m	xsi	m	m	.	.	m
<i>head</i>	m	m	m	m	m	m	m
<i>iconv</i>	m	m	m	m	.	.	.
<i>id</i>	m	m	m	m	m	m	.
<i>ipcrm</i>	m	xsi	m
<i>ipcs</i>	m	xsi	m
<i>jobs</i>	m	up	m	m	m	.	m
<i>join</i>	m	m	m	m	m	m	m
<i>kill</i>	m	m	m	m	m	m	m
<i>lex</i>	d	cd	d	d	o	m	m
<i>link</i>	m	xsi	m	.	.	.	m
<i>ln</i>	m	m	m	m	m	m	m
<i>locale</i>	m	m	m	m	m	.	.
<i>localedef</i>	m	m	m	m	m	.	.
<i>logger</i>	m	m	m	m	m	.	m
<i>logname</i>	m	m	m	m	m	m	.
<i>lp</i>	m	m	m	m	m	m	.
<i>ls</i>	m	xsi	m	m	m	m	m
<i>m4</i>	d	xsi	d	d	.	m	m
<i>mailx</i>	m	m	m	m	m	m	.
<i>make</i>	m	sd	d	d	m	m	m
<i>man</i>	m	m	m	m	m	.	m
<i>mesg</i>	m	up	m	m	m	m	m
<i>mkdir</i>	m	m	m	m	m	m	m
<i>mkfifo</i>	m	m	m	m	m	.	.
<i>more</i>	m	up	m	m	m	m	m
<i>mv</i>	m	m	m	m	m	m	m
<i>newgrp</i>	m	up	m	m	m	m	.
<i>nice</i>	m	up	m	m	m	m	m
<i>nl</i>	m	m	m	m	.	m	.
<i>nm</i>	m	up	d	d	m	m	m
<i>nohup</i>	m	m	m	m	m	m	.
<i>od</i>	m	m	m	m	m	m	m
<i>paste</i>	m	m	m	m	m	m	.
<i>patch</i>	m	up	m	m	m	.	m
<i>pathchk</i>	m	m	m	m	m	.	.
<i>pax</i>	m	m	m	m	m	.	.

Interface	XSI	POSIX Base	UNIX 98	UNIX 95	POSIX.2-92	SVID 3	4.3BSD
<i>pr</i>	m	m	m	m	m	m	m
<i>printf</i>	m	m	m	m	m	m	.
<i>prs</i>	d	xsi	d	d	.	m	.
<i>ps</i>	m	up	m	m	m	m	m
<i>pwd</i>	m	m	m	m	m	m	m
<i>qalter</i>	o	be	.	.	2d	.	
<i>qdel</i>	o	be	.	.	2d	.	
<i>qhold</i>	o	be	.	.	2d	.	
<i>qmove</i>	o	be	.	.	2d	.	
<i>qmsg</i>	o	be	.	.	2d	.	
<i>qrerun</i>	o	be	.	.	2d	.	
<i>qrls</i>	o	be	.	.	2d	.	
<i>qselect</i>	o	be	.	.	2d	.	
<i>qsig</i>	o	be	.	.	2d	.	
<i>qstat</i>	o	be	.	.	2d	.	
<i>qsub</i>	o	be	.	.	2d	.	
<i>read</i>	m	m	m	m	m	.	m
<i>renice</i>	m	up	m	m	m	.	m
<i>rm</i>	m	m	m	m	m	m	m
<i>rmdel</i>	d	xsi	d	d	.	m	.
<i>rmdir</i>	m	m	m	m	m	m	m
<i>sact</i>	d	xsi	d	d	.	m	.
<i>sccs</i>	d	xsi	d	d	.		m
<i>sed</i>	m	m	m	m	m	m	m
<i>sh</i>	m	m	m	m	m	m	m
<i>sleep</i>	m	m	m	m	m	m	m
<i>sort</i>	m	m	m	m	m	m	m
<i>split</i>	m	up	m	m	m	m	m
<i>strings</i>	m	up	m	m	m	.	m
<i>strip</i>	m	sd	d	d	o	m	m
<i>stty</i>	m	m	m	m	m	m	m
<i>tabs</i>	m	up	m	m	m	m	m
<i>tail</i>	m	m	m	m	m	m	m
<i>talk</i>	m	up	m	m	m	.	m
<i>tee</i>	m	m	m	m	.	m	m
<i>test</i>	m	m	m	m	m	m	m
<i>time</i>	m	up	m	m	m	m	m
<i>touch</i>	m	m	m	m	m	m	m
<i>tput</i>	m	up	m	m	m	.	m
<i>tr</i>	m	m	m	m	m	m	m
<i>true</i>	m	m	m	m	m	m	m
<i>tsort</i>	m	xsi	m	m	.	m	m
<i>tty</i>	m	m	m	m	m	m	m
<i>type</i>	m	xsi	m	m	.	.	m
<i>ulimit</i>	m	xsi	m	m	.	.	m
<i>umask</i>	m	m	m	m	m	m	.

Utilities Interface Table

Interface	XSI	POSIX Base	UNIX 98	UNIX 95	POSIX.2-92	SVID 3	4.3BSD
<i>unalias</i>	m	up	m	m	m	.	m
<i>uname</i>	m	m	m	m	m	m	.
<i>uncompress</i>	m	xsi	m	m	.	m	.
<i>unexpand</i>	m	up	m	m	m	.	.
<i>unget</i>	d	xsi	d	d	.	m	.
<i>uniq</i>	m	m	m	m	m	m	m
<i>unlink</i>	m	xsi	m	.	.	.	m
<i>uucp</i>	m	xsi	m	m	.	m	m
<i>uudecode</i>	m	up	m	m	m	m	m
<i>uuencode</i>	m	up	m	m	m	m	m
<i>uustat</i>	m	xsi	m	m	.	m	.
<i>uux</i>	m	xsi	m	m	.	m	m
<i>val</i>	d	xsi	d	d	.	m	.
<i>vi</i>	m	up	m	m	m	m	m
<i>wait</i>	m	m	m	m	m	m	m
<i>wc</i>	m	m	m	m	m	m	m
<i>what</i>	d	xsi	d	d	.	m	m
<i>who</i>	m	up	m	m	m	m	m
<i>write</i>	m	up	m	m	m	m	m
<i>xargs</i>	m	m	m	m	m	m	.
<i>yacc</i>	d	cd	d	d	o	m	m
<i>zcat</i>	m	xsi	m	m	.	m	.

Headers Interface Table

This chapter lists all the headers described in XBD, Issue 6, complete with an indication of their status for the XSI extension, the POSIX Base, and their availability in UNIX 98, UNIX 95, IEEE Std 1003.1-1996 (POSIX.1) (denoted P96), IEEE Std 1003.2-1992 (POSIX.2) (denoted P92), the ISO C standard (denoted C99), and C 89. The following conventions are used in columns 2 through 8:

- m** Indicates that the interface is defined as mandatory.
- o** Indicates that the interface is part of an Option or Feature Group.
- opt* In the POSIX Base column, two or three letter option codes are used as described in Section 3.3 (on page 16), denoting the option to which the interface belongs.
- r** In the P96 column, this indicates that the interface is part of the POSIX Realtime Extension.
- t** In the P96 column, this indicates that the interface is part of the POSIX Threads Extension.
- 1** In the C89 column, this indicates that the interface is part of the ISO/IEC 9899: 1990 standard.
- .** Indicates that the interface is not specified.

Interface	XSI	POSIX Base	UNIX 98	UNIX 95	P96 P92	C99	C89
<aio.h>	o	aio	o	.	r	.	.
<arpa/inet.h>	m	m	m	m	.	.	.
<assert.h>	m	m	m	m	.	m	m
<complex.h>	m	m	.	.	.	m	.
<cpio.h>	m	xsi	m	m	.	.	.
<ctype.h>	m	m	m	m	.	m	m
<dirent.h>	m	m	m	m	m	.	.
<dlfcn.h>	m	xsi	m
<errno.h>	m	m	m	m	m	m	m
<fcntl.h>	m	m	m	m	m	.	.
<fenv.h>	m	m	.	.	.	m	.
<float.h>	m	m	m	m	m	m	m
<fmtmsg.h>	m	xsi	m	m	m	.	.
<fnmatch.h>	m	m	m	m	m	.	.
<ftw.h>	m	xsi	m	m	.	.	.
<glob.h>	m	m	m	m	m	.	.
<grp.h>	m	m	m	m	m	.	.
<iconv.h>	m	xsi	m	m	.	.	.
<inttypes.h>	m	m	m	.	.	m	.
<iso646.h>	m	m	m	.	.	m	1
<langinfo.h>	m	xsi	m	m	.	.	.
<libgen.h>	m	xsi	m	m	.	.	.
<limits.h>	m	m	m	m	m	m	m
<locale.h>	m	m	m	m	m	m	m
<math.h>	m	m	m	m	m	m	m
<monetary.h>	m	xsi	m	m	.	.	.
<mqueue.h>	o	msg	o	.	r	.	.
<ndbm.h>	m	xsi	m	m	.	.	.
<net/if.h>	m	m	m	m	.	.	.
<netdb.h>	m	m	m	m	m	.	.
<netinet/in.h>	m	m	m	m	m	.	.
<netinet/tcp.h>	m	m	m	m	m	.	.
<nl_types.h>	m	xsi	m	m	.	.	.
<poll.h>	m	m	m	m	.	.	.
<pthread.h>	m	thr	m	.	t	.	.
<pwd.h>	m	m	m	m	m	.	.
<regex.h>	m	m	m	m	m	.	.
<sched.h>	o	ps	o	.	r	.	.
<search.h>	m	xsi	m	m	.	.	.
<semaphore.h>	o	sem	o	.	r	.	.
<setjmp.h>	m	m	m	m	m	m	m
<signal.h>	m	m	m	m	m	m	m
<spawn.h>	o	spn
<stdarg.h>	m	m	m	m	m	m	m
<stdbool.h>	m	m	m	m	m	m	.
<stddef.h>	m	m	m	m	m	m	m

Headers Interface Table

Interface	XSI	POSIX Base	UNIX 98	UNIX 95	P96 P92	C99	C89
<stdint.h>	m	m	.	.	.	m	.
<stdio.h>	m	m	m	m	m	m	m
<stdlib.h>	m	m	m	m	m	m	m
<string.h>	m	m	m	m	m	m	m
<strings.h>	m	xsi	m	m	.	.	.
<stropts.h>	m	xsr	m	m	.	.	.
<sys/ipc.h>	m	xsi	m	m	.	.	.
<sys/mman.h>	m	m	m	m	m	.	.
<sys/msg.h>	m	xsi	m	m	.	.	.
<sys/resource.h>	m	xsi	m	m	.	.	.
<sys/select.h>	m	m	m	m	.	.	.
<sys/sem.h>	m	xsi	m	m	m	.	.
<sys/shm.h>	m	xsi	m	m	.	.	.
<sys/socket.h>	m	m	m	m	.	.	.
<sys/stat.h>	m	m	m	m	m	.	.
<sys/statvfs.h>	m	xsi	m	m	m	.	.
<sys/time.h>	m	m	xsi	m	.	.	.
<sys/timeb.h>	m	xsi	m	m	.	.	.
<sys/times.h>	m	m	m	m	m	.	.
<sys/types.h>	m	m	m	m	m	.	.
<sys/uio.h>	m	xsi	m	m	.	.	.
<sys/un.h>	m	xsi	m	m	.	.	.
<sys/utsname.h>	m	m	m	m	m	.	.
<sys/wait.h>	m	m	m	m	m	.	.
<syslog.h>	m	xsi	m	m	.	.	.
<tar.h>	m	m	m	m	m	.	.
<termios.h>	m	m	m	m	m	.	.
<tgmath.h>	m	m	.	.	.	m	.
<time.h>	m	m	m	m	.	m	m
<trace.h>	o	trc
<ucontext.h>	m	xsi	m	m	.	.	.
<ulimit.h>	m	xsi	m	m	.	.	.
<unistd.h>	m	m	m	m	m	.	.
<utime.h>	m	m	m	m	m	.	.
<utmpx.h>	m	xsi	m	m	.	.	.
<wchar.h>	m	m	m	m	.	m	1
<wctype.h>	m	m	m	m	.	m	1
<wordexp.h>	m	m	m	m	m	.	.

XCURSES Interface Table

This chapter contains a table of all the interfaces defined in X/Open Curses, Issue 4, Version 2, complete with an indication of their availability for the Single UNIX Specification (denoted by SUSv3), and their availability in UNIX 98, UNIX 95, POSIX.1, POSIX.2, the ISO C standard, SVID 3, and 4.3BSD. The XCURSES interfaces are not specified for the POSIX base in IEEE Std 1003.1-2001.

The following conventions are used in columns 2 through 8:

m Indicates that the interface is defined as mandatory.

. Indicates that the interface is not specified.

The table is intended as a quick reference guide for programmers migrating to or developing applications for the Single UNIX Specification, Version 3.

There are 372 interfaces listed, and in addition XCURSES defines three headers: **<curses.h>**, **<term.h>**, and **<unctrl.h>**.

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>add_wch()</i>	m	m	m
<i>add_wchnstr()</i>	m	m	m
<i>add_wchstr()</i>	m	m	m
<i>addch()</i>	m	m	m	.	.	m	.
<i>addchnstr()</i>	m	m	m	.	.	m	.
<i>addchstr()</i>	m	m	m	.	.	m	.
<i>addnstr()</i>	m	m	m	.	.	m	.
<i>addnwstr()</i>	m	m	m
<i>addstr()</i>	m	m	m	.	.	m	.
<i>addwstr()</i>	m	m	m
<i>attr_get()</i>	m	m	m
<i>attr_off()</i>	m	m	m
<i>attr_on()</i>	m	m	m
<i>attr_set()</i>	m	m	m
<i>attroff()</i>	m	m	m	.	.	m	.
<i>attron()</i>	m	m	m	.	.	m	.
<i>attrset()</i>	m	m	m	.	.	m	.
<i>baudrate()</i>	m	m	m	.	.	m	.
<i>beep()</i>	m	m	m	.	.	m	.
<i>bkgd()</i>	m	m	m	.	.	m	.
<i>bkgdset()</i>	m	m	m	.	.	m	.

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>bkgrnd()</i>	m	m	m
<i>bkgrndset()</i>	m	m	m
<i>border()</i>	m	m	m	.	.	m	.
<i>border_set()</i>	m	m	m
<i>box()</i>	m	m	m	.	.	m	.
<i>box_set()</i>	m	m	m
<i>can_change_color()</i>	m	m	m	.	.	m	.
<i>chgat()</i>	m	m	m
<i>clear()</i>	m	m	m	.	.	m	.
<i>clearerr()</i>	m	m	m	m	m	m	m
<i>clearok()</i>	m	m	m	.	.	m	.
<i>clrtoebot()</i>	m	m	m	.	.	m	.
<i>clrtoeol()</i>	m	m	m	.	.	m	.
<i>color_content()</i>	m	m	m
<i>color_set()</i>	m	m	m
<i>copywin()</i>	m	m	m	.	.	m	.
<i>curs_set()</i>	m	m	m	.	.	m	.
<i>def_prog_mode()</i>	m	m	m	.	.	m	.
<i>def_shell_mode()</i>	m	m	m	.	.	m	.
<i>del_curterm()</i>	m	m	m	.	.	m	.
<i>delay_output()</i>	m	m	m	.	.	m	.
<i>delch()</i>	m	m	m	.	.	m	.
<i>deleteln()</i>	m	m	m	.	.	m	.
<i>delscreen()</i>	m	m	m	.	.	m	.
<i>delwin()</i>	m	m	m	.	.	m	.
<i>derwin()</i>	m	m	m	.	.	m	.
<i>dupwin()</i>	m	m	m	.	.	m	.
<i>echo()</i>	m	m	m	.	.	m	.
<i>echo_wchar()</i>	m	m	m
<i>echochar()</i>	m	m	m	.	.	m	.
<i>endwin()</i>	m	m	m	.	.	m	.
<i>erase()</i>	m	m	m	.	.	m	.
<i>erasechar()</i>	m	m	m	.	.	m	.
<i>erasewchar()</i>	m	m	m
<i>filter()</i>	m	m	m	.	.	m	.
<i>flash()</i>	m	m	m	.	.	m	.
<i>flushinp()</i>	m	m	m	.	.	m	.
<i>get_wch()</i>	m	m	m
<i>get_wstr()</i>	m	m	m
<i>getbegyx()</i>	m	m	m	.	.	m	.
<i>getbkgd()</i>	m	m	m
<i>getbkgrnd()</i>	m	m	m
<i>getmaxyx()</i>	m	m	m	.	.	m	.
<i>getn_wstr()</i>	m	m	m
<i>getparyx()</i>	m	m	m	.	.	m	.
<i>getstr()</i>	m	m	m	.	.	m	.

XCURSES Interface Table

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>getwin()</i>	m	m	m
<i>getyx()</i>	m	m	m	.	.	m	.
<i>halfdelay()</i>	m	m	m	.	.	m	.
<i>has_colors()</i>	m	m	m	.	.	m	.
<i>has_ic()</i>	m	m	m	.	.	m	.
<i>has_il()</i>	m	m	m	.	.	m	.
<i>hline()</i>	m	m	m
<i>hline_set()</i>	m	m	m
<i>idcok()</i>	m	m	m	.	.	m	.
<i>idlok()</i>	m	m	m	.	.	m	.
<i>ilogb()</i>	m	m	m
<i>immedok()</i>	m	m	m	.	.	m	.
<i>in_wch()</i>	m	m	m
<i>in_wchnstr()</i>	m	m	m
<i>in_wchstr()</i>	m	m	m
<i>inch()</i>	m	m	m	.	.	m	.
<i>inchnstr()</i>	m	m	m	.	.	m	.
<i>inchstr()</i>	m	m	m	.	.	m	.
<i>init_color()</i>	m	m	m	.	.	m	.
<i>init_pair()</i>	m	m	m	.	.	m	.
<i>initscr()</i>	m	m	m	.	.	m	.
<i>innstr()</i>	m	m	m	.	.	m	.
<i>innwstr()</i>	m	m	m
<i>ins_nwstr()</i>	m	m	m
<i>ins_wch()</i>	m	m	m
<i>ins_wstr()</i>	m	m	m
<i>insch()</i>	m	m	m	.	.	m	.
<i>insdelln()</i>	m	m	m	.	.	m	.
<i>insertln()</i>	m	m	m	.	.	m	.
<i>insnstr()</i>	m	m	m	.	.	m	.
<i>insstr()</i>	m	m	m	.	.	m	.
<i>instr()</i>	m	m	m	.	.	m	.
<i>intrflush()</i>	m	m	m	.	.	m	.
<i>inwstr()</i>	m	m	m
<i>is_linetouched()</i>	m	m	m	.	.	m	.
<i>is_wintouched()</i>	m	m	m	.	.	m	.
<i>isendwin()</i>	m	m	m	.	.	m	.
<i>key_name()</i>	m	m	m
<i>keyname()</i>	m	m	m	.	.	m	.
<i>keypad()</i>	m	m	m	.	.	m	.
<i>killchar()</i>	m	m	m
<i>killwchar()</i>	m	m	m
<i>leaveok()</i>	m	m	m	.	.	m	.
<i>longname()</i>	m	m	m	.	.	m	.
<i>meta()</i>	m	m	m	.	.	m	.
<i>move()</i>	m	m	m	.	.	m	.

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>mvadd_wch()</i>	m	m	m
<i>mvadd_wchnstr()</i>	m	m	m
<i>mvadd_wchstr()</i>	m	m	m
<i>mvaddch()</i>	m	m	m	.	.	m	.
<i>mvaddchnstr()</i>	m	m	m	.	.	m	.
<i>mvaddchstr()</i>	m	m	m	.	.	m	.
<i>mvaddnstr()</i>	m	m	m	.	.	m	.
<i>mvaddnwstr()</i>	m	m	m
<i>mvaddstr()</i>	m	m	m	.	.	m	.
<i>mvaddwstr()</i>	m	m	m
<i>mvchgat()</i>	m	m	m
<i>mvcur()</i>	m	m	m	.	.	m	.
<i>mvdelch()</i>	m	m	m	.	.	m	.
<i>mvderwin()</i>	m	m	m	.	.	m	.
<i>mvget_wch()</i>	m	m	m
<i>mvget_wstr()</i>	m	m	m
<i>mvgetch()</i>	m	m	m	.	.	m	.
<i>mvgetn_wstr()</i>	m	m	m
<i>mvgetnstr()</i>	m	m	m	.	.	m	.
<i>mvgetstr()</i>	m	m	m
<i>mvhline()</i>	m	m	m
<i>mvhline_set()</i>	m	m	m
<i>mvin_wch()</i>	m	m	m
<i>mvin_wchnstr()</i>	m	m	m
<i>mvin_wchstr()</i>	m	m	m
<i>mvinch()</i>	m	m	m	.	.	m	.
<i>mvinchnstr()</i>	m	m	m	.	.	m	.
<i>mvinchstr()</i>	m	m	m	.	.	m	.
<i>mvinnstr()</i>	m	m	m	.	.	m	.
<i>mvinnwstr()</i>	m	m	m
<i>mvins_nwstr()</i>	m	m	m
<i>mvins_wch()</i>	m	m	m
<i>mvins_wstr()</i>	m	m	m
<i>mvinsch()</i>	m	m	m	.	.	m	.
<i>mvinsnstr()</i>	m	m	m	.	.	m	.
<i>mvinsstr()</i>	m	m	m	.	.	m	.
<i>mvinstr()</i>	m	m	m	.	.	m	.
<i>mvinwstr()</i>	m	m	m
<i>mvprintw()</i>	m	m	m	.	.	m	.
<i>mvscanw()</i>	m	m	m	.	.	m	.
<i>mvvline()</i>	m	m	m
<i>mvvline_set()</i>	m	m	m
<i>mvwadd_wch()</i>	m	m	m
<i>mvwadd_wchnstr()</i>	m	m	m
<i>mvwadd_wchstr()</i>	m	m	m
<i>mvwaddch()</i>	m	m	m	.	.	m	.

XCURSES Interface Table

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>mvwaddchnstr()</i>	m	m	m	.	.	m	.
<i>mvwaddchstr()</i>	m	m	m	.	.	m	.
<i>mvwaddnstr()</i>	m	m	m	.	.	m	.
<i>mvwaddnwstr()</i>	m	m	m
<i>mvwaddstr()</i>	m	m	m	.	.	m	.
<i>mvwaddwstr()</i>	m	m	m
<i>mvwchgat()</i>	m	m	m
<i>mvwdelch()</i>	m	m	m	.	.	m	.
<i>mvwget_wch()</i>	m	m	m
<i>mvwget_wstr()</i>	m	m	m
<i>mvwgetch()</i>	m	m	m	.	.	m	.
<i>mvwgetn_wstr()</i>	m	m	m
<i>mvwgetnstr()</i>	m	m	m
<i>mvwgetstr()</i>	m	m	m	.	.	m	.
<i>mvwhline()</i>	m	m	m
<i>mvwhline_set()</i>	m	m	m
<i>mvwin()</i>	m	m	m	.	.	m	.
<i>mvwin_wch()</i>	m	m	m
<i>mvwin_wchnstr()</i>	m	m	m
<i>mvwin_wchstr()</i>	m	m	m
<i>mvwinch()</i>	m	m	m	.	.	m	.
<i>mvwinchnstr()</i>	m	m	m	.	.	m	.
<i>mvwinchstr()</i>	m	m	m	.	.	m	.
<i>mvwinnstr()</i>	m	m	m	.	.	m	.
<i>mvwinnwstr()</i>	m	m	m
<i>mvwins_nwstr()</i>	m	m	m
<i>mvwins_wch()</i>	m	m	m
<i>mvwins_wstr()</i>	m	m	m
<i>mvwinsch()</i>	m	m	m	.	.	m	.
<i>mvwinsnstr()</i>	m	m	m	.	.	m	.
<i>mvwinsstr()</i>	m	m	m	.	.	m	.
<i>mvwinstr()</i>	m	m	m	.	.	m	.
<i>mvwinwstr()</i>	m	m	m
<i>mvwprintw()</i>	m	m	m	.	.	m	.
<i>mvwscanw()</i>	m	m	m	.	.	m	.
<i>mvwvline()</i>	m	m	m
<i>mvwvline_set()</i>	m	m	m
<i>napms()</i>	m	m	m	.	.	m	.
<i>newpad()</i>	m	m	m	.	.	m	.
<i>newterm()</i>	m	m	m	.	.	m	.
<i>newwin()</i>	m	m	m	.	.	m	.
<i>nl()</i>	m	m	m	.	.	m	.
<i>nocbreak()</i>	m	m	m	.	.	m	.
<i>nodelay()</i>	m	m	m	.	.	m	.
<i>noecho()</i>	m	m	m	.	.	m	.
<i>nonl()</i>	m	m	m	.	.	m	.

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>noqiflush()</i>	m	m	m	.	.	m	.
<i>noraw()</i>	m	m	m	.	.	m	.
<i>notimeout()</i>	m	m	m	.	.	m	.
<i>overlay()</i>	m	m	m	.	.	m	.
<i>overwrite()</i>	m	m	m	.	.	m	.
<i>pair_content()</i>	m	m	m	.	.	m	.
<i>pecho_wchar()</i>	m	m	m
<i>pechochar()</i>	m	m	m	.	.	m	.
<i>pnoutrefresh()</i>	m	m	m	.	.	m	.
<i>prefresh()</i>	m	m	m	.	.	m	.
<i>printw()</i>	m	m	m	.	.	m	.
<i>putp()</i>	m	m	m	.	.	m	.
<i>putwin()</i>	m	m	m	.	.	m	.
<i>qiflush()</i>	m	m	m	.	.	m	.
<i>raw()</i>	m	m	m	.	.	m	.
<i>redrawwin()</i>	m	m	m	.	.	m	.
<i>refresh()</i>	m	m	m	.	.	m	.
<i>reset_prog_mode()</i>	m	m	m	.	.	m	.
<i>reset_shell_mode()</i>	m	m	m	.	.	m	.
<i>resetty()</i>	m	m	m	.	.	m	.
<i>restartterm()</i>	m	m	m	.	.	m	.
<i>ripoffline()</i>	m	m	m	.	.	m	.
<i>savetty()</i>	m	m	m	.	.	m	.
<i>scr_dump()</i>	m	m	m	.	.	m	.
<i>scr_init()</i>	m	m	m	.	.	m	.
<i>scr_restore()</i>	m	m	m	.	.	m	.
<i>scr_set()</i>	m	m	m	.	.	m	.
<i>scr1()</i>	m	m	m	.	.	m	.
<i>scroll()</i>	m	m	m	.	.	m	.
<i>scrollok()</i>	m	m	m	.	.	m	.
<i>set_curterm()</i>	m	m	m	.	.	m	.
<i>set_term()</i>	m	m	m	.	.	m	.
<i>setcchar()</i>	m	m	m
<i>setscrreg()</i>	m	m	m	.	.	m	.
<i>setupterm()</i>	m	m	m	.	.	m	.
<i>slk_attr_off()</i>	m	m	m
<i>slk_attr_on()</i>	m	m	m
<i>slk_attr_set()</i>	m	m	m
<i>slk_attroff()</i>	m	m	m	.	.	m	.
<i>slk_attron()</i>	m	m	m	.	.	m	.
<i>slk_attrset()</i>	m	m	m	.	.	m	.
<i>slk_clear()</i>	m	m	m	.	.	m	.
<i>slk_color()</i>	m	m	m
<i>slk_init()</i>	m	m	m	.	.	m	.
<i>slk_label()</i>	m	m	m	.	.	m	.
<i>slk_noutrefresh()</i>	m	m	m	.	.	m	.

XCURSES Interface Table

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>slk_refresh()</i>	m	m	m	.	.	m	.
<i>slk_restore()</i>	m	m	m	.	.	m	.
<i>slk_set()</i>	m	m	m	.	.	m	.
<i>slk_touch()</i>	m	m	m	.	.	m	.
<i>slk_wset()</i>	m	m	m
<i>standend()</i>	m	m	m	.	.	m	.
<i>standout()</i>	m	m	m	.	.	m	.
<i>start_color()</i>	m	m	m	.	.	m	.
<i>stdscr()</i>	m	m	m	.	.	m	.
<i>subpad()</i>	m	m	m	.	.	m	.
<i>subwin()</i>	m	m	m	.	.	m	.
<i>syncok()</i>	m	m	m	.	.	m	.
<i>term_attrs()</i>	m	m	m
<i>termattrs()</i>	m	m	m	.	.	m	.
<i>termname()</i>	m	m	m	.	.	m	.
<i>tgetent()</i>	m	m	m	.	.	m	.
<i>tgetflag()</i>	m	m	m	.	.	m	.
<i>tgetnum()</i>	m	m	m	.	.	m	.
<i>tgetstr()</i>	m	m	m	.	.	m	.
<i>tgoto()</i>	m	m	m	.	.	m	.
<i>tigetflag()</i>	m	m	m	.	.	m	.
<i>tigetnum()</i>	m	m	m	.	.	m	.
<i>tigetstr()</i>	m	m	m	.	.	m	.
<i>timeout()</i>	m	m	m	.	.	m	.
<i>touchline()</i>	m	m	m	.	.	m	.
<i>touchwin()</i>	m	m	m	.	.	m	.
<i>tparm()</i>	m	m	m	.	.	m	.
<i>tputs()</i>	m	m	m	.	.	m	.
<i>typeahead()</i>	m	m	m	.	.	m	.
<i>unctrl()</i>	m	m	m	.	.	m	.
<i>unget_wch()</i>	m	m	m
<i>ungetch()</i>	m	m	m	.	.	m	.
<i>untouchwin()</i>	m	m	m	.	.	m	.
<i>use_env()</i>	m	m	m	.	.	m	.
<i>vid_attr()</i>	m	m	m
<i>vid_puts()</i>	m	m	m
<i>vidattr()</i>	m	m	m	.	.	m	.
<i>vidputs()</i>	m	m	m	.	.	m	.
<i>vline()</i>	m	m	m
<i>vline_set()</i>	m	m	m
<i>vw_printw()</i>	m	m	m
<i>vw_scanw()</i>	m	m	m
<i>vwprintw()</i>	m	m	m	.	.	m	.
<i>wscanw()</i>	m	m	m	.	.	m	.
<i>wadd_wch()</i>	m	m	m
<i>wadd_wchnstr()</i>	m	m	m

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>wadd_wchstr()</i>	m	m	m
<i>waddch()</i>	m	m	m	.	.	m	.
<i>waddchnstr()</i>	m	m	m	.	.	m	.
<i>waddchstr()</i>	m	m	m	.	.	m	.
<i>waddnstr()</i>	m	m	m	.	.	m	.
<i>waddnwstr()</i>	m	m	m
<i>waddstr()</i>	m	m	m	.	.	m	.
<i>waddwstr()</i>	m	m	m
<i>wattr_get()</i>	m	m	m
<i>wattr_off()</i>	m	m	m
<i>wattr_on()</i>	m	m	m
<i>wattr_set()</i>	m	m	m
<i>wattroff()</i>	m	m	m	.	.	m	.
<i>wattron()</i>	m	m	m	.	.	m	.
<i>wattrset()</i>	m	m	m	.	.	m	.
<i>wbkgd()</i>	m	m	m	.	.	m	.
<i>wbkgdset()</i>	m	m	m	.	.	m	.
<i>wbkgrnd()</i>	m	m	m
<i>wbkgrndset()</i>	m	m	m
<i>wborder()</i>	m	m	m	.	.	m	.
<i>wborder_set()</i>	m	m	m
<i>wchgat()</i>	m	m	m
<i>wclear()</i>	m	m	m	.	.	m	.
<i>wclrtoebot()</i>	m	m	m	.	.	m	.
<i>wclrtoeol()</i>	m	m	m	.	.	m	.
<i>wcolor_set()</i>	m	m	m
<i>wcursyncup()</i>	m	m	m	.	.	m	.
<i>wdelch()</i>	m	m	m	.	.	m	.
<i>wdeleteln()</i>	m	m	m	.	.	m	.
<i>wecho_wchar()</i>	m	m	m
<i>wechochar()</i>	m	m	m	.	.	m	.
<i>werase()</i>	m	m	m	.	.	m	.
<i>wget_wch()</i>	m	m	m
<i>wget_wstr()</i>	m	m	m
<i>wgetbkgrnd()</i>	m	m	m
<i>wgetch()</i>	m	m	m	.	.	m	.
<i>wgetn_wstr()</i>	m	m	m
<i>wgetnstr()</i>	m	m	m	.	.	m	.
<i>wgetstr()</i>	m	m	m	.	.	m	.
<i>whline()</i>	m	m	m	.	.	m	.
<i>whline_set()</i>	m	m	m
<i>win_wch()</i>	m	m	m
<i>win_wchnstr()</i>	m	m	m
<i>win_wchstr()</i>	m	m	m
<i>winch()</i>	m	m	m	.	.	m	.
<i>winchnstr()</i>	m	m	m	.	.	m	.

XCURSES Interface Table

Interface	SUSv3	UNIX 98	UNIX 95	POSIX.1/ POSIX.2	ISO C Standard	SVID 3	4.3BSD
<i>winchstr()</i>	m	m	m	.	.	m	.
<i>winnstr()</i>	m	m	m	.	.	m	.
<i>winnwstr()</i>	m	m	m
<i>wins_nwstr()</i>	m	m	m
<i>wins_wch()</i>	m	m	m
<i>wins_wstr()</i>	m	m	m
<i>winsch()</i>	m	m	m	.	.	m	.
<i>winsdelln()</i>	m	m	m	.	.	m	.
<i>winsertln()</i>	m	m	m	.	.	m	.
<i>winsnstr()</i>	m	m	m	.	.	m	.
<i>winsstr()</i>	m	m	m	.	.	m	.
<i>winstr()</i>	m	m	m	.	.	m	.
<i>winwstr()</i>	m	m	m
<i>wmove()</i>	m	m	m	.	.	m	.
<i>wnoutrefresh()</i>	m	m	m	.	.	m	.
<i>wprintw()</i>	m	m	m	.	.	m	.
<i>wredrawln()</i>	m	m	m	.	.	m	.
<i>wrefresh()</i>	m	m	m	.	.	m	.
<i>wscr1()</i>	m	m	m	.	.	m	.
<i>wsetscrreg()</i>	m	m	m	.	.	m	.
<i>wstandend()</i>	m	m	m	.	.	m	.
<i>wstandout()</i>	m	m	m	.	.	m	.
<i>wsyncdown()</i>	m	m	m	.	.	m	.
<i>wsyncup()</i>	m	m	m	.	.	m	.
<i>wtimeout()</i>	m	m	m	.	.	m	.
<i>wtouchln()</i>	m	m	m	.	.	m	.
<i>wunctrl()</i>	m	m	m
<i>wvline()</i>	m	m	m	.	.	m	.
<i>wvline_set()</i>	m	m	m

System Interfaces Migration

This chapter contains a section for each system interface defined in XSH, Issue 6 (including Technical Corrigendum 1 and Technical Corrigendum 2). Each section contains the SYNOPSIS and identifies syntax and semantic changes made to the interface in Issue 6 (if any), complete with examples where appropriate. Only changes that might affect an application programmer are identified. It should be noted that the following changes made in Issue 6 are not further discussed in this chapter.

ISO/IEC Terminology Changes

In order for the XSH document to also be an ISO/IEC standard, a number of terminology changes in the normative text were made, notably use of the term “shall” instead of the term “will” for requirements, and avoiding the term “must” for application requirements. Although these terminology changes affected most interfaces, no functional change is intended and these changes are not further noted in this chapter.

ISO C Extension Markings

Extensions beyond the ISO C standard have been marked in many interfaces. This explicitly shows where the ISO C standard has been extended within the text. No functional changes have been intended by this change, and this change is not further noted in this chapter.

_longjmp, _setjmp

Non-local goto

```
xSI #include <setjmp.h>
void _longjmp(jmp_buf env, int val);
int _setjmp(jmp_buf env);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

_tolower

Transliterate uppercase characters to lowercase

```
xSI #include <ctype.h>
int _tolower(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

_toupper

Transliterate lowercase characters to uppercase

```
xSI #include <ctype.h>
int _toupper(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

a64l, l64a

Convert between a 32-bit integer and a radix-64 ASCII string

```
xSI #include <stdlib.h>
long a64l(const char *s);
char *l64a(long value);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

abort

Generate an abnormal process abort

```
#include <stdlib.h>

void abort(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/10 is applied, changing the DESCRIPTION of abnormal termination processing.

The ISO/IEC 9899:1999 standard requires the *abort()* function to be async-signal-safe. Since IEEE Std 1003.1-2001 defers to the ISO C standard, this required a change to the DESCRIPTION from “shall include the effect of *fclose()*” to “may include an attempt to effect *fclose()*.”

The revised wording permits some backwards-compatibility and avoids a potential deadlock situation.

The Open Group Base Resolution bwg2002-003 is applied, removing the following XSI shaded paragraph from the DESCRIPTION:

“On XSI-conformant systems, in addition the abnormal termination processing shall include the effect of *fclose()* on message catalog descriptors.”

There were several reasons to remove this paragraph:

- No special processing of open message catalogs needs to be performed prior to abnormal process termination.
- The main reason to specifically mention that *abort()* includes the effect of *fclose()* on open streams is to flush output queued on the stream. Message catalogs in this context are read-only and, therefore, do not need to be flushed.
- The effect of *fclose()* on a message catalog descriptor is unspecified. Message catalog descriptors are allowed, but not required to be implemented using a file descriptor, but there is no mention in IEEE Std 1003.1-2001 of a message catalog descriptor using a standard I/O stream FILE object as would be expected by *fclose()*.

abs

Return an integer absolute value

```
#include <stdlib.h>

int abs(int i);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

accept

Accept a new connection on a socket

```
#include <sys/socket.h>

int accept(int socket, struct sockaddr *restrict address,
           socklen_t *restrict address_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The following change is made over the XNS, Issue 5.2 specification:

- The **restrict** keyword is added to the *accept()* prototype for alignment with the ISO/IEC 9899:1999 standard.

access

Determine accessibility of a file

```
#include <unistd.h>

int access(const char *path, int amode);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

acos, acosf, acosl

Arc cosine functions

```
#include <math.h>

double acos(double x);
float acosf(float x);
long double acosl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *acosf()* and *acosl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

acosh, acoshf, acoshl

Inverse hyperbolic cosine functions

```
#include <math.h>

double acosh(double x);
float acoshf(float x);
long double acoshl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *acosh()* function is no longer marked as an extension.

The *acoshf()* and *acoshl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

aio_cancel

Cancel an asynchronous I/O request (**REALTIME**)

```
AIO #include <aio.h>

int aio_cancel(int fildes, struct aiocb *aiocbp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/10 is applied, removing the words “to the calling process” in the RETURN VALUE section. The term was unnecessary and precluded threads.

aio_error

Retrieve errors status for an asynchronous I/O operation (**REALTIME**)

```
AIO #include <aio.h>

int aio_error(const struct aiocb *aiocbp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

aio_fsyncAsynchronous file synchronization (**REALTIME**)

```
AIO #include <aio.h>
int aio_fsync(int op, struct aiocb *aiocbp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/11 is applied, removing the words “to the calling process” in the RETURN VALUE section. The term was unnecessary and precluded threads.

aio_readAsynchronous read from a file (**REALTIME**)

```
AIO #include <aio.h>
int aio_read(struct aiocb *aiocbp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/12 is applied, rewording the DESCRIPTION when prioritized I/O is supported to account for threads, and removing the words “to the calling process” in the RETURN VALUE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/13 is applied, updating the [EINVAL] error, so that detection of an [EINVAL] error for an invalid value of *aiocbp->aio_reqprio* is only required if the Prioritized Input and Output option is supported.

aio_return

Retrieve return status of an asynchronous I/O operation (**REALTIME**)

```
AIO #include <aio.h>
    ssize_t aio_return(struct aiocb *aiocbp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.
The [EINVAL] error condition is updated as a “may fail”. This is for consistency with the DESCRIPTION.

aio_suspend

Wait for an asynchronous I/O request (**REALTIME**)

```
AIO #include <aio.h>
    int aio_suspend(const struct aiocb *const list[], int nent,
        const struct timespec *timeout);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.
The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that the CLOCK_MONOTONIC clock, if supported, is used.

aio_write

Asynchronous write to a file (**REALTIME**)

```
AIO #include <aio.h>
    int aio_write(struct aiocb *aiocbp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.
The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate that for regular files no data transfer occurs past the offset maximum established in the open file description associated with *aiocbp->aio_fildes*.

- The [EFBIG] error is added as part of the large file support extensions.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/14 is applied, rewording the DESCRIPTION when prioritized I/O is supported to account for threads, and removing the words “to the calling process” in the RETURN VALUE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/15 is applied, updating the [EINVAL] error, so that detection of an [EINVAL] error for an invalid value of *aiocbp->ai_reqprio* is only required if the Prioritized Input and Output option is supported.

alarm

Schedule an alarm signal

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

```
unsigned alarm(unsigned seconds);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to indicate that interactions with the *setitimer()*, *ualarm()*, and *usleep()* functions are unspecified.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/16 is applied, replacing “an implementation-defined maximum value” with “an implementation-specific maximum value” in the RATIONALE.

asctime, asctime_r

Convert date and time to a string

```
#include <time.h>
```

```
char *asctime(const struct tm *timeptr);
```

```
TSF char *asctime_r(const struct tm *restrict tm, char *restrict buf);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *asctime_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The DESCRIPTION of *asctime_r()* is updated to describe the format of the string returned.

The **restrict** keyword is added to the *asctime_r()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/17 is applied, adding the CX extension in the RETURN VALUE section requiring that if the *asctime()* function is unsuccessful it returns NULL.

asin, asinf, asinl

Arc sine function

```
#include <math.h>

double asin(double x);
float asinf(float x);
long double asinl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *asinf()* and *asinl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

asinh, asinhf, asinhl

Inverse hyperbolic sine functions

```
#include <math.h>

double asinh(double x);
float asinhf(float x);
long double asinhl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *asinh()* function is no longer marked as an extension.

The *asinhf()* and *asinhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

assert

Insert program diagnostics

```
#include <assert.h>

void assert(scalar expression);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The prototype for the *expression* argument to *assert()* is changed from **int** to **scalar** for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION of *assert()* is updated for alignment with the ISO/IEC 9899:1999 standard.

atan, atanf, atanl

Arc tangent function

```
#include <math.h>

double atan(double x);
float atanf(float x);
long double atanl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *atanf()* and *atanl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

atan2, atan2f, atan2l

Arc tangent functions

```
#include <math.h>

double atan2(double y, double x);
float atan2f(float y, float x);
long double atan2l(long double y, long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *atan2f()* and *atan2l()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

atanh, atanhf, atanh1

Inverse hyperbolic tangent functions

```
#include <math.h>

double atanh(double x);
float atanhf(float x);
long double atanh1(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *atanh()* function is no longer marked as an extension.

The *atanhf()* and *atanh1()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

atexit

Register a function to run at process termination

```
#include <stdlib.h>

int atexit(void (*func)(void));
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

atof

Convert a string to double-precision number

```
#include <stdlib.h>

double atof(const char *str);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

atoi

Convert a string to an integer

```
#include <stdlib.h>

int atoi(const char *str);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

atol, atoll

Convert a string to a long integer

```
#include <stdlib.h>

long atol(const char *str);
long long atoll(const char *nptr);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *atoll()* function is added for alignment with the ISO/IEC 9899:1999 standard.

basename

Return the last component of a pathname

XSI `#include <libgen.h>`

`char *basename(char *path);`

Derivation First released in Issue 4, Version 2.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/20 is applied, changing the DESCRIPTION to make it clear that the string referenced is the string pointed to by *path*.

bcmp

Memory operations (**LEGACY**)

XSI `#include <strings.h>`

`int bcmp(const void *s1, const void *s2, size_t n);`

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations.

The *memcmp()* function is preferred over this function.

For maximum portability, it is recommended to replace the function call to *bcmp()* as follows:

```
#define bcmp(b1,b2,len) memcmp((b1), (b2), (size_t)(len))
```

bcopy

Memory operations (**LEGACY**)

XSI `#include <strings.h>`

`void bcopy(const void *s1, void *s2, size_t n);`

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations.

The *memmove()* function is preferred over this function.

The following are approximately equivalent (note the order of the arguments):

```
bcopy(s1,s2,n) ~ memmove(s2,s1,n)
```

For maximum portability, it is recommended to replace the function call to *bcopy()* as follows:

```
#define bcopy(b1,b2,len) (memmove((b2), (b1), (len)), (void) 0)
```

bind

Bind a name to a socket

```
#include <sys/socket.h>

int bind(int socket, const struct sockaddr *address,
         socklen_t address_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes from the XNS, Issue 5.2 specification.

bsd_signal

Simplified signal facilities

```
OB XSI #include <signal.h>

void (*bsd_signal(int sig, void (*func)(int)))(int);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked obsolescent and is recommended not for use in new applications.

This function is a direct replacement for the BSD *signal()* function for simple applications that are installing a single-argument signal handler function. If a BSD signal handler function is being installed that expects more than one argument, the application has to be modified to use *sigaction()*. The *bsd_signal()* function differs from *signal()* in that the SA_RESTART flag is set and the SA_RESETHAND is clear when *bsd_signal()* is used. The state of these flags is not specified for *signal()*.

It is recommended that new applications use the *sigaction()* function.

bsearch

Binary search a sorted table

```
#include <stdlib.h>

void *bsearch(const void *key, const void *base, size_t nel,
             size_t width, int (*compar)(const void *, const void *));
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/11 is applied, adding to the DESCRIPTION the last sentence of the first non-shaded paragraph, and the following three paragraphs. These changes are for alignment with the ISO C standard.

btowc

Single byte to wide character conversion

```
#include <stdio.h>
#include <wchar.h>

wint_t btowc(int c);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

bzero

Memory operations (**LEGACY**)

```
xsl #include <strings.h>
void bzero(void *s, size_t n);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations. The *memset()* function is preferred over this function.

For maximum portability, it is recommended to replace the function call to *bzero()* as follows:

```
#define bzero(b,len) (memset((b), '\0', (len)), (void) 0)
```

cabs, cabsf, cabsl

Return a complex absolute value

```
#include <complex.h>

double cabs(double complex z);
float cabsf(float complex z);
long double cabsl(long double complex z);
```

These functions compute the complex absolute value (also called norm, modulus, or magnitude) of *z*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

cacos, cacosf, cacosl

Complex arc cosine functions

```
#include <complex.h>

double complex cacos(double complex z);
float complex cacosf(float complex z);
long double complex cacosl(long double complex z);
```

These functions compute the complex arc cosine of z , with branch cuts outside the interval $[-1, +1]$ along the real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

cacosh, cacoshf, cacoshl

Complex arc hyperbolic cosine functions

```
#include <complex.h>
```

```
double complex cacosh(double complex z);  
float complex cacoshf(float complex z);  
long double complex cacoshl(long double complex z);
```

These functions compute the complex arc hyperbolic cosine of z , with a branch cut at values less than 1 along the real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

calloc

A memory allocator

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

```
void *calloc(size_t nelem, size_t elsize);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The setting of *errno* and the [ENOMEM] error condition are mandatory if an insufficient memory condition occurs.

carg, cargf, cargl

Complex argument functions

```
#include <complex.h>
```

```
double carg(double complex z);  
float cargf(float complex z);  
long double cargl(long double complex z);
```

These functions compute the argument (also called phase angle) of z , with a branch cut along the negative real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

casin, casinf, casinl

Complex arc sine functions

```
#include <complex.h>

double complex casin(double complex z);
float complex casinf(float complex z);
long double complex casinl(long double complex z);
```

These functions compute the complex arc sine of z , with branch cuts outside the interval $[-1, +1]$ along the real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

casinh, casinhf, casinhl

Complex arc hyperbolic sine functions

```
#include <complex.h>

double complex casinh(double complex z);
float complex casinhf(float complex z);
long double complex casinhl(long double complex z);
```

These functions compute the complex arc hyperbolic sine of z , with branch cuts outside the interval $[-i, +i]$ along the imaginary axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

catan, catanf, catanl

Complex arc tangent functions

```
#include <complex.h>

double complex catan(double complex z);
float complex catanf(float complex z);
long double complex catanl(long double complex z);
```

These functions compute the complex arc tangent of z , with branch cuts outside the interval $[-i, +i]$ along the imaginary axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

catanh, catanhf, catanh1

Complex arc hyperbolic tangent functions

```
#include <complex.h>
```

```
double complex catanh(double complex z);  
float complex catanhf(float complex z);  
long double complex catanh1(long double complex z);
```

These functions compute the complex arc hyperbolic tangent of z , with branch cuts outside the interval $[-1, +1]$ along the real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

catclose

Close a message catalog descriptor

```
xSI #include <nl_types.h>  
int catclose(nl_catd catd);
```

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

catgets

Read a program message

```
xSI #include <nl_types.h>  
char *catgets(nl_catd catd, int set_id, int msg_id, const char *s);
```

Derivation First released in Issue 2.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

catopen

Open a message catalog

```
xSI #include <nl_types.h>  
nl_catd catopen(const char *name, int oflag);
```

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

cbrt, cbrtf, cbrtl

Cube root functions

```
#include <math.h>

double cbrt(double x);
float cbrtf(float x);
long double cbrtl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *cbrt()* function is no longer marked as an extension.

The *cbrtf()* and *cbrtl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

ccos, ccosf, ccosl

Complex cosine functions

```
#include <complex.h>

double complex ccos(double complex z);
float complex ccosf(float complex z);
long double complex ccosl(long double complex z);
```

These functions compute the complex cosine of *z*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

ccosh, ccoshf, ccoshl

Complex hyperbolic cosine functions

```
#include <complex.h>

double complex ccosh(double complex z);
float complex ccoshf(float complex z);
long double complex ccoshl(long double complex z);
```

These functions compute the complex hyperbolic cosine of *z*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

ceil, ceilf, ceill

Ceiling value function

```
#include <math.h>

double ceil(double x);
float ceilf(float x);
long double ceill(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `ceilf()` and `ceilll()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

cexp, cexpf, cexpl

Complex exponential functions

```
#include <complex.h>
```

```
double complex cexp(double complex z);  
float complex cexpf(float complex z);  
long double complex cexpl(long double complex z);
```

These functions compute the complex exponent of z , defined as e^z .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

cfgetispeed

Get input baud rate

```
#include <termios.h>
```

```
speed_t cfgetispeed(const struct termios *termios_p);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 No functional changes are made in this issue.

cfgetospeed

Get output baud rate

```
#include <termios.h>
```

```
speed_t cfgetospeed(const struct termios *termios_p);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 No functional changes are made in this issue.

cfsetispeed

Set input baud rate

```
#include <termios.h>
```

```
int cfsetispeed(struct termios *termios_p, speed_t speed);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional setting of *errno* and the [EINVAL] error conditions are added.

cfsetospeed

Set output baud rate

```
#include <termios.h>
```

```
int cfsetospeed(struct termios *termios_p, speed_t speed);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional setting of *errno* and the [EINVAL] error conditions are added.

chdir

Change working directory

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

```
int chdir(const char *path);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

chmod

Change mode of a file

```
#include <sys/stat.h>
```

```
int chmod(const char *path, mode_t mode);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] and [EINTR] optional error conditions are added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

chown

Change owner and group of a file

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

```
int chown(const char *path, uid_t owner, gid_t group);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The wording describing the optional dependency on `_POSIX_CHOWN_RESTRICTED` is restored.
- The `[EPERM]` error is restored as an error dependent on `_POSIX_CHOWN_RESTRICTED`. This is since its operand is a pathname and applications should be aware that the error may not occur for that pathname if the file system does not support `_POSIX_CHOWN_RESTRICTED`.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The value for *owner* of `(uid_t)-1` allows the use of `-1` by the owner of a file to change the group ID only. A corresponding change is made for group.
- The `[ELOOP]` mandatory error condition is added.
- The `[EIO]` and `[EINTR]` optional error conditions are added.
- A second `[ENAMETOOLONG]` is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that the `S_ISUID` and `S_ISGID` bits do not need to be cleared when the process has appropriate privileges.
- The `[ELOOP]` optional error condition is added.

cimag, cimagf, cimagl

Complex imaginary functions

```
#include <complex.h>
```

```
double cimag(double complex z);
```

```
float cimagf(float complex z);
```

```
long double cimagl(long double complex z);
```

These functions compute the imaginary part of *z*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

clearerr

Clear indicators on a stream

```
#include <stdio.h>
void clearerr(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

clock

Report CPU time used

```
#include <time.h>
clock_t clock(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

clock_getcpuclockid

Access a process CPU-time clock (**ADVANCED REALTIME**)

```
CPT #include <time.h>
int clock_getcpuclockid(pid_t pid, clockid_t *clock_id);
```

The *clock_getcpuclockid()* function returns the clock ID of the CPU-time clock of the process specified by *pid*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

Issue 6 As a change over IEEE Std 1003.1d-1999, in the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

clock_getres, clock_gettime, clock_settime

Clock and timer functions (**REALTIME**)

```
TMR #include <time.h>
int clock_getres(clockid_t clock_id, struct timespec *res);
int clock_gettime(clockid_t clock_id, struct timespec *tp);
int clock_settime(clockid_t clock_id, const struct timespec *tp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. These functions are part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added of the effect of resetting the clock resolution.

CPU-time clocks and the `clock_getcpuclockid()` function are added for alignment with IEEE Std 1003.1d-1999.

The following changes are added for alignment with IEEE Std 1003.1j-2000:

- The DESCRIPTION is updated as follows:
 - The value returned by `clock_gettime()` for CLOCK_MONOTONIC is specified.
 - `clock_settime()` failing for CLOCK_MONOTONIC is specified.
 - The effects of `clock_settime()` on the `clock_nanosleep()` function with respect to CLOCK_REALTIME are specified.
- An [EINVAL] error is added to the ERRORS section, indicating that `clock_settime()` fails for CLOCK_MONOTONIC.
- The APPLICATION USAGE section notes that the CLOCK_MONOTONIC clock need not and shall not be set by `clock_settime()` since the absolute value of the CLOCK_MONOTONIC clock is meaningless.

clock_nanosleep

High resolution sleep with specifiable clock (**ADVANCED REALTIME**)

```
cs #include <time.h>
int clock_nanosleep(clockid_t clock_id, int flags,
    const struct timespec *rqtp, struct timespec *rmtp);
```

The `clock_nanosleep()` function causes the current thread to be suspended from execution until either an interval timer or an absolute timer has elapsed.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

Issue 6 This is a new function included for alignment with the IEEE Std 1003.1j-2000.

clog, clogf, clogl

Complex natural logarithm functions

```
#include <complex.h>
double complex clog(double complex z);
float complex clogf(float complex z);
long double complex clogl(long double complex z);
```

These functions compute the complex natural (base e) logarithm of z , with a branch cut along the negative real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

close

Close a file descriptor

```
#include <unistd.h>
int close(int fd);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The DESCRIPTION related to a STREAMS-based file or pseudo-terminal is marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] error condition is added as an optional error.
- The DESCRIPTION is updated to describe the state of the *fd* file descriptor as unspecified if an I/O error occurs and an [EIO] error condition is returned.

Text referring to sockets is added to the DESCRIPTION.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that shared memory objects and memory mapped files (and not typed memory objects) are the types of memory objects to which the paragraph on last closes applies.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/12 is applied, correcting the XSH shaded text relating to the master side of a pseudo-terminal. The reason for the change is that the behavior of pseudo-terminals and regular terminals should be as much alike as possible in this case; the change achieves that and matches historical behavior.

closedir

Close a directory stream

```
#include <dirent.h>
int closedir(DIR *dirp);
```

Derivation First released in Issue 2.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINTR] error condition is added as an optional error condition.

closelog, openlog, setlogmask, syslog

Control system log

```
xsi #include <syslog.h>
void closelog(void);
void openlog(const char *ident, int logopt, int facility);
int setlogmask(int maskpri);
void syslog(int priority, const char *message, ... /* arguments */);
```


Derivation First released in Issue 4, Version 2.
Issue 6 No functional changes are made in this issue.

confstr

Get configurable variables

```
#include <unistd.h>

size_t confstr(int name, char *buf, size_t len);
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 The Open Group Corrigendum U033/7 is applied. The return value for the case returning the size of the buffer now explicitly states that this includes the terminating null.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated with new arguments which can be used to determine configuration strings for C compiler flags, linker/loader flags, and libraries for each different supported programming environment. This is a change to support data size neutrality.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION is updated to include text describing how `_CS_PATH` can be used to obtain a *PATH* to access the standard utilities.

The macros associated with the *c89* programming models are marked LEGACY and new equivalent macros associated with *c99* are introduced.

conj, conjf, conjl

Complex conjugate functions

```
#include <complex.h>

double complex conj(double complex z);
float complex conjf(float complex z);
long double complex conjl(long double complex z);
```

These functions compute the complex conjugate of *z*, by reversing the sign of its imaginary part.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

connect

Connect a socket

```
#include <sys/socket.h>

int connect(int socket, const struct sockaddr *address,
            socklen_t address_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

copysign, copysignf, copysignl

Number manipulation function

```
#include <math.h>

double copysign(double x, double y);
float copysignf(float x, float y);
long double copysignl(long double x, long double y);
```

These functions produce a value with the magnitude of *x* and the sign of *y*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

cos, cosf, cosl

Cosine function

```
#include <math.h>

double cos(double x);
float cosf(float x);
long double cosl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *cosf()* and *cosl()* functions are added for alignment with the ISO/IEC 9899: 1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899: 1999 standard.

IEC 60559: 1989 standard floating-point extensions over the ISO/IEC 9899: 1999 standard are marked.

cosh, coshf, coshl

Hyperbolic cosine functions

```
#include <math.h>

double cosh(double x);
float coshf(float x);
long double coshl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *coshf()* and *coshl()* functions are added for alignment with the ISO/IEC 9899: 1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899: 1999 standard.

IEC 60559: 1989 standard floating-point extensions over the ISO/IEC 9899: 1999 standard are marked.

cpow, cpowf, cpowl

Complex power functions

```
#include <complex.h>

double complex cpow(double complex x, double complex y);
float complex cpowf(float complex x, float complex y);
long double complex cpowl(long double complex x,
    long double complex y);
```

These functions compute the complex power function x^y , with a branch cut for the first parameter along the negative real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

cproj, cprojf, cprojl

Complex projection functions

```
#include <complex.h>

double complex cproj(double complex z);
float complex cprojf(float complex z);
long double complex cprojl(long double complex z);
```

These functions compute a projection of z onto the Riemann sphere.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

creal, crealf, creall

Complex real functions

```
#include <complex.h>

double creal(double complex z);
float crealf(float complex z);
long double creall(long double complex z);
```

These functions compute the real part of z .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

creat

Create a new file or rewrite an existing one

```
OH #include <sys/stat.h>
#include <fcntl.h>

int creat(const char *path, mode_t mode);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

crypt

String encoding function (**CRYPT**)

```
XSI #include <unistd.h>

char *crypt(const char *key, const char *salt);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

csin, csinf, csinl

Complex sine functions

```
#include <complex.h>

double complex csin(double complex z);
float complex csinf(float complex z);
long double complex csinl(long double complex z);
```

These functions compute the complex sine of z .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

csinh, csinhf, csinhl

Complex hyperbolic sine functions

```
#include <complex.h>

double complex csinh(double complex z);
float complex csinhf(float complex z);
long double complex csinhl(long double complex z);
```

These functions compute the complex hyperbolic sine of z .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

csqrt, csqrtf, csqrtl

Complex square root functions

```
#include <complex.h>

double complex csqrt(double complex z);
float complex csqrtf(float complex z);
long double complex csqrtl(long double complex z);
```

These functions compute the complex square root of z , with a branch cut along the negative real axis.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

ctan, ctanf, ctanl

Complex tangent functions

```
#include <complex.h>

double complex ctan(double complex z);
float complex ctanf(float complex z);
long double complex ctanl(long double complex z);
```

These functions compute the complex tangent of z .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

ctanh, ctanhf, ctanhl

Complex hyperbolic tangent functions

```
#include <complex.h>

double complex ctanh(double complex z);
float complex ctanhf(float complex z);
long double complex ctanhl(long double complex z);
```

These functions compute the complex hyperbolic tangent of z .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 These are new functions included for alignment with the ISO/IEC 9899:1999 standard.

ctermid

Generate a pathname for controlling terminal

```
cx #include <stdio.h>
char *ctermid(char *s);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

ctime, ctime_r

Convert a time value to date and time string

```
#include <time.h>
char *ctime(const time_t *clock);
char *ctime_r(const time_t *clock, char *buf);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

dbm_clearerr, dbm_close, dbm_delete, dbm_error, dbm_fetch, dbm_firstkey, dbm_nextkey, dbm_open, dbm_store

Database functions

```
#include <ndbm.h>
int dbm_clearerr(DBM *db);
void dbm_close(DBM *db);
int dbm_delete(DBM *db, datum key);
int dbm_error(DBM *db);
datum dbm_fetch(DBM *db, datum key);
datum dbm_firstkey(DBM *db);
datum dbm_nextkey(DBM *db);
DBM *dbm_open(const char *file, int open_flags, mode_t file_mode);
int dbm_store(DBM *db, datum key, datum content, int store_mode);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

difftime

Compute the difference between two calendar time values

```
#include <time.h>
double difftime(time_t time1, time_t time0);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 No functional changes are made in this issue.

dirname

Report the parent directory name of a file pathname

```
xSI #include <libgen.h>
char *dirname(char *path);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

div

Compute the quotient and remainder of an integer division

```
#include <stdlib.h>
div_t div(int numer, int denom);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 No functional changes are made in this issue.

dlclose

Close a *dlopen()* object

```
xSI #include <dlfcn.h>
int dlclos(void *handle);
```

Derivation First released in Issue 5.

Issue 6 The DESCRIPTION is updated to say that the referenced object is closed “if this is the last reference to it”.

dlerror

Get diagnostic information

```
xSI #include <dlfcn.h>
char *dlerror(void);
```

Derivation First released in Issue 5.

Issue 6 In the DESCRIPTION the note about reentrancy and thread-safety is added. A portable multi-threaded application may only safely call the function when access to the function is serialized.

dlopen

Gain access to an executable object file

```
xSI #include <dlfcn.h>
void *dlopen(const char *file, int mode);
```

Derivation First released in Issue 5.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/21 is applied, changing the default behavior in the DESCRIPTION when neither RTLD_GLOBAL nor RTLD_LOCAL are specified from implementation-defined to unspecified.

dlsym

Obtain the address of a symbol from a *dlopen()* object

```
xSI #include <dlfcn.h>
void *dlsym(void *restrict handle, const char *restrict name);
```

It should be noted that the ISO C standard does not require that pointers to functions can be cast back and forth to pointers to data. Indeed, the ISO C standard does not require that an object of type **void *** can hold a pointer to a function. UNIX system implementations, however, do require that an object of type **void *** can hold a pointer to a function. The result of converting a pointer to a function into a pointer to another data type (except **void ***) is still undefined, however. Note that compilers conforming to the ISO C standard are required to generate a warning if a conversion from a **void *** pointer to a function pointer is attempted as in:

```
fptr = (int (*)(int))dlsym(handle, "my_function");
```

Due to the problem noted here, a future version may either add a new function to return function pointers, or the current interface may be deprecated in favor of two new functions: one that returns data pointers and the other that returns function pointers.

Derivation First released in Issue 5.

Issue 6 The **restrict** keyword is added to the *dlsym()* prototype for alignment with the ISO/IEC 9899:1999 standard.

drand48, erand48, jrand48, lcong48, lrand48, mrand48, nrand48, seed48, srand48

Generate uniformly distributed pseudo-random numbers

```
xSI #include <stdlib.h>
double drand48(void);
double erand48(unsigned short xsubi[3]);
long jrand48(unsigned short xsubi[3]);
void lcong48(unsigned short param[7]);
long lrand48(void);
long mrand48(void);
long nrand48(unsigned short xsubi[3]);
unsigned short *seed48(unsigned short seed16v[3]);
void srand48(long seedval);
```


Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

dup, dup2

Duplicate an open file descriptor

```
#include <unistd.h>

int dup(int fd);
int dup2(int fd, int fd2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

ecvt, fcvt, gcvt

Convert a floating-point number to a string (**LEGACY**)

```
xsi #include <stdlib.h>

char *ecvt(double value, int ndigit, int *restrict decpt,
           int *restrict sign);
char *fcvt(double value, int ndigit, int *restrict decpt,
           int *restrict sign);
char *gcvt(double value, int ndigit, char *buf);
```

Derivation First released in Issue 4, Version 2.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

These functions are marked LEGACY and may not be available on all implementations.

The *sprintf()* function is preferred over these functions.

The **restrict** keyword is added to the *ecvt()* and *fcvt()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION is updated to explicitly use “conversion specification” to describe %g, %f, and %e.

encrypt

Encoding function (**CRYPT**)

```
xsi #include <unistd.h>

void encrypt(char block[64], int edflag);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

endgrent, getgrent, setgrent

Group database entry functions

```
xsl #include <grp.h>
void endgrent(void);
struct group *getgrent(void);
void setgrent(void);
```

Derivation First released in Issue 4, Version 2.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

endhostent, gethostent, sethostent

Network host database functions

```
#include <netdb.h>
void endhostent(void);
struct hostent *gethostent(void);
void sethostent(int stayopen);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

endnetent, getnetbyaddr, getnetbyname, getnetent, setnetent

Network database functions

```
#include <netdb.h>
void endnetent(void);
struct netent *getnetbyaddr(uint32_t net, int type);
struct netent *getnetbyname(const char *name);
struct netent *getnetent(void);
void setnetent(int stayopen);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

endprotoent, getprotobyname, getprotobynumber, getprotoent, setprotoent

Network protocol database functions

```
#include <netdb.h>
void endprotoent(void);
struct protoent *getprotobyname(const char *name);
struct protoent *getprotobynumber(int proto);
struct protoent *getprotoent(void);
void setprotoent(int stayopen);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

endpwent, getpwent, setpwent

User database functions

```
xsl #include <pwd.h>
void endpwent(void);
struct passwd *getpwent(void);
void setpwent(void);
```

Derivation First released in Issue 4, Version 2.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

endservent, getservbyname, getservbyport, getservent, setservent

Network services database functions

```
#include <netdb.h>
void endservent(void);
struct servent *getservbyname(const char *name, const char *proto);
struct servent *getservbyport(int port, const char *proto);
struct servent *getservent(void);
void setservent(int stayopen);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

endutxent, getutxent, getutxid, getutxline, pututxline, setutxent

User accounting database functions

```
xsl #include <utmpx.h>
void endutxent(void);
struct utmpx *getutxent(void);
struct utmpx *getutxid(const struct utmpx *id);
struct utmpx *getutxline(const struct utmpx *line);
struct utmpx *pututxline(const struct utmpx *utmpx);
void setutxent(void);
```

Derivation First released in Issue 4, Version 2.

Issue 6 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

erf, erff, erfl

Error functions

```
#include <math.h>

double erf(double x);
float erff(float x);
long double erfl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *erf()* function is no longer marked as an extension.

The *erfc()* function is split out onto its own reference page.

The *erff()* and *erfl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

erfc, erfcf, erfcl

Complementary error functions

```
#include <math.h>

double erfc(double x);
float erfcf(float x);
long double erfcl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *erfc()* function is no longer marked as an extension.

These functions are split out from the *erf()* reference page.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

errno

Error return value

```
#include <errno.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 Obsolescent text regarding defining *errno* as:

```
extern int errno

is removed.
```

Text regarding no function setting *errno* to zero to indicate an error is changed to no function shall set *errno* to zero. This is for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/23 is applied, adding text to the DESCRIPTION stating that the setting of *errno* after a successful call to a function is unspecified unless the description of the function requires that it will not be modified.

environ, execl, execv, execl, execve, execlp, execvp

Execute a file

```
#include <unistd.h>

extern char **environ;
int execl(const char *path, const char *arg0, ... /*, (char *)0 */);
int execv(const char *path, char *const argv[]);
int execl(const char *path, const char *arg0, ... /*,
          (char *)0, char *const envp[ ]*/);
int execve(const char *path, char *const argv[], char *const envp[]);
int execlp(const char *file, const char *arg0, ... /*, (char *)0 */);
int execvp(const char *file, char *const argv[]);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, behavior is defined for when the process image file is not a valid executable.
- In this issue, `_POSIX_SAVED_IDS` is mandated, thus the effective user ID and effective group ID of the new process image are saved (as the saved set-user-ID and the saved set-group-ID) for use by the `setuid()` function.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [EINVAL] mandatory error condition is added.
- The [ELOOP] optional error condition is added.

The description of CPU-time clock semantics is added for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics for typed memory.

The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

IEEE PASC Interpretation 1003.1 #132 is applied. This addresses the issue of whether or not an implementation allows `SIG_IGN` as a `SIGCHLD` disposition to be inherited across a call to one of the *exec* family of functions or *posix_spawn()*, leaving it explicitly unspecified.

The DESCRIPTION is updated to make it explicit that the floating-point environment in the new process image is set to the default.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/15 is applied, adding a new paragraph to the DESCRIPTION and text to the end of the APPLICATION USAGE section. This change addresses a security concern, where implementations may want to reopen file descriptors 0, 1, and 2 for programs with the set-user-id or set-group-id file mode bits calling the *exec* family of functions.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/24 is applied, applying changes to the DESCRIPTION, addressing which attributes are inherited by threads, and behavioral requirements for threads attributes.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/25 is applied, updating text in the RATIONALE from “the process signal mask be unchanged across an *exec*” to “the new process image inherits the signal mask of the thread that called *exec* in the old process image”.

exit, _Exit, _exit

Terminate a process

```
#include <stdlib.h>

void exit(int status);
void _Exit(int status);

#include <unistd.h>

void _exit(int status);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics for typed memory.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *_Exit()* function is included.
- The DESCRIPTION is updated.

The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

References to the *wait3()* function are removed.

exp, expf, expl

Exponential function

```
#include <math.h>

double exp(double x);
float expf(float x);
long double expl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *expf()* and *expl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

exp2, exp2f, exp2l

Exponential base 2 functions

```
#include <math.h>

double exp2(double x);
float exp2f(float x);
long double exp2l(long double x);
```

These functions compute the base-2 exponential of x .

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

expm1, expm1f, expm1l

Compute exponential functions

```
#include <math.h>

double expm1(double x);
float expm1f(float x);
long double expm1l(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *expm1f()* and *expm1l()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The *expm1()* function is no longer marked as an extension.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

fabs, fabsf, fabsl

Absolute value function

```
#include <math.h>

double fabs(double x);
float fabsf(float x);
long double fabsl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *fabsf()* and *fabsl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

fattach

Attach a STREAMS-based file descriptor to a file in the file system name space (**STREAMS**)

```
XSR #include <stropts.h>
int fattach(int fildev, const char *path);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked as part of the XSI STREAMS Option Group and need not be supported on all implementations supporting the Single UNIX Specification.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

fchdir

Change working directory

```
XSI #include <unistd.h>
int fchdir(int fildev);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

fchmod

Change mode of a file

```
#include <sys/stat.h>
int fchmod(int fildev, mode_t mode);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by stating that *fchmod()* behavior is unspecified for typed memory objects.

fchown

Change owner and group of a file

```
#include <unistd.h>
int fchown(int fildev, uid_t owner, gid_t group);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that a call to *fchown()* may not be allowed on a pipe.

The *fchown()* function is defined as mandatory.

fclose

Close a stream

```
#include <stdio.h>
int fclose(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EFBIG] error is added as part of the large file support extensions.
- The [ENXIO] optional error condition is added.

The DESCRIPTION is updated to note that the stream and any buffer are disassociated whether or not the call succeeds. This is for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/28 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fcntl

File control

OH

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

```
int fcntl(int fildes, int cmd, ...);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, sentences describing behavior when *l_len* is negative are now mandated, and the description of unlock (F_UNLOCK) when *l_len* is non-negative is mandated.
- In the ERRORS section, the [EINVAL] error condition has the case mandated when the *cmd* is invalid, and two [E_OVERFLOW] error conditions are added.

The F_GETOWN and F_SETOWN values are added for sockets.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that the extent of the bytes locked is determined prior to the blocking action.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *fcntl()* results are unspecified for typed memory objects.

fdatasync

Synchronize the data of a file (**REALTIME**)

```
sio #include <unistd.h>
int fdatasync(int fildes);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Synchronized Input and Output option. The *fdatasync()* function is marked as part of the Synchronized Input and Output option.

fdetach

Detach a name from a STREAMS-based file descriptor (**STREAMS**)

```
xsr #include <stropts.h>
int fdetach(const char *path);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

fdim, fdimf, fdiml

Compute positive difference between two floating-point numbers

```
#include <math.h>
double fdim(double x, double y);
float fdimf(float x, float y);
long double fdiml(long double x, long double y);
```

These functions determine the positive difference between their arguments.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

fdopen

Associate a stream with a file descriptor

```
cx #include <stdio.h>
FILE *fdopen(int fildes, const char *mode);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the use and setting of the *mode* argument are changed to include binary streams.

- In the DESCRIPTION, text is added for large file support to indicate setting of the offset maximum in the open file description.
- All errors identified in the ERRORS section are added.
- In the DESCRIPTION, text is added that the *fdopen()* function may cause *st_atime* to be updated.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that it is the responsibility of the application to ensure that the mode is compatible with the open file descriptor.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *fdopen()* results are unspecified for typed memory objects.

feclearexcept

Clear floating-point exception

```
#include <fenv.h>

int feclearexcept(int excepts);
```

The *feclearexcept()* function attempts to clear the supported floating-point exceptions represented by *excepts*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

fegetenv, fesetenv

Get and set current floating-point environment

```
#include <fenv.h>

int fegetenv(fenv_t *envp);
int fesetenv(const fenv_t *envp);
```

The *fegetenv()* function attempts to store the current floating-point environment in the object pointed to by *envp*.

The *fesetenv()* function attempts to establish the floating-point environment represented by the object pointed to by *envp*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

fegetexceptflag, fesetexceptflag

Get and set floating-point status flags

```
#include <fenv.h>

int fegetexceptflag(fexcept_t *flagp, int excepts);
int fesetexceptflag(const fexcept_t *flagp, int excepts);
```

The *fegetexceptflag()* function attempts to store an implementation-defined representation of the states of the floating-point status flags indicated by the argument *excepts* in the object pointed to by the argument *flagp*.

The *fesetexceptflag()* function attempts to set the floating-point status flags indicated by the argument *excepts* to the states stored in the object pointed to by *flagp*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.
ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

fegetround, fesetround

Get and set current rounding direction

```
#include <fenv.h>

int fegetround(void);
int fesetround(int round);
```

The *fegetround()* function gets the current rounding direction.

The *fesetround()* function establishes the rounding direction represented by its argument *round*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.
ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

feholdexcept

Save current floating-point environment

```
#include <fenv.h>

int feholdexcept(fenv_t *envp);
```

The *feholdexcept()* function saves the current floating-point environment in the object pointed to by *envp*, clears the floating-point status flags, and then installs a non-stop (continue on floating-point exceptions) mode, if available, for all floating-point exceptions.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

feof

Test end-of-file indicator on a stream

```
#include <stdio.h>

int feof(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

feraiseexcept

Raise floating-point exception

```
#include <fenv.h>

int feraiseexcept(int excepts);
```

The *feraiseexcept()* function attempts to raise the supported floating-point exceptions represented by the argument *excepts*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.
ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

ferror

Test error indicator on a stream

```
#include <stdio.h>
int ferror(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

fetestexcept

Test floating-point exception flags

```
#include <fenv.h>
int fetestexcept(int excepts);
```

The *fetestexcept()* function determines which of a specified subset of the floating-point exception flags are currently set. The *excepts* argument specifies the floating-point status flags to be queried.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

feupdateenv

Update floating-point environment

```
#include <fenv.h>
int feupdateenv(const fenv_t *envp);
```

The *feupdateenv()* function attempts to save the currently raised floating-point exceptions in its automatic storage, attempts to install the floating-point environment represented by the object pointed to by *envp*, and then attempts to raise the saved floating-point exceptions.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

fflush

Flush a stream

```
#include <stdio.h>
int fflush(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EFBIG] error is added as part of the large file support extensions.
- The [ENXIO] optional error condition is added.

The RETURN VALUE section is updated to note that the error indicator is set for the stream. This is for alignment with the ISO/IEC 9899: 1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/31 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

ffs

Find first set bit

```
xSI #include <strings.h>
int ffs(int i);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

fgetc

Get a byte from a stream

```
#include <stdio.h>
int fgetc(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] and [EOVERFLOW] mandatory error conditions are added.
- The [ENOMEM] and [ENXIO] optional error conditions are added.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The DESCRIPTION is updated to clarify the behavior when the end-of-file indicator for the input stream is not set.
- The RETURN VALUE section is updated to note that the error indicator is set for the stream.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/32 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fgetpos

Get current file position information

```
#include <stdio.h>
int fgetpos(FILE *restrict stream, fpos_t *restrict pos);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] mandatory error condition is added.
- The [EBADF] and [ESPIPE] optional error conditions are added.

An additional [ESPIPE] error condition is added for sockets.

The prototype for *fgetpos()* is changed for alignment with the ISO/IEC 9899:1999 standard.

fgets

Get a string from a stream

```
#include <stdio.h>

char *fgets(char *restrict s, int n, FILE *restrict stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The prototype for *fgets()* is changed for alignment with the ISO/IEC 9899:1999 standard.

fgetwc

Get a wide-character code from a stream

```
#include <stdio.h>
#include <wchar.h>

wint_t fgetwc(FILE *stream);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] and [Eoverflow] mandatory error conditions are added.
- The [ENOMEM] and [ENXIO] optional error conditions are added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/33 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fgetws

Get a wide-character string from a stream

```
#include <stdio.h>
#include <wchar.h>

wchar_t *fgetws(wchar_t *restrict ws, int n,
                FILE *restrict stream);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The prototype for *fgetws()* is changed for alignment with the ISO/IEC 9899:1999 standard.

fileno

Map a stream pointer to a file descriptor

```
cx #include <stdio.h>
int fileno(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EBADF] optional error condition is added.

flockfile, ftrylockfile, funlockfile

Stdio locking functions

```
TSF #include <stdio.h>

void flockfile(FILE *file);
int ftrylockfile(FILE *file);
void funlockfile(FILE *file);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 These functions are marked as part of the Thread-Safe Functions option. Support for this option is mandatory on all systems supporting the Single UNIX Specification.

floor, floorf, floorl

Floor function

```
#include <math.h>

double floor(double x);
float floorf(float x);
long double floorl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *floorf()* and *floorl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

fma, fmaf, fmal

Floating-point multiply-add

```
#include <math.h>

double fma(double x, double y, double z);
float fmaf(float x, float y, float z);
long double fmal(long double x, long double y, long double z);
```

These functions compute $(x * y) + z$, rounded as one ternary operation: they compute the value (as if) to infinite precision and round once to the result format, according to the rounding mode characterized by the value of FLT_ROUNDS.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

fmax, fmaxf, fmaxl

Determine maximum numeric value of two floating-point numbers

```
#include <math.h>

double fmax(double x, double y);
float fmaxf(float x, float y);
long double fmaxl(long double x, long double y);
```

These functions determine the maximum numeric value of their arguments. NaN arguments are treated as missing data: if one argument is a NaN and the other numeric, then these functions choose the numeric value.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

fmin, fminf, fminl

Determine minimum numeric value of two floating-point numbers

```
#include <math.h>

double fmin(double x, double y);
float fminf(float x, float y);
long double fminl(long double x, long double y);
```

These functions determine the minimum numeric value of their arguments. NaN arguments are treated as missing data: if one argument is a NaN and the other numeric, then these functions choose the numeric value.

Derivation First released in Issue 6. Derived from ISO/IEC 9899: 1999 standard.

fmod, fmodf, fmodl

Floating-point remainder value function

```
#include <math.h>

double fmod(double x, double y);
float fmodf(float x, float y);
long double fmodl(long double x, long double y);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The behavior for when the *y* argument is zero is now defined.

The *fmodf()* and *fmodl()* functions are added for alignment with the ISO/IEC 9899: 1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899: 1999 standard.

IEC 60559: 1989 standard floating-point extensions over the ISO/IEC 9899: 1999 standard are marked.

fmtmsg

Display a message in the specified format on standard error and/or a system console

xSI

```
#include <fmtmsg.h>

int fmtmsg(long classification, const char *label, int severity,
           const char *text, const char *action, const char *tag);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

fnmatch

Match a filename or a pathname

```
#include <fnmatch.h>

int fnmatch(const char *pattern, const char *string, int flags);
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 No functional changes are made in this issue.

fopen

Open a stream

```
#include <stdio.h>

FILE *fopen(const char *restrict filename, const char *restrict mode);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
- The [ELOOP] mandatory error condition is added.
- The [EINVAL], [EMFILE], [ENAMETOOLONG], [ENOMEM], and [ETXTBSY] optional error conditions are added.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototype for *fopen()* is updated.
- The DESCRIPTION is updated to note that if the argument *mode* points to a string other than those listed, then the behavior is undefined.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

fork

Create a new process

```
#include <unistd.h>

pid_t fork(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes were made to align with the IEEE P1003.1a draft standard:

- The effect of *fork()* on a pending alarm call in the child process is clarified.

The description of CPU-time clock semantics is added for alignment with IEEE Std 1003.1d-1999.

The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/17 is applied, adding text to the DESCRIPTION relating to fork handlers registered by the *pthread_atfork()* function and async-signal safety. If any of the fork handlers calls a function that is not async-signal safe, the behavior is undefined.

fpathconf, pathconf

Get configurable pathname variables

```
#include <unistd.h>

long fpathconf(int fildes, int name);
long pathconf(const char *path, int name);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to include {FILESIZEBITS}.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The `_PC_SYMLINK_MAX` entry is added to the table in the DESCRIPTION.

The following *pathconf()* variables and their associated names are added for alignment with IEEE Std 1003.1d-1999:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/18 is applied, changing the fourth paragraph of the DESCRIPTION and removing shading and margin markers from the table. This change is needed since implementations are required to support all of these symbols.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/34 is applied, adding the table entry for POSIX2_SYMLINKS in the DESCRIPTION.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/35 is applied, updating the DESCRIPTION and RATIONALE sections to clarify behavior for the {POSIX_ALLOC_SIZE_MIN}, {POSIX_REC_INCR_XFER_SIZE}, {POSIX_REC_MAX_XFER_SIZE}, {POSIX_REC_MIN_XFER_SIZE}, and {POSIX_REC_XFER_ALIGN} variables.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/36 is applied, updating the RETURN VALUE and APPLICATION USAGE sections to state that the results are unspecified if a variable is dependent on an unsupported option, and advising application writers to check for supported options prior to obtaining and using such values.

fpclassify

Classify real floating type

```
#include <math.h>

int fpclassify(real-floating x);
```

The *fpclassify()* macro classifies its argument value as NaN, infinite, normal, subnormal, zero, or into another implementation-defined category.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

Issue 6 This is a new function included for alignment with the ISO/IEC 9899:1999 standard.

fprintf, printf, snprintf, sprintf

Print formatted output

```
#include <stdio.h>

int fprintf(FILE *restrict stream, const char *restrict format, ...);
int printf(const char *restrict format, ...);
int snprintf(char *restrict s, size_t n,
             const char *restrict format, ...);
int sprintf(char *restrict s, const char *restrict format, ...);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fprintf()*, *printf()*, *snprintf()*, and *sprintf()* are updated, and the XSI shading is removed from *snprintf()*.
- The description of *snprintf()* is aligned with the ISO C standard. Note that this supersedes the *snprintf()* description in The Open Group Base Resolution bwg98-006, which changed the behavior from Issue 5.
- The DESCRIPTION is updated.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

ISO/IEC 9899:1999 standard, Technical Corrigendum No. 1 is incorporated.

fputc

Put a byte on a stream

```
#include <stdio.h>
int fputc(int c, FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] and [EFBIG] mandatory error conditions are added.
- The [ENOMEM] and [ENXIO] optional error conditions are added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/37 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fputs

Put a string on a stream

```
#include <stdio.h>
int fputs(const char *restrict s, FILE *restrict stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *fputs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

fputwc

Put a wide-character code on a stream

```
#include <stdio.h>
#include <wchar.h>
wint_t fputwc(wchar_t wc, FILE *stream);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EFBIG] and [EIO] mandatory error conditions are added.
- The [ENOMEM] and [ENXIO] optional error conditions are added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/38 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fputws

Put a wide-character string on a stream

```
#include <stdio.h>
#include <wchar.h>
```

```
int fputws(const wchar_t *restrict ws, FILE *restrict stream);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The *fputws()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

fread

Binary input

```
#include <stdio.h>
```

```
size_t fread(void *restrict ptr, size_t size, size_t nitems,
             FILE *restrict stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *fread()* prototype is updated.
- The DESCRIPTION is updated to describe how the bytes from a call to *fgetc()* are stored.

free

Free allocated memory

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

```
void free(void *ptr);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 Reference to the *valloc()* function is removed.

freeaddrinfo, getaddrinfo

Get address information

```
#include <sys/socket.h>
```

```
#include <netdb.h>
```

```
void freeaddrinfo(struct addrinfo *ai);
int getaddrinfo(const char *restrict nodename,
               const char *restrict servname,
               const struct addrinfo *restrict hints,
               struct addrinfo **restrict res);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The **restrict** keyword is added to the *getaddrinfo()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/20 is applied, making changes for alignment with IPv6. These include the following:

- Adding AI_V4MAPPED, AI_ALL, and AI_ADDRCONFIG to the allowed values for the *ai_flags* field
- Adding a description of AI_ADDRCONFIG
- Adding a description of the consequences of ignoring the AI_PASSIVE flag

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/39 is applied, changing “corresponding value” to “corresponding error value” in the ERRORS section.

freopen

Open a stream

```
#include <stdio.h>
```

```
FILE *freopen(const char *restrict filename, const char *restrict mode,  
FILE *restrict stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [EINVAL], [ENOMEM], [ENXIO], and [ETXTBSY] optional error conditions are added.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *freopen()* prototype is updated.
- The DESCRIPTION is updated.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

The DESCRIPTION is updated regarding failure to close, changing the “file” to “file descriptor”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/40 is applied, adding the following sentence to the DESCRIPTION: “In this case, the file descriptor associated with the stream need not be closed if the call to *freopen()* succeeds.”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/41 is applied, adding an mandatory [EBADF] error, and an optional [EBADF] error to the ERRORS section.

frexp, frexpf, frexpl

Extract mantissa and exponent from a double precision number

```
#include <math.h>

double frexp(double num, int *exp);
float frexpf(float num, int *exp);
long double frexpl(long double num, int *exp);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *frexpf()* and *frexpl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

fscanf, scanf, sscanf

Convert formatted input

```
#include <stdio.h>

int fscanf(FILE *restrict stream, const char *restrict format, ... );
int scanf(const char *restrict format, ... );
int sscanf(const char *restrict s, const char *restrict format, ... );
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The Open Group Corrigendum U021/7 and U028/10 are applied. These correct several occurrences of “characters” in the text which have been replaced with the term “bytes”.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fscanf()*, *scanf()*, and *sscanf()* are updated.
- The DESCRIPTION is updated.
- The hh, ll, j, t, and z length modifiers are added.
- The a, A, and F conversion characters are added.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

fseek, fseeko

Reposition a file-position indicator in a stream

```
#include <stdio.h>

int fseek(FILE *stream, long offset, int whence);
int fseeko(FILE *stream, off_t offset, int whence);
```

CX

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The `fseeko()` function is added.
- The [EFBIG], [EOVERFLOW], and [ENXIO] mandatory error conditions are added.

The following change is incorporated for alignment with the FIPS requirements:

- The [EINTR] error is no longer an indication that the implementation does not report partial transfers.

The DESCRIPTION is updated to explicitly state that `fseek()` sets the file-position indicator, and then on error the error indicator is set and `fseek()` fails. This is for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/42 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fsetpos

Set current file position

```
#include <stdio.h>
```

```
int fsetpos(FILE *stream, const fpos_t *pos);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 An additional [ESPIPE] error condition is added for sockets.

The DESCRIPTION is updated to clarify that the error indicator is set for the stream on a read or write error. This is for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/21 is applied, deleting an erroneous [EINVAL] error case from the ERRORS section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/43 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

fstat

Get file status

```
#include <sys/stat.h>
```

```
int fstat(int fildes, struct stat *buf);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] mandatory error condition is added.
- The [EOVERFLOW] mandatory error condition is added. This change is to support large files.

- The [Eoverflow] optional error condition is added.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that shared memory object semantics apply to typed memory objects.

fstatvfs, statvfs

Get file system information

```
XSI #include <sys/statvfs.h>
int fstatvfs(int fildev, struct statvfs *buf);
int statvfs(const char *restrict path, struct statvfs *restrict buf);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The **restrict** keyword is added to the *statvfs()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

fsync

Synchronize changes to a file

```
FSC #include <unistd.h>
int fsync(int fildev);
```

Derivation First released in Issue 3.

Issue 6 This function is marked as part of the File Synchronization option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] and [EIO] mandatory error conditions are added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/44 is applied, applying an editorial rewording of the DESCRIPTION. No change in meaning is intended.

ftell, ftello

Return a file offset in a stream

```
CX #include <stdio.h>
long ftell(FILE *stream);
off_t ftello(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The *ftello()* function is added.
- The [Eoverflow] error conditions are added.

An additional [ESPIPE] error condition is added for sockets.

ftime

Get date and time (**LEGACY**)

```
xsi #include <sys/timeb.h>
int ftime(struct timeb *tp);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations. For applications portability, the *time()* function should be used to determine the current time instead of *ftime()*. Realtime applications should use *clock_gettime()* to determine the current time instead of *ftime()*.

The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since 00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time* functions.

ftok

Generate an IPC key

```
xsi #include <sys/ipc.h>
key_t ftok(const char *path, int id);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

ftruncate

Truncate a file to a specified length

```
#include <unistd.h>
int ftruncate(int fildes, off_t length);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *truncate()* function is split out into a separate reference page.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is changed to indicate that if the file size is changed, and if the file is a regular file, the S_ISUID and S_ISGID bits in the file mode may be cleared.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION text is updated.

XSI-conformant systems are required to increase the size of the file if the file was previously smaller than the size requested.

ftw

Traverse (walk) a file tree

XSI

```
#include <ftw.h>

int ftw(const char *path, int (*fn)(const char *,
    const struct stat *ptr, int flag), int ndirs);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The ERRORS section is updated as follows:

- The wording of the mandatory [ELOOP] error condition is updated.
- A second optional [ELOOP] error condition is added.
- The [E_OVERFLOW] mandatory error condition is added.

Text is added to the DESCRIPTION to say that the *ftw()* function need not be reentrant and that the results are unspecified if the application-supplied *fn* function does not preserve the current working directory.

fwide

Set stream orientation

```
#include <stdio.h>
#include <wchar.h>

int fwide(FILE *stream, int mode);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

fwprintf, swprintf, wprintf

Print formatted wide-character output

```
#include <stdio.h>
#include <wchar.h>

int fwprintf(FILE *restrict stream, const wchar_t *restrict format, ...);
int swprintf(wchar_t *restrict ws, size_t n,
    const wchar_t *restrict format, ...);
int wprintf(const wchar_t *restrict format, ...);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The Open Group Corrigendum U040/1 is applied to the RETURN VALUE section, describing the case if *n* or more wide characters are requested to be written using *swprintf()*.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fwprintf()*, *swprintf()*, and *wprintf()* are updated.

- The DESCRIPTION is updated.
- The hh, ll, j, t, and z length modifiers are added.
- The a, A, and F conversion characters are added.
- XSI shading is removed from the description of character string representations of infinity and NaN floating-point values.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

ISO/IEC 9899:1999 standard, Technical Corrigendum No. 1 is incorporated.

fwrite

Binary output

```
#include <stdio.h>

size_t fwrite(const void *restrict ptr, size_t size, size_t nitems,
              FILE *restrict stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *fwrite()* prototype is updated.
- The DESCRIPTION is updated to clarify how the data is written out using *fputc()*.

fwscanf, swscanf, wscanf

Convert formatted wide-character input

```
#include <stdio.h>
#include <wchar.h>

int fwscanf(FILE *restrict stream, const wchar_t *restrict format, ... );
int swscanf(const wchar_t *restrict ws,
            const wchar_t *restrict format, ... );
int wscanf(const wchar_t *restrict format, ... );
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fwscanf()* and *swscanf()* are updated.
- The DESCRIPTION is updated.
- The hh, ll, j, t, and z length modifiers are added.
- The a, A, and F conversion characters are added.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

gai_strerror

Address and name information error description

```
#include <netdb.h>

const char *gai_strerror(int ecode);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The Open Group Base Resolution bwg2001-009 is applied, which changes the return type from **char *** to **const char ***. This is for coordination with the IPnG Working Group.

getc

Get a byte from a stream

```
#include <stdio.h>

int getc(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

getc_unlocked, getchar_unlocked, putc_unlocked, putchar_unlocked

Stdio with explicit client locking

```
TSF #include <stdio.h>

int getc_unlocked(FILE *stream);
int getchar_unlocked(void);
int putc_unlocked(int c, FILE *stream);
int putchar_unlocked(int c);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 These functions are marked as part of the Thread-Safe Functions option. Support for this option is mandatory on all systems supporting the Single UNIX Specification.

The Open Group Corrigendum U030/2 is applied, adding APPLICATION USAGE describing how applications should be written to avoid the case when the functions are implemented as macros.

getchar

Get a byte from a stdin stream

```
#include <stdio.h>

int getchar(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

getcontext, setcontext

Get and set current user context

```
OB XSI #include <ucontext.h>
int getcontext(ucontext_t *ucp);
int setcontext(const ucontext_t *ucp);
```

Derivation First released in Issue 4, Version 2.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/45 is applied, updating the SYNOPSIS and APPLICATION USAGE sections to note that the *getcontext()* and *setcontext()* functions are obsolescent.

getcwd

Get the pathname of the current working directory

```
#include <unistd.h>
char *getcwd(char *buf, size_t size);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ENOMEM] optional error condition is added.

getdate

Convert user format date and time

```
XSI #include <time.h>
struct tm *getdate(const char *string);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since 00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time* functions.

The description of %s is updated so that the valid range is [00,60] rather than [00,61].

The DESCRIPTION is updated to refer to conversion specifications instead of field descriptors for consistency with other functions.

getegid

Get the effective group ID

```
#include <unistd.h>
gid_t getegid(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getenv

Get value of an environment variable

```
#include <stdlib.h>

char *getenv(const char *name);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes were made to align with the IEEE P1003.1a draft standard:

- References added to the new *setenv()* and *unsetenv()* functions.

geteuid

Get the effective user ID

```
#include <unistd.h>

uid_t geteuid(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getgid

Get the real group ID

```
#include <unistd.h>

gid_t getgid(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getgrgid, getgrgid_r

Get group database entry for a group ID

```
#include <grp.h>

struct group *getgrgid(gid_t gid);
TSF int getgrgid_r(gid_t gid, struct group *grp, char *buffer,
                size_t bufsize, struct group **result);
```

Derivation First released in Issue 1. Derived from System V Release 2.0.

Issue 6 The *getgrgid_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *gid*.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

getgrnam, getgrnam_r

Search group database for a name
#include <grp.h>

```
struct group *getgrnam(const char *name);
TSF int getgrnam_r(const char *name, struct group *grp, char *buffer,
    size_t bufsize, struct group **result);
```

Derivation First released in Issue 1. Derived from System V Release 2.0.

Issue 6 The *getgrnam_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

getgroups

Get supplementary group IDs

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

```
int getgroups(int gidsetsize, gid_t grouplist[]);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.
The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- A return value of 0 is not permitted, because `{NGROUPS_MAX}` cannot be 0. This is a FIPS requirement.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added that the effective group ID may be included in the supplementary group list.

gethostbyaddr, gethostbyname

Network host database functions

```
#include <netdb.h>
```

```
OB struct hostent *gethostbyaddr(const void *addr, socklen_t len,
    int type);
    struct hostent *gethostbyname(const char *name);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 These functions are marked obsolescent. The `getaddrinfo()` and `getnameinfo()` functions are preferred over the `gethostbyaddr()` and `gethostbyname()` functions.

gethostid

Get an identifier for the current host

```
XSI #include <unistd.h>
    long gethostid(void);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

gethostname

Get name of current host

```
#include <unistd.h>
    int gethostname(char *name, size_t namelen);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The Open Group Base Resolution bwg2001-008 is applied, changing the `namelen` parameter from **socklen_t** to **size_t**.

getitimer, setitimer

Get and set value of interval timer

```
XSI #include <sys/time.h>

int getitimer(int which, struct itimerval *value);
int setitimer(int which, const struct itimerval *restrict value,
              struct itimerval *restrict ovalue);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The **restrict** keyword is added to the *setitimer()* prototype for alignment with the ISO/IEC 9899:1999 standard.

getlogin, getlogin_r

Get login name

```
TSF #include <unistd.h>

char *getlogin(void);
int getlogin_r(char *name, size_t namesize);
```

Derivation First released in Issue 1. Derived from System V Release 2.0.

Issue 6 The *getlogin_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EMFILE], [ENFILE], and [ENXIO] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

getmsg, getpmsg

Receive next message from a STREAMS file (**STREAMS**)

```
XSR #include <stropts.h>

int getmsg(int fildes, struct strbuf *restrict ctlptr,
           struct strbuf *restrict dataptr, int *restrict flagsp);
int getpmsg(int fildes, struct strbuf *restrict ctlptr,
            struct strbuf *restrict dataptr, int *restrict bandp,
            int *restrict flagsp);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked as part of the XSI STREAMS Option Group and need not be supported on all implementations supporting the Single UNIX Specification.

The **restrict** keyword is added to the *getmsg()* and *getpmsg()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

getnameinfo

Get name information

```
#include <sys/socket.h>
#include <netdb.h>

int getnameinfo(const struct sockaddr *restrict sa, socklen_t salen,
    char *restrict node, socklen_t nodelen, char *restrict service,
    socklen_t servicelen, unsigned flags);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The **restrict** keyword is added to the *getnameinfo()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/23 is applied, making various changes in the SYNOPSIS and DESCRIPTION for alignment with IPv6.

getopt, optarg, opterr, optind, optopt

Command option parsing

```
#include <unistd.h>

int getopt(int argc, char *const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 IEEE PASC Interpretation 1003.2 #150 is applied, changing the words “not less than *argc*” to “not greater than *argc*”.

getpeername

Get the name of the peer socket

```
#include <sys/socket.h>

int getpeername(int socket, struct sockaddr *restrict address,
    socklen_t *restrict address_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The **restrict** keyword is added to the *getpeername()* prototype for alignment with the ISO/IEC 9899:1999 standard.

getpgid

Get the process group ID for a process

```
xSI #include <unistd.h>
pid_t getpgid(pid_t pid);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

getpgrp

Get the process group ID of the calling process

```
#include <unistd.h>
pid_t getpgrp(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getpid

Get the process ID

```
#include <unistd.h>
pid_t getpid(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getppid

Get the parent process ID

```
#include <unistd.h>
pid_t getppid(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getpriority, setpriority

Get and set the nice value

```
xSI #include <sys/resource.h>
int getpriority(int which, id_t who);
int setpriority(int which, id_t who, int value);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

getpwnam, getpwnam_r

Search user database for a name

```
#include <pwd.h>
```

```
struct passwd *getpwnam(const char *name);
TSF int getpwnam_r(const char *name, struct passwd *pwd, char *buffer,
    size_t bufsize, struct passwd **result);
```

Derivation First released in Issue 1. Derived from System V Release 2.0.

Issue 6 The *getpwnam_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *name*.

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EMFILE], [ENFILE], and [ENXIO] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

getpwuid, getpwuid_r

Search user database for a user ID

```
#include <pwd.h>
```

```
struct passwd *getpwuid(uid_t uid);
TSF int getpwuid_r(uid_t uid, struct passwd *pwd, char *buffer,
    size_t bufsize, struct passwd **result);
```

Derivation First released in Issue 1. Derived from System V Release 2.0.

Issue 6 The *getpwuid_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *uid*.

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when

access to the function is serialized.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

getrlimit, setrlimit

Control maximum resource consumption

```
xsi #include <sys/resource.h>
int getrlimit(int resource, struct rlimit *rlp);
int setrlimit(int resource, const struct rlimit *rlp);
```

Derivation First released in Issue 4, Version 2.

Issue 6 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/25 is applied, changing wording for RLIMIT_NOFILE in the DESCRIPTION related to functions that allocate a file descriptor failing with [EMFILE]. Text is added to the APPLICATION USAGE section noting the consequences of a process attempting to set the hard or soft limit for RLIMIT_NOFILE less than the highest currently open file descriptor +1.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/46 is applied, updating the definition of RLIMIT_STACK in the DESCRIPTION section from “the maximum size of a process stack” to “the maximum size of the initial thread’s stack”. Text is added to the RATIONALE section.

getrusage

Get information about resource utilization

```
xsi #include <sys/resource.h>
int getrusage(int who, struct rusage *r_usage);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

gets

Get a string from a stdin stream

```
#include <stdio.h>
char *gets(char *s);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

getsid

Get the process group ID of a session leader

```
xSI #include <unistd.h>
pid_t getsid(pid_t pid);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

getsockname

Get the socket name

```
#include <sys/socket.h>
int getsockname(int socket, struct sockaddr *restrict address,
                socklen_t *restrict address_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The **restrict** keyword is added to the *getsockname()* prototype for alignment with the ISO/IEC 9899:1999 standard.

getsockopt

Get the socket options

```
#include <sys/socket.h>
int getsockopt(int socket, int level, int option_name,
               void *restrict option_value, socklen_t *restrict option_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The **restrict** keyword is added to the *getsockopt()* prototype for alignment with the ISO/IEC 9899:1999 standard.

getsubopt

Parse suboption arguments from a string

```
xSI #include <stdlib.h>
int getsubopt(char **optionp, char *const *keylistp, char **valuep);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

gettimeofday

Get the date and time

```
xSI #include <sys/time.h>
int gettimeofday(struct timeval *restrict tp, void *restrict tzp);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since 00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time* functions.

The **restrict** keyword is added to the *gettimeofday()* prototype for alignment with the ISO/IEC 9899:1999 standard.

getuid

Get a real user ID

```
#include <unistd.h>
uid_t getuid(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

getwc

Get a wide character from a stream

```
#include <stdio.h>
#include <wchar.h>
wint_t getwc(FILE *stream);
```

Derivation First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

getwchar

Get a wide character from a stdin stream

```
#include <wchar.h>
wint_t getwchar(void);
```

Derivation First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

getwd

Get the current working directory pathname (**LEGACY**)

XSI

```
#include <unistd.h>
char *getwd(char *path_name);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations. For applications portability, the *getcwd()* function should be used to determine the current working directory instead of *getwd()*.

glob, globfree

Generate pathnames matching a pattern

```
#include <glob.h>
int glob(const char *restrict pattern, int flags,
         int(*errfunc)(const char *epath, int errno),
         glob_t *restrict pglob);
void globfree(glob_t *pglob);
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 The **restrict** keyword is added to the *glob()* prototype for alignment with the ISO/IEC 9899:1999 standard.

gmtime, gmtime_r

Convert a time value to a broken-down UTC time

```
#include <time.h>
struct tm *gmtime(const time_t *timer);
struct tm *gmtime_r(const time_t *restrict timer,
                   struct tm *restrict result);
```

TSF

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *gmtime_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *gmtime_r()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/27 is applied, adding the [Eoverflow] error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/48 is applied, updating the error handling for *gmtime_r()*.

grantpt

Grant access to the slave pseudo-terminal device

```
XSI #include <stdlib.h>
int grantpt(int fildes);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

h_errno

Error return value for network database operations

```
OB #include <netdb.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 Applications should obtain the definition of *h_errno* by the inclusion of the **<netdb.h>** header.

Note that this method of returning errors is used only in connection with obsolescent functions and may be withdrawn in the future.

hcreate, hdestroy, hsearch

Manage hash search table

```
XSI #include <search.h>
int hcreate(size_t nel);
void hdestroy(void);
ENTRY *hsearch(ENTRY item, ACTION action);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

htonl, htons, ntohl, ntohs

Convert values between host and network byte order

```
#include <arpa/inet.h>
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

hypot, hypotf, hypotl

Euclidean distance function

```
#include <math.h>

double hypot(double x, double y);
float hypotf(float x, float y);
long double hypotl(long double x, long double y);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *hypot()* function is no longer marked as an extension.

The *hypotf()* and *hypotl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

iconv

Codeset conversion function

XSI

```
#include <iconv.h>

size_t iconv(iconv_t cd, char **restrict inbuf,
             size_t *restrict inbytesleft, char **restrict outbuf,
             size_t *restrict outbytesleft);
```

Derivation First released in Issue 4. Derived from the HP-UX Manual.

Issue 6 The SYNOPSIS has been corrected to align with the **<iconv.h>** reference page.

The **restrict** keyword is added to the *iconv()* prototype for alignment with the ISO/IEC 9899:1999 standard.

iconv_close

Codeset conversion deallocation function

XSI

```
#include <iconv.h>

int iconv_close(iconv_t cd);
```

Derivation First released in Issue 4. Derived from the HP-UX Manual.

Issue 6 No functional changes are made in this issue.

iconv_open

Codeset conversion allocation function

```
xSI #include <iconv.h>
iconv_t iconv_open(const char *tocode, const char *fromcode);
```

Derivation First released in Issue 4. Derived from the HP-UX Manual.

Issue 6 No functional changes are made in this issue.

if_freenameindex

Free memory allocated by *if_nameindex()*

```
#include <net/if.h>
void if_freenameindex(struct if_nameindex *ptr);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

if_indextoname

Map a network interface index to its corresponding name

```
#include <net/if.h>
char *if_indextoname(unsigned ifindex, char *ifname);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/28 is applied, changing {IFNAMSIZ} to {IF_NAMESIZ} in the DESCRIPTION.

if_nameindex

Return all network interface names and indexes

```
#include <net/if.h>
struct if_nameindex *if_nameindex(void);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

if_nametoindex

Map a network interface name to its corresponding index

```
#include <net/if.h>
unsigned if_nametoindex(const char *ifname);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

ilogb, ilogbf, ilogbl

Return an unbiased exponent

```
#include <math.h>

int ilogb(double x);
int ilogbf(float x);
int ilogbl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *ilogb()* function is no longer marked as an extension.

The *ilogbf()* and *ilogbl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The RETURN VALUE section is revised for alignment with the ISO/IEC 9899:1999 standard.

imaxabs

Return absolute value

```
#include <inttypes.h>

intmax_t imaxabs(intmax_t j);
```

The *imaxabs()* function computes the absolute value of an integer *j*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

imaxdiv

Return quotient and remainder

```
#include <inttypes.h>

imaxdiv_t imaxdiv(intmax_t numer, intmax_t denom);
```

The *imaxdiv()* function computes *numer / denom* and *numer % denom* in a single operation.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

index

Character string operations (**LEGACY**)

```
xsi #include <strings.h>
char *index(const char *s, int c);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations.

The *strchr()* function is preferred over this function.

For maximum portability, it is recommended to replace the function call to *index()* as follows:

```
#define index(a,b) strchr((a),(b))
```

inet_addr, inet_ntoa

IPv4 address manipulation

```
#include <arpa/inet.h>

in_addr_t inet_addr(const char *cp);
char *inet_ntoa(struct in_addr in);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

inet_ntop, inet_pton

Convert IPv4 and IPv6 addresses between binary and text form

```
#include <arpa/inet.h>

const char *inet_ntop(int af, const void *restrict src,
    char *restrict dst, socklen_t size);
int inet_pton(int af, const char *restrict src, void *restrict dst);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 IPv6 extensions are marked.

The **restrict** keyword is added to the *inet_ntop()* and *inet_pton()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/29 is applied, adding “the address must be in network byte order” to the end of the fourth sentence of the first paragraph in the DESCRIPTION.

initstate, random, setstate, srandom

Pseudo-random number functions

```
xSI #include <stdlib.h>

char *initstate(unsigned seed, char *state, size_t size);
long random(void);
char *setstate(const char *state);
void srandom(unsigned seed);
```

Derivation First released in Issue 4, Version 2.

Issue 6 In the DESCRIPTION, duplicate text “For values greater than or equal to 8 ...” is removed.

insque, remque

Insert or remove an element in a queue

```
xSI #include <search.h>

void insque(void *element, void *pred);
void remque(void *element);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

ioctl

Control a STREAMS device (**STREAMS**)

```
XSR #include <stropts.h>
int ioctl(int fildes, int request, ... /* arg */);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The Open Group Corrigendum U028/4 is applied, correcting text in the I_FDINSERT, [EINVAL] case to refer to *ctlbuf*.

This function is marked as part of the XSI STREAMS Option Group and need not be supported on all implementations supporting the Single UNIX Specification.

isalnum

Test for an alphanumeric character

```
#include <ctype.h>
int isalnum(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isalpha

Test for an alphabetic character

```
#include <ctype.h>
int isalpha(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isascii

Test for a 7-bit US-ASCII character

```
XSI #include <ctype.h>
int isascii(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isastream

Test a file descriptor (**STREAMS**)

```
XSR #include <stropts.h>
int isastream(int fildes);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

isatty

Test for a terminal device

```
#include <unistd.h>
int isatty(int fildes);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional setting of *errno* to indicate an error is added.
- The [EBADF] and [ENOTTY] optional error conditions are added.

isblank

Test for a blank character

```
#include <ctype.h>
int isblank(int c);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

iscntrl

Test for a control character

```
#include <ctype.h>
int iscntrl(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isdigit

Test for a decimal digit

```
#include <ctype.h>
int isdigit(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isfinite

Test for finite value

```
#include <math.h>
int isfinite(real-floating x);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

isgraph

Test for a visible character

```
#include <ctype.h>
int isgraph(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isgreater

Test if x greater than y

```
#include <math.h>
int isgreater(real-floating x, real-floating y);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

isgreaterequal

Test if x greater than or equal to y

```
#include <math.h>
int isgreaterequal(real-floating x, real-floating y);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

isinf

Test for infinity

```
#include <math.h>
int isinf(real-floating x);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

isless

Test if x is less than y

```
#include <math.h>
int isless(real-floating x, real-floating y);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

islessequal

Test if x is less than or equal to y

```
#include <math.h>
```

```
int islessequal(real-floating x, real-floating y);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

islessgreater

Test if x is less than or greater than y

```
#include <math.h>
```

```
int islessgreater(real-floating x, real-floating y);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

islower

Test for a lowercase letter

```
#include <ctype.h>
```

```
int islower(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 An example is added.

isnan

Test for a NaN

```
#include <math.h>
```

```
int isnan(real-floating x);
```

Derivation First released in Issue 3.

Issue 6 Re-written for alignment with the ISO/IEC 9899: 1999 standard.

isnormal

Test for a normal value

```
#include <math.h>
```

```
int isnormal(real-floating x);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

isprint

Test for a printable character

```
#include <ctype.h>
```

```
int isprint(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

ispunct

Test for a punctuation character

```
#include <ctype.h>
int ispunct(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isspace

Test for a white-space character

```
#include <ctype.h>
int isspace(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

isunordered

Test if arguments are unordered

```
#include <math.h>
int isunordered(real-floating x, real-floating y);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/50 is applied, correcting the RETURN VALUE to be 1 when *x* or *y* is NaN.

isupper

Test for an uppercase letter

```
#include <ctype.h>
int isupper(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

iswalnum

Test for an alphanumeric wide-character code

```
#include <wctype.h>
int iswalnum(wint_t wc);
```

Derivation First released as a World-wide Portability Interface in Issue 4.

Issue 6 No functional changes are made in this issue.

iswalpha

Test for an alphabetic wide-character code

```
#include <wctype.h>
int iswalpha(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswblank

Test for a blank wide-character code

```
#include <wctype.h>
int iswblank(wint_t wc);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

iswcntrl

Test for a control wide-character code

```
#include <wctype.h>
int iswcntrl(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswctype

Test character for a specified class

```
#include <wctype.h>
int iswctype(wint_t wc, wctype_t charclass);
```

Derivation First released as World-wide Portability Interfaces in Issue 4.

Issue 6 The behavior of $n=0$ is now described.

An example is added.

A new function, *iswblank()*, is added to the list in the APPLICATION USAGE.

iswdigit

Test for a decimal digit wide-character code

```
#include <wctype.h>
int iswdigit(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswgraph

Test for a visible wide-character code

```
#include <wctype.h>
int iswgraph(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswlower

Test for a lowercase letter wide-character code

```
#include <wctype.h>
int iswlower(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswprint

Test for a printable wide-character code

```
#include <wctype.h>
int iswprint(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswpunct

Test for a punctuation wide-character code

```
#include <wctype.h>
int iswpunct(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswspace

Test for a white-space wide-character code

```
#include <wctype.h>
int iswspace(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswupper

Test for an uppercase letter wide-character code

```
#include <wctype.h>
int iswupper(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

iswxdigit

Test for a hexadecimal digit wide-character code

```
#include <wctype.h>
int iswxdigit(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

isxdigit

Test for a hexadecimal digit

```
#include <ctype.h>
int isxdigit(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

j0, j1, jn

Bessel functions of the first kind

```
xsi #include <math.h>
double j0(double x);
double j1(double x);
double jn(int n, double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The may fail [EDOM] error is removed for the case for NaN.

The RETURN VALUE and ERRORS sections are reworked for alignment of the error handling with the ISO/IEC 9899:1999 standard.

kill

Send a signal to a process or a group of processes

```
cx #include <signal.h>
int kill(pid_t pid, int sig);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.
The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the second paragraph is reworded to indicate that the saved set-user-ID of the calling process is checked in place of its effective user ID. This is a FIPS requirement.
- The behavior when *pid* is `-1` is now specified. It was previously explicitly unspecified in IEEE Std 1003.1-1988 (POSIX.1). This may require a change for systems that have previously implemented the BSD semantics.

killpg

Send a signal to a process group

```
xSI #include <signal.h>
int killpg(pid_t pgrp, int sig);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

labs, llabs

Return a long integer absolute value

```
#include <stdlib.h>
long labs(long i);
long long llabs(long long i);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 The `llabs()` function is added for alignment with the ISO/IEC 9899:1999 standard.

lchown

Change the owner and group of a symbolic link

```
xSI #include <unistd.h>
int lchown(const char *path, uid_t owner, gid_t group);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

ldexp, ldexpf, ldexpl

Load exponent of a floating-point number

```
#include <math.h>

double ldexp(double x, int exp);
float ldexpf(float x, int exp);
long double ldexpl(long double x, int exp);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *ldexpf()* and *ldexpl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

ldiv, lldiv

Compute quotient and remainder of a long division

```
#include <stdlib.h>

ldiv_t ldiv(long numer, long denom);
lldiv_t lldiv(long long numer, long long denom);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 The *lldiv()* function is added for alignment with the ISO/IEC 9899:1999 standard.

lgamma, lgammaf, lgammal

Log gamma function

```
#include <math.h>

double lgamma(double x);
float lgammaf(float x);
long double lgammal(long double x);
extern int signgam;
```

xSI

Derivation First released in Issue 3.

Issue 6 The *lgamma()* function is no longer marked as an extension.

The *lgammaf()* and *lgammal()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

link

Link to a file

```
#include <unistd.h>

int link(const char *path1, const char *path2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added of action when *path2* refers to a symbolic link.
- The [ELOOP] optional error condition is added.

lio_listio

List directed I/O (**REALTIME**)

AIO

```
#include <aio.h>

int lio_listio(int mode, struct aiocb *restrict const list[restrict],
               int nent, struct sigevent *restrict sig);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

The *lio_listio()* function is marked as part of the Asynchronous Input and Output option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate that for regular files no data transfer occurs past the offset maximum established in the open file description associated with *aiocbp->aio_fildes*. This change is to support large files.
- The [EBIG] and [EOVERFLOW] error conditions are defined. This change is to support large files.

The **restrict** keyword is added to the *lio_listio()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/54 is applied, adding text to the DESCRIPTION making it explicit that the user is required to keep the structure pointed to by *sig->sigev_notify_attributes* valid until the last asynchronous operation finished and the notification has been sent.

listen

Listen for socket connections and limit the queue of incoming connections

```
#include <sys/socket.h>

int listen(int socket, int backlog);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 The DESCRIPTION is updated to describe the relationship of SOMAXCONN and the *backlog* argument.

llrint, llrintf, llrintl

Round to nearest integer value using current rounding direction

```
#include <math.h>

long long llrint(double x);
long long llrintf(float x);
long long llrintl(long double x);
```

These functions round their argument to the nearest integer value, rounding according to the current rounding direction.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

llround, llroundf, llroundl

Round to nearest integer value

```
#include <math.h>

long long llround(double x);
long long llroundf(float x);
long long llroundl(long double x);
```

These functions round their argument to the nearest integer value, rounding halfway cases away from zero, regardless of the current rounding direction.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

localeconv

Return locale-specific information

```
#include <locale.h>

struct lconv *localeconv(void);
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

This reference page is updated for alignment with the ISO/IEC 9899: 1999 standard.

ISO/IEC 9899: 1999 standard, Technical Corrigendum No. 1 is incorporated.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/31 is applied, removing references to **int_curr_symbol** and updating the descriptions of **p_sep_by_space**

and `n_sep_by_space`. These changes are for alignment with the ISO C standard.

localtime, localtime_r

Convert a time value to a broken-down local time

```
#include <time.h>
```

```
struct tm *localtime(const time_t *timer);
TSF struct tm *localtime_r(const time_t *restrict timer,
    struct tm *restrict result);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `localtime_r()` function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the `localtime_r()` prototype for alignment with the ISO/IEC 9899:1999 standard.

Examples are added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/32 is applied, adding the [Eoverflow] error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/56 is applied, adding a requirement that if `localtime_r()` does not set the `tzname` variable, it shall not set the `daylight` or `timezone` variables. On systems supporting XSI, the `daylight`, `timezone`, and `tzname` variables should all be set to provide information for the same timezone. This updates the description of `localtime_r()` to mention `daylight` and `timezone` as well as `tzname`.

lockf

Record locking on files

```
XSI #include <unistd.h>
int lockf(int fildes, int function, off_t size);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

log, logf, logl

Natural logarithm function

```
#include <math.h>
```

```
double log(double x);
float logf(float x);
long double logl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *logf()* and *logl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

log10, log10f, log10l

Base 10 logarithm function

```
#include <math.h>

double log10(double x);
float log10f(float x);
long double log10l(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *log10f()* and *log10l()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

log1p, log1pf, log1pl

Compute a natural logarithm

```
#include <math.h>

double log1p(double x);
float log1pf(float x);
long double log1pl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *log1p()* function is no longer marked as an extension.

The *log1pf()* and *log1pl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

log2, log2f, log2l

Compute base 2 logarithm functions

```
#include <math.h>

double log2(double x);
float log2f(float x);
long double log2l(long double x);
```

These functions compute the base-2 logarithm of their argument x , $\log_2(x)$.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

logb, logbf, logbl

Radix-independent exponent

```
#include <math.h>

double logb(double x);
float logbf(float x);
long double logbl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *logb()* function is no longer marked as an extension.

The *logbf()* and *logbl()* functions are added for alignment with the ISO/IEC 9899: 1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899: 1999 standard.

IEC 60559: 1989 standard floating-point extensions over the ISO/IEC 9899: 1999 standard are marked.

longjmp

Non-local goto

```
#include <setjmp.h>

void longjmp(jmp_buf env, int val);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION now explicitly makes *longjmp()*'s effect on the signal mask unspecified.

The DESCRIPTION is updated for alignment with the ISO/IEC 9899: 1999 standard.

lrint, lrintf, lrintl

Round to nearest integer value using current rounding direction

```
#include <math.h>

long lrint(double x);
long lrintf(float x);
long lrintl(long double x);
```

These functions round their argument to the nearest integer value, rounding according to the current rounding direction.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

lround, lroundf, lroundl

Round to nearest integer value

```
#include <math.h>

long lround(double x);
long lroundf(float x);
long lroundl(long double x);
```

These functions round their argument to the nearest integer value, rounding halfway cases away from zero, regardless of the current rounding direction.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

lsearch, lfind

Linear search and update

```
xsl #include <search.h>

void *lsearch(const void *key, void *base, size_t *nelp, size_t width,
             int (*compar)(const void *, const void *));
void *lfind(const void *key, const void *base, size_t *nelp,
            size_t width, int (*compar)(const void *, const void *));
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

lseek

Move the read/write file offset

```
#include <unistd.h>

off_t lseek(int fildes, off_t offset, int whence);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EOVERFLOW] error condition is added. This change is to support large files.

An additional [ESPIPE] error condition is added for sockets.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *lseek()* results are unspecified for typed memory objects.

lstat

Get symbolic link status

```
#include <sys/stat.h>
```

```
int lstat(const char *restrict path, struct stat *restrict buf);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The following changes were made to align with the IEEE P1003.1a draft standard:

- This function is now mandatory.
- The [ELOOP] optional error condition is added.

The **restrict** keyword is added to the *lstat()* prototype for alignment with the ISO/IEC 9899:1999 standard.

makecontext, swapcontext

Manipulate user contexts

```
OB XSI #include <ucontext.h>
```

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

```
int swapcontext(ucontext_t *restrict oucp,
               const ucontext_t *restrict ucp);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The **restrict** keyword is added to the *swapcontext()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/57 is applied, obsoleting the functions. The prototype makes use of an obsolescent feature of the ISO C standard. Therefore, a strictly conforming POSIX application cannot use this form, and thus, use of *getcontext()*, *makecontext()*, and *swapcontext()* is marked obsolescent. It is recommended that applications that use these functions be converted to use threads instead.

malloc

A memory allocator

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

```
void *malloc(size_t size);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* to indicate an error is added.

- The [ENOMEM] error condition is added.

mblen

Get number of bytes in a character

```
#include <stdlib.h>

int mblen(const char *s, size_t n);
```

Derivation First released in Issue 4. Aligned with the ISO C standard.

Issue 6 No functional changes are made in this issue.

mbrlen

Get number of bytes in a character (restartable)

```
#include <wchar.h>

size_t mbrlen(const char *restrict s, size_t n,
              mbstate_t *restrict ps);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The *mbrlen()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

mbrtowc

Convert a character to a wide-character code (restartable)

```
#include <wchar.h>

size_t mbrtowc(wchar_t *restrict pwc, const char *restrict s,
              size_t n, mbstate_t *restrict ps);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The *mbrtowc()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] error condition is added.

ISO/IEC 9899:1999 standard, Technical Corrigendum No. 1 is incorporated.

mbsinit

Determine conversion object status

```
#include <wchar.h>

int mbsinit(const mbstate_t *ps);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

mbsrtowcs

Convert a character string to a wide-character string (restartable)

```
#include <wchar.h>

size_t mbsrtowcs(wchar_t *restrict dst, const char **restrict src,
                 size_t len, mbstate_t *restrict ps);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The *mbsrtowcs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

The [EINVAL] error condition is marked CX.

mbstowcs

Convert a character string to a wide-character string

```
#include <stdlib.h>

size_t mbstowcs(wchar_t *restrict pwcs, const char *restrict s,
                 size_t n);
```

Derivation First released in Issue 4. Aligned with the ISO C standard.

Issue 6 The *mbstowcs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

mbtowc

Convert a character to a wide-character code

```
#include <stdlib.h>

int mbtowc(wchar_t *restrict pwc, const char *restrict s, size_t n);
```

Derivation First released in Issue 4. Aligned with the ISO C standard.

Issue 6 The *mbtowc()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

memccpy

Copy bytes in memory

```
xSI #include <string.h>

void *memccpy(void *restrict s1, const void *restrict s2,
              int c, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The **restrict** keyword is added to the *memccpy()* prototype for alignment with the ISO/IEC 9899:1999 standard.

memchr

Find byte in memory

```
#include <string.h>
```

```
void *memchr(const void *s, int c, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

memcmp

Compare bytes in memory

```
#include <string.h>
```

```
int memcmp(const void *s1, const void *s2, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

memcpy

Copy bytes in memory

```
#include <string.h>
```

```
void *memcpy(void *restrict s1, const void *restrict s2, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *memcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

memmove

Copy bytes in memory with overlapping areas

```
#include <string.h>
```

```
void *memmove(void *s1, const void *s2, size_t n);
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 No functional changes are made in this issue.

memset

Set bytes in memory

```
#include <string.h>
```

```
void *memset(void *s, int c, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

mkdir

Make a directory

```
#include <sys/stat.h>

int mkdir(const char *path, mode_t mode);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed. The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

mkfifo

Make a FIFO special file

```
#include <sys/stat.h>

int mkfifo(const char *path, mode_t mode);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed. The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

mknod

Make a directory, a special file, or a regular file

```
xsi #include <sys/stat.h>

int mknod(const char *path, mode_t mode, dev_t dev);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

mkstemp

Make a unique filename

```
xSI #include <stdlib.h>
int mkstemp(char *template);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

mktemp

Make a unique filename (**LEGACY**)

```
xSI #include <stdlib.h>
char *mktemp(char *template);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations. Between the time a pathname is created and the file opened, it is possible for some other process to create a file with the same name. The *mkstemp()* function avoids this problem and is preferred over this function.

mktime

Convert broken-down time into time since the Epoch

```
#include <time.h>
time_t mktime(struct tm *timeptr);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1) and ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/58 is applied, updating the RETURN VALUE and ERRORS sections to add the optional [E_OVERFLOW] error as a CX extension.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/59 is applied, adding the *tzset()* function to the SEE ALSO section.

mlock, munlock

Lock or unlock a range of process address space (**REALTIME**)

```
MLR #include <sys/mman.h>
int mlock(const void *addr, size_t len);
int munlock(const void *addr, size_t len);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mlock()* and *munlock()* functions are marked as part of the Range Memory Locking option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Range Memory Locking option.

mlockall, munlockall

Lock/unlock the address space of a process (**REALTIME**)

```
ML #include <sys/mman.h>

int mlockall(int flags);
int munlockall(void);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mlockall()* and *munlockall()* functions are marked as part of the Process Memory Locking option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Memory Locking option.

mmap

Map pages of memory

```
MC3 #include <sys/mman.h>

void *mmap(void *addr, size_t len, int prot, int flags,
           int fildes, off_t off);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *mmap()* function is marked as part of the Memory Mapped Files, Shared Memory Objects option and Typed Memory Objects option.

The Open Group Corrigendum U028/6 is applied, changing (**void ***)–1 to MAP_FAILED.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to describe the use of MAP_FIXED.
- The DESCRIPTION is updated to describe the addition of an extra reference to the file associated with the file descriptor passed to *mmap()*.
- The DESCRIPTION is updated to state that there may be implementation-defined limits on the number of memory regions that can be mapped.
- The DESCRIPTION is updated to describe constraints on the alignment and size of the *off* argument.
- The [EINVAL] and [EMFILE] error conditions are added.
- The [EOVERFLOW] error condition is added. This change is to support large files.

The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The DESCRIPTION is updated to describe the cases when MAP_PRIVATE and MAP_FIXED need not be supported.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- Semantics for typed memory objects are added to the DESCRIPTION.
- New [ENOMEM] and [ENXIO] errors are added to the ERRORS section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/60 is applied, updating the DESCRIPTION and ERRORS sections to add the [EINVAL] error when *len* is zero.

modf, modff, modfl

Decompose a floating-point number

```
#include <math.h>
```

```
double modf(double x, double *iptr);  
float modff(float value, float *iptr);  
long double modfl(long double value, long double *iptr);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *modff()* and *modfl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

mprotect

Set protection of memory mapping

MPR

```
#include <sys/mman.h>
```

```
int mprotect(void *addr, size_t len, int prot);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *mprotect()* function is marked as part of the Memory Protection option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to state that implementations require *addr* to be a multiple of the page size as returned by *sysconf()*.
- The [EINVAL] error condition is added.

mq_closeClose a message queue (**REALTIME**)

MSG

```
#include <mqueue.h>
int mq_close(mqd_t mqdes);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mq_close()* function is marked as part of the Message Passing option. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

mq_getattrGet message queue attributes (**REALTIME**)

MSG

```
#include <mqueue.h>
int mq_getattr(mqd_t mqdes, struct mq_attr *mqstat);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mq_getattr()* function is marked as part of the Message Passing option. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option. IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/61 is applied, updating the ERRORS section to change the [EBADF] error from mandatory to optional.

mq_notifyNotify process that a message is available (**REALTIME**)

MSG

```
#include <mqueue.h>
int mq_notify(mqd_t mqdes, const struct sigevent *notification);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mq_notify()* function is marked as part of the Message Passing option. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

mq_open

Open a message queue (**REALTIME**)

```
MSG #include <mqueue.h>
mqd_t mq_open(const char *name, int oflag, ...);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mq_open()* function is marked as part of the Message Passing option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

The DESCRIPTION of O_EXCL is updated in response to IEEE PASC Interpretation 1003.1c #48.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/62 is applied, updating the description of the permission bits in the DESCRIPTION section. The change is made for consistency with the *shm_open()* and *sem_open()* functions.

mq_receive, mq_timedreceive

Receive a message from a message queue (**REALTIME**)

```
MSG #include <mqueue.h>
ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len,
    unsigned *msg_prio);
```

```
MSG TMO #include <mqueue.h>
#include <time.h>
ssize_t mq_timedreceive(mqd_t mqdes, char *restrict msg_ptr,
    size_t msg_len, unsigned *restrict msg_prio,
    const struct timespec *restrict abs_timeout);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mq_receive()* function is marked as part of the Message Passing option.

The Open Group Corrigendum U021/4 is applied. The DESCRIPTION is changed to refer to *msg_len* rather than *maxsize*.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In this function it is possible for the return value to exceed the range of the type **ssize_t** (since **size_t** has a larger range of positive values than **ssize_t**). A sentence restricting the size of the **size_t** object is added to the description to resolve this conflict.

The `mq_timedreceive()` function is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the `mq_timedreceive()` prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE PASC Interpretation 1003.1 #109 is applied, correcting the return type for `mq_timedreceive()` from **int** to **ssize_t**.

mq_send, mq_timedsend

Send a message to a message queue (**REALTIME**)

```
MSG #include <mqqueue.h>
int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
            unsigned msg_prio);
```

```
MSG TMO #include <mqqueue.h>
#include <time.h>
int mq_timedsend(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
                unsigned msg_prio, const struct timespec *abs_timeout);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The `mq_send()` function is marked as part of the Message Passing option. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option. The `mq_timedsend()` function is added for alignment with IEEE Std 1003.1d-1999.

mq_setattr

Set message queue attributes (**REALTIME**)

```
MSG #include <mqqueue.h>
int mq_setattr(mqd_t mqdes, const struct mq_attr *restrict mqstat,
              struct mq_attr *restrict omqstat);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The `mq_setattr()` function is marked as part of the Message Passing option. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option. The **restrict** keyword is added to the `mq_setattr()` prototype for alignment with the ISO/IEC 9899:1999 standard.

mq_unlink

Remove a message queue (**REALTIME**)

```
MSG #include <mqueue.h>
int mq_unlink(const char *name);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *mq_unlink()* function is marked as part of the Message Passing option. The Open Group Corrigendum U021/5 is applied, clarifying that upon unsuccessful completion, the named message queue is unchanged by this function. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

msgctl

XSI message control operations

```
XSI #include <sys/msg.h>
int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

msgget

Get the XSI message queue identifier

```
XSI #include <sys/msg.h>
int msgget(key_t key, int msgflg);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

msgrcv

XSI message receive operation

```
XSI #include <sys/msg.h>
ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp,
int msgflg);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

msgsnd

XSI message send operation

```
XSI #include <sys/msg.h>
int msgsnd(int msgid, const void *msgp, size_t msgsz, int msgflg);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

msync

Synchronize memory with physical storage

```
MF SIO #include <sys/mman.h>
int msync(void *addr, size_t len, int flags);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *msync()* function is marked as part of the Memory Mapped Files and Synchronized Input and Output options.

The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The [EBUSY] mandatory error condition is added.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to state that implementations require *addr* to be a multiple of the page size.
- The second [EINVAL] error condition is made mandatory.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding reference to typed memory objects.

munmap

Unmap pages of memory

```
MC3 #include <sys/mman.h>
int munmap(void *addr, size_t len);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *munmap()* function is marked as part of the Memory Mapped Files, Shared Memory Objects option and Typed Memory Objects option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to state that implementations require *addr* to be a multiple of the page size.

- The [EINVAL] error conditions are added.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- Semantics for typed memory objects are added to the DESCRIPTION.

nan, nanf, nanl

Return quiet NaN

```
#include <math.h>

double nan(const char *tagp);
float nanf(const char *tagp);
long double nanl(const char *tagp);
```

The function call *nan("n-char-sequence")* is equivalent to:

```
strtod("NaN(n-char-sequence)", (char **) NULL);
```

The function call *nan("")* is equivalent to:

```
strtod("NaN()", (char **) NULL)
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

nanosleep

High resolution sleep (**REALTIME**)

TMR

```
#include <time.h>

int nanosleep(const struct timespec *rqtp, struct timespec *rmtp);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *nanosleep()* function is marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

nearbyint, nearbyintf, nearbyintl

Floating-point rounding functions

```
#include <math.h>

double nearbyint(double x);
float nearbyintf(float x);
long double nearbyintl(long double x);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

nextafter, nextafterf, nextafterl, nexttoward, nexttowardf, nexttowardl

Next representable floating-point number

```
#include <math.h>

double nextafter(double x, double y);
float nextafterf(float x, float y);
long double nextafterl(long double x, long double y);
double nexttoward(double x, long double y);
float nexttowardf(float x, long double y);
long double nexttowardl(long double x, long double y);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *nextafter()* function is no longer marked as an extension.

The *nextafterf()*, *nextafterl()*, *nexttoward()*, *nexttowardf()*, *nexttowardl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

nftw

Walk a file tree

```
xsi #include <ftw.h>

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

Derivation First released in Issue 4, Version 2.

Issue 6 The Open Group Base Resolution bwg97-003 is applied.

The ERRORS section is updated as follows:

- The wording of the mandatory [ELOOP] error condition is updated.
- A second optional [ELOOP] error condition is added.
- The [EOVERFLOW] mandatory error condition is added.

Text is added to the DESCRIPTION to say that the *nftw()* function need not be reentrant and that the results are unspecified if the application-supplied *fn* function does not preserve the current working directory.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/64 is applied, changing the argument *depth* to *fd_limit* throughout and changing “to a maximum of 5 levels deep” to “using a maximum of 5 file descriptors” in the EXAMPLES section.

nice

Change the nice value of a process

```
XSI #include <unistd.h>
int nice(int incr);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

nl_langinfo

Language information

```
XSI #include <langinfo.h>
char *nl_langinfo(nl_item item);
```

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

open

Open a file

```
OH #include <sys/stat.h>
#include <fcntl.h>
int open(const char *path, int oflag, ... );
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, O_CREAT is amended to state that the group ID of the file is set to the group ID of the file's parent directory or to the effective group ID of the process. This is a FIPS requirement.
- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
- The [ENXIO] mandatory error condition is added.
- The [EINVAL], [ENAMETOOLONG], and [ETXTBSY] optional error conditions are added.

The DESCRIPTION and ERRORS sections are updated so that items related to the optional XSI STREAMS Option Group are marked.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added of the effect of the O_CREAT and O_EXCL flags when the path refers to a symbolic link.

- The [ELOOP] optional error condition is added.

The DESCRIPTION of O_EXCL is updated in response to IEEE PASC Interpretation 1003.1c #48.

opendir

Open a directory

```
#include <dirent.h>
```

```
DIR *opendir(const char *dirname);
```

Derivation First released in Issue 2.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

pause

Suspend the thread until a signal is received

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

```
int pause(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

pclose

Close a pipe stream to or from a process

```
cx #include <stdio.h>
```

```
int pclose(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

perror

Write error messages to standard error

```
#include <stdio.h>
```

```
void perror(const char *s);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

pipe

Create an interprocess channel

```
#include <unistd.h>
int pipe(int fildes[2]);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to indicate that certain dispositions of *fildes*[0] and *fildes*[1] are unspecified.

poll

Input/output multiplexing

```
XSI #include <poll.h>
int poll(struct pollfd fds[], nfd_t nfd, int timeout);
```

Derivation First released in Issue 4, Version 2.

Issue 6 Text referring to sockets is added to the DESCRIPTION.

Text relating to the XSI STREAMS Option Group is marked.

The Open Group Corrigendum U055/3 is applied, updating the DESCRIPTION of POLLWRBAND from “This event only examines bands that have been written to at least once.” to “If any priority band has been written to on this STREAM, this event only examines bands that have been written to at least once.”

popen

Initiate pipe streams to or from a process

```
CX #include <stdio.h>
FILE *popen(const char *command, const char *mode);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional [EMFILE] error condition is added.

posix_fadvise

File advisory information (**ADVANCED REALTIME**)

```
ADV #include <fcntl.h>
int posix_fadvise(int fd, off_t offset, off_t len, int advice);
```

The *posix_fadvise*() function advises the implementation on the expected behavior of the application with respect to the data in the file associated with the open file descriptor, *fd*, starting at *offset* and continuing for *len* bytes.

Implementations may use information conveyed by a previous *posix_fadvise()* call to influence the manner in which allocation is performed. For example, if an application did the following calls:

```
fd = open("file");
posix_fadvise(fd, offset, len, POSIX_FADV_SEQUENTIAL);
posix_fallocate(fd, len, size);
```

an implementation might allocate the file contiguously on disk.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/68 is applied, changing the function prototype in the SYNOPSIS section. The previous prototype was not large-file aware, and the standard developers felt it acceptable to make this change before implementations of the functions become widespread.

posix_fallocate

File space control (**ADVANCED REALTIME**)

ADV

```
#include <fcntl.h>

int posix_fallocate(int fd, off_t offset, off_t len);
```

The *posix_fallocate()* function can be used to guarantee no [ENOSPC] errors and to improve performance by prepaying any overhead required for block allocation.

The *posix_fallocate()* function ensures that any required storage for regular file data starting at *offset* and continuing for *len* bytes is allocated on the file system storage media. If the function returns successfully, subsequent writes to the specified file data will not fail due to the lack of free space on the file system storage media.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/69 is applied, changing the function prototype in the SYNOPSIS section. The previous prototype was not large-file aware, and the standard developers felt it acceptable to make this change before implementations of the functions become widespread.

posix_madvise

Memory advisory information and alignment control (**ADVANCED REALTIME**)

ADV

```
#include <sys/mman.h>

int posix_madvise(void *addr, size_t len, int advice);
```

The *posix_madvise()* function advises the implementation on the expected behavior of the application with respect to the data in the memory starting at address *addr*, and continuing for *len* bytes.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #102 is applied, clarifying that this function should be declared in **<sys/mman.h>** and not **<fcntl.h>**.

posix_mem_offset

Find offset and length of a mapped typed memory block (**ADVANCED REALTIME**)

TYM

```
#include <sys/mman.h>
```

```
int posix_mem_offset(const void *restrict addr, size_t len,  
off_t *restrict off, size_t *restrict contig_len,  
int *restrict fildes);
```

The *posix_mem_offset()* function returns in the variable pointed to by *off* a value that identifies the offset (or location), within a memory object, of the memory block currently mapped at *addr*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_memalign

Aligned memory allocation (**ADVANCED REALTIME**)

ADV

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

```
int posix_memalign(void **memptr, size_t alignment, size_t size);
```

The *posix_memalign()* function was added to allow for the allocation of specifically aligned buffers.

The *posix_memalign()* function allocates *size* bytes aligned on a boundary specified by *alignment*, and returns a pointer to the allocated memory in *memptr*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

posix_openpt

Open a pseudo-terminal device

XSI

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

```
int posix_openpt(int oflag);
```

This function is a method for portably obtaining a file descriptor of a master terminal device for a pseudo-terminal. The *grantpt()* and *ptsname()* functions can be used to manipulate mode and ownership permissions and to obtain the name of the slave device, respectively.

The standard developers considered the matter of adding a special device for cloning master pseudo-terminals: the `/dev/ptmx` device. However, consensus could not be reached, and it was felt that adding a new function would permit other implementations.

An example that opens a pseudo-terminal and returns the name of the slave device and a file descriptor is given below:

```
#include <fcntl.h>
#include <stdio.h>

int masterfd, slavefd;
char *slavedevice;

masterfd = posix_openpt(O_RDWR|O_NOCTTY);

if (masterfd == -1
    || grantpt (masterfd) == -1
    || unlockpt (masterfd) == -1
    || (slavedevice = ptsname (masterfd)) == NULL)
    return -1;

printf("slave device is: %s\n", slavedevice);

slavefd = open(slave, O_RDWR|O_NOCTTY);
if (slavefd < 0)
    return -1;
```

Derivation First released in Issue 6.

posix_spawn, posix_spawnp

Spawn a process (**ADVANCED REALTIME**)

SPN

```
#include <spawn.h>

int posix_spawn(pid_t *restrict pid, const char *restrict path,
               const posix_spawn_file_actions_t *file_actions,
               const posix_spawnattr_t *restrict attrp,
               char *const argv[restrict], char *const envp[restrict]);
int posix_spawnp(pid_t *restrict pid, const char *restrict file,
                const posix_spawn_file_actions_t *file_actions,
                const posix_spawnattr_t *restrict attrp,
                char *const argv[restrict], char *const envp[restrict]);
```

The `posix_spawn()` and `posix_spawnp()` functions create a new process (child process) from the specified process image.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #103 is applied, noting that the signal default actions are changed as well as the signal mask in step 2.

IEEE PASC Interpretation 1003.1 #132 is applied. This addresses the issue of whether or not an implementation allows `SIG_IGN` as a `SIGCHLD` disposition to be inherited across a call to one of the `exec` family of functions or `posix_spawn()`, leaving it explicitly unspecified.

posix_spawn_file_actions_addclose, posix_spawn_file_actions_addopen

Add close or open action to spawn file actions object (**ADVANCED REALTIME**)

```
SPN #include <spawn.h>

int posix_spawn_file_actions_addclose(posix_spawn_file_actions_t *
    file_actions, int fildes);
int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t *restrict
    file_actions, int fildes, const char *restrict path,
    int oflag, mode_t mode);
```

These functions add or delete a close or open action to a spawn file actions object.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #105 is applied, adding a note to the DESCRIPTION that the string pointed to by *path* is copied by the *posix_spawn_file_actions_addopen()* function.

posix_spawn_file_actions_adddup2

Add dup2 action to spawn file actions object (**ADVANCED REALTIME**)

```
SPN #include <spawn.h>

int posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t *
    file_actions, int fildes, int newfildes);
```

The *posix_spawn_file_actions_adddup2()* function adds a *dup2()* action to the object referenced by *file_actions* that causes the file descriptor *fildes* to be duplicated as *newfildes* (as if *dup2(fildes, newfildes)* had been called) when a new process is spawned using this file actions object.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #104 is applied, noting that the [EBADF] error can apply to the *newfildes* argument in addition to *fildes*.

posix_spawn_file_actions_destroy, posix_spawn_file_actions_init

Destroy and initialize spawn file actions object (**ADVANCED REALTIME**)

```
SPN #include <spawn.h>

int posix_spawn_file_actions_destroy(posix_spawn_file_actions_t *
    file_actions);
int posix_spawn_file_actions_init(posix_spawn_file_actions_t *
    file_actions);
```

The *posix_spawn_file_actions_destroy()* function destroys the object referenced by *file_actions*; the object becomes, in effect, uninitialized.

The `posix_spawn_file_actions_init()` function initializes the object referenced by `file_actions` to contain no file actions for `posix_spawn()` or `posix_spawnnp()` to perform.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

posix_spawnattr_destroy, posix_spawnattr_init

Destroy and initialize spawn attributes object (**ADVANCED REALTIME**)

```
SPN #include <spawn.h>

int posix_spawnattr_destroy(posix_spawnattr_t *attr);
int posix_spawnattr_init(posix_spawnattr_t *attr);
```

The `posix_spawnattr_destroy()` function destroys a spawn attributes object.

The `posix_spawnattr_init()` function initializes a spawn attributes object `attr` with the default value for all of the individual attributes used by the implementation.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #106 is applied, noting that the effect of initializing an already initialized spawn attributes option is undefined.

posix_spawnattr_getflags, posix_spawnattr_setflags

Get and set spawn-flags attribute of spawn attributes object (**ADVANCED REALTIME**)

```
SPN #include <spawn.h>

int posix_spawnattr_getflags(const posix_spawnattr_t *restrict attr,
                             short *restrict flags);
int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
```

The `posix_spawnattr_getflags()` function obtains the value of the `spawn-flags` attribute from the attributes object referenced by `attr`.

The `posix_spawnattr_setflags()` function sets the `spawn-flags` attribute in an initialized attributes object referenced by `attr`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_spawnattr_getpgroup, posix_spawnattr_setpgroup

Get and set spawn-pgroup attribute of spawn attributes object (**ADVANCED REALTIME**)

```
SPN #include <spawn.h>

int posix_spawnattr_getpgroup(const posix_spawnattr_t *restrict attr,
    pid_t *restrict pgroup);
int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);
```

The *posix_spawnattr_getpgroup()* function obtains the value of the *spawn-pgroup* attribute from the attributes object referenced by *attr*.

The *posix_spawnattr_setpgroup()* function sets the *spawn-pgroup* attribute in an initialized attributes object referenced by *attr*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_spawnattr_getschedparam, posix_spawnattr_setschedparam

Get and set spawn-schedparam attribute of spawn attributes object (**ADVANCED REALTIME**)

```
SPN PS #include <spawn.h>
#include <sched.h>

int posix_spawnattr_getschedparam(
    const posix_spawnattr_t *restrict attr,
    struct sched_param *restrict schedparam);
int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict attr,
    const struct sched_param *restrict schedparam);
```

The *posix_spawnattr_getschedparam()* function obtains the value of the *spawn-schedparam* attribute from the attributes object referenced by *attr*.

The *posix_spawnattr_setschedparam()* function sets the *spawn-schedparam* attribute in an initialized attributes object referenced by *attr*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_spawnattr_getschedpolicy, posix_spawnattr_setschedpolicy

Get and set spawn-schedpolicy attribute of spawn attributes object (**ADVANCED REALTIME**)

```
SPN PS #include <spawn.h>
#include <sched.h>

int posix_spawnattr_getschedpolicy(
    const posix_spawnattr_t *restrict attr,
    int *restrict schedpolicy);
int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
    int schedpolicy);
```

The `posix_spawnattr_getschedpolicy()` function obtains the value of the `spawn-schedpolicy` attribute from the attributes object referenced by `attr`.

The `posix_spawnattr_setschedpolicy()` function sets the `spawn-schedpolicy` attribute in an initialized attributes object referenced by `attr`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_spawnattr_getsigdefault, posix_spawnattr_setsigdefault

Get and set spawn-sigdefault attribute of spawn attributes object (**ADVANCED REALTIME**)

```
SPN #include <signal.h>
#include <spawn.h>

int posix_spawnattr_getsigdefault(
    const posix_spawnattr_t *restrict attr,
    sigset_t *restrict sigdefault);
int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict attr,
    const sigset_t *restrict sigdefault);
```

The `posix_spawnattr_getsigdefault()` function obtains the value of the `spawn-sigdefault` attribute from the attributes object referenced by `attr`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_spawnattr_getsigmask, posix_spawnattr_setsigmask

Get and set spawn-sigmask attribute of spawn attributes object (**ADVANCED REALTIME**)

```
SPN #include <signal.h>
#include <spawn.h>

int posix_spawnattr_getsigmask(const posix_spawnattr_t *restrict attr,
    sigset_t *restrict sigmask);
int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict attr,
    const sigset_t *restrict sigmask);
```

The `posix_spawnattr_getsigmask()` function obtains the value of the `spawn-sigmask` attribute from the attributes object referenced by `attr`.

The `posix_spawnattr_setsigmask()` function sets the `spawn-sigmask` attribute in an initialized attributes object referenced by `attr`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_trace_attr_destroy, posix_trace_attr_init

Trace stream attributes object destroy and initialization (**TRACING**)

```
TRC #include <trace.h>

int posix_trace_attr_destroy(trace_attr_t *attr);
int posix_trace_attr_init(trace_attr_t *attr);
```

The *posix_trace_attr_destroy()* function destroys an initialized trace attributes object.

The *posix_trace_attr_init()* function initializes a trace attributes object *attr* with the default value for all of the individual attributes used by a given implementation.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

posix_trace_attr_getclockres, posix_trace_attr_getcreatetime, posix_trace_attr_getgenversion, posix_trace_attr_getname, posix_trace_attr_setname

Retrieve and set information about a trace stream (**TRACING**)

```
TRC #include <time.h>
#include <trace.h>

int posix_trace_attr_getclockres(const trace_attr_t *attr,
    struct timespec *resolution);
int posix_trace_attr_getcreatetime(const trace_attr_t *attr,
    struct timespec *createtime);

#include <trace.h>

int posix_trace_attr_getgenversion(const trace_attr_t *attr,
    char *genversion);
int posix_trace_attr_getname(const trace_attr_t *attr,
    char *tracename);
int posix_trace_attr_setname(trace_attr_t *attr,
    const char *tracename);
```

The *posix_trace_attr_getclockres()* function copies the clock resolution of the clock used to generate timestamps from the *clock-resolution* attribute of the attributes object pointed to by the *attr* argument into the structure pointed to by the *resolution* argument.

The *posix_trace_attr_getcreatetime()* function copies the trace stream creation time from the *creation-time* attribute of the attributes object pointed to by the *attr* argument into the structure pointed to by the *createtime* argument.

The *posix_trace_attr_getgenversion()* function copies the string containing version information from the *generation-version* attribute of the attributes object pointed to by the *attr* argument into the string pointed to by the *genversion* argument.

The *posix_trace_attr_getname()* function copies the string containing the trace name from the *trace-name* attribute of the attributes object pointed to by the *attr* argument into the string pointed to by the *tracename* argument.

The `posix_trace_attr_setname()` function sets the name in the `trace-name` attribute of the attributes object pointed to by the `attr` argument, using the trace name string supplied by the `tracename` argument.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

**posix_trace_attr_getinherited, posix_trace_attr_getlogfullpolicy,
posix_trace_attr_getstreamfullpolicy, posix_trace_attr_setinherited,
posix_trace_attr_setlogfullpolicy, posix_trace_attr_setstreamfullpolicy**

Retrieve and set the behavior of a trace stream (**TRACING**)

```
TRC #include <trace.h>
TRC TRI int posix_trace_attr_getinherited(const trace_attr_t *restrict attr,
TRC     int *restrict inheritancepolicy);
TRC TRL int posix_trace_attr_getlogfullpolicy(const trace_attr_t *restrict attr,
TRC     int *restrict logpolicy);
TRC     int posix_trace_attr_getstreamfullpolicy(const trace_attr_t *attr,
TRC     int *streampolicy);
TRC TRI int posix_trace_attr_setinherited(trace_attr_t *attr,
TRC     int inheritancepolicy);
TRC TRL int posix_trace_attr_setlogfullpolicy(trace_attr_t *attr,
TRC     int logpolicy);
TRC     int posix_trace_attr_setstreamfullpolicy(trace_attr_t *attr,
TRC     int streampolicy);
```

The `posix_trace_attr_getinherited()` and `posix_trace_attr_setinherited()` functions, respectively, get and set the inheritance policy stored in the `inheritance` attribute for traced processes across the `fork()` and `spawn()` operations.

The `posix_trace_attr_getlogfullpolicy()` and `posix_trace_attr_setlogfullpolicy()` functions, respectively, get and set the trace log full policy stored in the `log-full-policy` attribute of the attributes object pointed to by the `attr` argument.

The `posix_trace_attr_getstreamfullpolicy()` and `posix_trace_attr_setstreamfullpolicy()` functions, respectively, get and set the trace stream full policy stored in the `stream-full-policy` attribute of the attributes object pointed to by the `attr` argument.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

**posix_trace_attr_getlogsize, posix_trace_attr_getmaxdatasize,
posix_trace_attr_getmaxsystemeventsize, posix_trace_attr_getmaxusereventsize,
posix_trace_attr_getstreamsize, posix_trace_attr_setlogsize,
posix_trace_attr_setmaxdatasize, posix_trace_attr_setstreamsize**

Retrieve and set trace stream size attributes (**TRACING**)

```
TRC #include <sys/types.h>
TRC #include <trace.h>
TRC TRL int posix_trace_attr_getlogsize(const trace_attr_t *restrict attr,
TRC     size_t *restrict logsize);
```

```
TRC      int posix_trace_attr_getmaxdatasize(const trace_attr_t *restrict attr,
      size_t *restrict maxdatasize);
      int posix_trace_attr_getmaxsystemeventszize(
      const trace_attr_t *restrict attr,
      size_t *restrict eventszize);
      int posix_trace_attr_getmaxusereventsizize(
      const trace_attr_t *restrict attr,
      size_t data_len, size_t *restrict eventszize);
      int posix_trace_attr_getstreamszize(const trace_attr_t *restrict attr,
      size_t *restrict streamszize);
TRC TRL  int posix_trace_attr_setlogszize(trace_attr_t *attr,
      size_t logszize);
TRC      int posix_trace_attr_setmaxdatasize(trace_attr_t *attr,
      size_t maxdatasize);
      int posix_trace_attr_setstreamszize(trace_attr_t *attr,
      size_t streamszize);
```

The `posix_trace_attr_getlogszize()` function copies the log size, in bytes, from the `log-max-size` attribute of the attributes object pointed to by the `attr` argument into the variable pointed to by the `logszize` argument.

The `posix_trace_attr_setlogszize()` function sets the maximum allowed size, in bytes, in the `log-max-size` attribute of the attributes object pointed to by the `attr` argument, using the size value supplied by the `logszize` argument.

The `posix_trace_attr_getmaxdatasize()` function copies the maximum user trace event data size, in bytes, from the `max-data-size` attribute of the attributes object pointed to by the `attr` argument into the variable pointed to by the `maxdatasize` argument.

The `posix_trace_attr_getmaxsystemeventszize()` function calculates the maximum memory size, in bytes, required to store a single system trace event for the trace stream attributes object pointed to by the `attr` argument, returning the result in the variable pointed to by the `eventszize` argument.

The `posix_trace_attr_getmaxusereventsizize()` function calculates the maximum memory size, in bytes, required to store a single user trace event generated by a call to `posix_trace_event()` with a `data_len` parameter equal to the `data_len` value specified in this call. This value is calculated for the trace stream attributes object pointed to by the `attr` argument and is returned in the variable pointed to by the `eventszize` argument.

The `posix_trace_attr_getstreamszize()` function copies the stream size, in bytes, from the `stream-min-size` attribute of the attributes object pointed to by the `attr` argument into the variable pointed to by the `streamszize` argument.

The `posix_trace_attr_setmaxdatasize()` function sets the maximum allowed size, in bytes, in the `max-data-size` attribute of the attributes object pointed to by the `attr` argument, using the size value supplied by the `maxdatasize` argument.

The `posix_trace_attr_setstreamszize()` function sets the minimum allowed size, in bytes, in the `stream-min-size` attribute of the attributes object pointed to by the `attr` argument, using the size value supplied by the `streamszize` argument.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

posix_trace_clearClear trace stream and trace log (**TRACING**)

```
TRC #include <sys/types.h>
#include <trace.h>

int posix_trace_clear(trace_id_t trid);
```

The *posix_trace_clear()* function reinitializes the trace stream identified by the argument *trid* as if it were returning from the *posix_trace_create()* function, except that the same allocated resources will be reused, the mapping of trace event type identifiers to trace event names is unchanged, and the trace stream status remains unchanged (that is, if it was running, it remains running and if it was suspended, it remains suspended).

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000.
IEEE PASC Interpretation 1003.1 #123 is applied.

posix_trace_close, posix_trace_open, posix_trace_rewindTrace log management (**TRACING**)

```
TRC TRL #include <trace.h>

int posix_trace_close(trace_id_t trid);
int posix_trace_open(int file_desc, trace_id_t *trid);
int posix_trace_rewind(trace_id_t trid);
```

The *posix_trace_close()* function deallocates the trace log identifier indicated by *trid*, and all of its associated resources.

The *posix_trace_open()* function allocates the necessary resources and establishes the connection between a trace log identified by the *file_desc* argument and a trace stream identifier identified by the object pointed to by the *trid* argument.

The *posix_trace_rewind()* function resets the current trace event timestamp, which specifies the timestamp of the trace event that will be read by the next call to *posix_trace_getnext_event()*, to the timestamp of the oldest trace event recorded in the trace log identified by *trid*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

posix_trace_create, posix_trace_create_withlog, posix_trace_flush, posix_trace_shutdownTrace stream initialization, flush, and shutdown from a process (**TRACING**)

```
TRC #include <sys/types.h>
#include <trace.h>

int posix_trace_create(pid_t pid,
const trace_attr_t *restrict attr,
trace_id_t *restrict trid);
TRC TRL int posix_trace_create_withlog(pid_t pid,
const trace_attr_t *restrict attr, int file_desc,
```

```
    trace_id_t *restrict trid);
int posix_trace_flush(trace_id_t trid);
TRC int posix_trace_shutdown(trace_id_t trid);
```

The `posix_trace_create()` function creates an active trace stream.

The `posix_trace_create_withlog()` function is equivalent to `posix_trace_create()`, except that it associates a trace log with the stream.

The `posix_trace_flush()` function initiates a flush operation which copies the contents of the trace stream identified by the argument `trid` into the trace log associated with the trace stream at the creation time.

The `posix_trace_shutdown()` function stops the tracing of trace events in the trace stream identified by `trid`, as if `posix_trace_stop()` had been invoked.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

posix_trace_event, posix_trace_eventid_open

Trace functions for instrumenting application code (**TRACING**)

```
TRC #include <sys/types.h>
#include <trace.h>

void posix_trace_event(trace_event_id_t event_id,
    const void *restrict data_ptr, size_t data_len);
int posix_trace_eventid_open(const char *restrict event_name,
    trace_event_id_t *restrict event_id);
```

The `posix_trace_event()` function records the `event_id` and the user data pointed to by `data_ptr` in the trace stream into which the calling process is being traced and in which `event_id` is not filtered out.

The `posix_trace_eventid_open()` function associates a user trace event name with a trace event type identifier for the calling process.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

IEEE PASC Interpretation 1003.1 #127 is applied, correcting some editorial errors in the names of the `posix_trace_eventid_open()` function and the `event_id` argument.

**posix_trace_eventid_equal, posix_trace_eventid_get_name,
posix_trace_trid_eventid_open**Manipulate trace event type identifier (**TRACING**)

```

TRC      #include <trace.h>

int posix_trace_eventid_equal(trace_id_t trid, trace_event_id_t event1,
    trace_event_id_t event2);
int posix_trace_eventid_get_name(trace_id_t trid,
    trace_event_id_t event, char *event_name);
TRC TEF  int posix_trace_trid_eventid_open(trace_id_t trid,
    const char *restrict event_name,
    trace_event_id_t *restrict event);

```

The *posix_trace_eventid_equal()* function compares the trace event type identifiers *event1* and *event2* from the same trace stream or the same trace log identified by the *trid* argument.

The *posix_trace_eventid_get_name()* function returns, in the argument pointed to by *event_name*, the trace event name associated with the trace event type identifier identified by the argument *event*, for the trace stream or for the trace log identified by the *trid* argument.

The *posix_trace_trid_eventid_open()* function associates a user trace event name with a trace event type identifier for a given trace stream.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretations 1003.1 #123 and #129 are applied.

**posix_trace_eventset_add, posix_trace_eventset_del, posix_trace_eventset_empty,
posix_trace_eventset_fill, posix_trace_eventset_ismember**Manipulate trace event type sets (**TRACING**)

```

TRC TEF  #include <trace.h>

int posix_trace_eventset_add(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_del(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_empty(trace_event_set_t *set);
int posix_trace_eventset_fill(trace_event_set_t *set, int what);
int posix_trace_eventset_ismember(trace_event_id_t event_id,
    const trace_event_set_t *restrict set,
    int *restrict ismember);

```

The *posix_trace_eventset_add()* and *posix_trace_eventset_del()* functions, respectively, add or delete the individual trace event type specified by the value of the argument *event_id* to or from the trace event type set pointed to by the argument *set*.

The *posix_trace_eventset_empty()* function initializes the trace event type set pointed to by the *set* argument such that all trace event types defined, both system and user, are excluded from the set.

The `posix_trace_eventset_fill()` function initializes the trace event type set pointed to by the argument `set`, such that the set of trace event types defined by the argument `what` are included in the set.

The `posix_trace_eventset_ismember()` function tests whether the trace event type specified by the value of the argument `event_id` is a member of the set pointed to by the argument `set`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

posix_trace_eventtypelist_getnext_id, posix_trace_eventtypelist_rewind

Iterate over a mapping of trace event types (**TRACING**)

```
TRC #include <trace.h>

int posix_trace_eventtypelist_getnext_id(trace_id_t trid,
    trace_event_id_t *restrict event, int *restrict unavailable);
int posix_trace_eventtypelist_rewind(trace_id_t trid);
```

The first time `posix_trace_eventtypelist_getnext_id()` is called, the function returns in the variable pointed to by `event` the first trace event type identifier of the list of trace events of the trace stream identified by the `trid` argument. Successive calls to `posix_trace_eventtypelist_getnext_id()` return in the variable pointed to by `event` the next trace event type identifier in that same list.

The `posix_trace_eventtypelist_rewind()` function resets the next trace event type identifier to be read to the first trace event type identifier from the list of trace events used in the trace stream identified by `trid`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretations 1003.1 #123 and #129 are applied.

posix_trace_get_attr, posix_trace_get_status

Retrieve the trace attributes or trace statuses (**TRACING**)

```
TRC #include <trace.h>

int posix_trace_get_attr(trace_id_t trid, trace_attr_t *attr);
int posix_trace_get_status(trace_id_t trid,
    struct posix_trace_status_info *statusinfo);
```

The `posix_trace_get_attr()` function copies the attributes of the active trace stream identified by `trid` into the object pointed to by the `attr` argument.

The `posix_trace_get_status()` function returns, in the structure pointed to by the `statusinfo` argument, the current trace status for the trace stream identified by the `trid` argument.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

posix_trace_get_filter, posix_trace_set_filter

Retrieve and set filter of an initialized trace stream (**TRACING**)

```
TRC TEF #include <trace.h>

int posix_trace_get_filter(trace_id_t trid, trace_event_set_t *set);
int posix_trace_set_filter(trace_id_t trid,
    const trace_event_set_t *set, int how);
```

The *posix_trace_get_filter()* function retrieves, into the argument pointed to by *set*, the actual trace event filter from the trace stream specified by *trid*.

The *posix_trace_set_filter()* function changes the set of filtered trace event types after a trace stream identified by the *trid* argument is created.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

posix_trace_getnext_event, posix_trace_timedgetnext_event, posix_trace_trygetnext_event

Retrieve a trace event (**TRACING**)

```
TRC #include <sys/types.h>
#include <trace.h>

int posix_trace_getnext_event(trace_id_t trid,
    struct posix_trace_event_info *restrict event,
    void *restrict data, size_t num_bytes,
    size_t *restrict data_len, int *restrict unavailable);
TRC TMO int posix_trace_timedgetnext_event(trace_id_t trid,
    struct posix_trace_event_info *restrict event,
    void *restrict data, size_t num_bytes,
    size_t *restrict data_len, int *restrict unavailable,
    const struct timespec *restrict abs_timeout);
TRC int posix_trace_trygetnext_event(trace_id_t trid,
    struct posix_trace_event_info *restrict event,
    void *restrict data, size_t num_bytes,
    size_t *restrict data_len, int *restrict unavailable);
```

The *posix_trace_getnext_event()* function reports a recorded trace event either from an active trace stream without log or a pre-recorded trace stream identified by the *trid* argument.

The *posix_trace_trygetnext_event()* function reports a recorded trace event from an active trace stream without log identified by the *trid* argument.

The *posix_trace_timedgetnext_event()* function attempts to get another trace event from an active trace stream without log, as in the *posix_trace_getnext_event()* function. However, if no trace event is available from the trace stream, the implied wait is terminated when the timeout specified by the argument *abs_timeout* expires, and the function returns the error [ETIMEDOUT].

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

posix_trace_start, posix_trace_stop

Trace start and stop (**TRACING**)

```
TRC #include <trace.h>

int posix_trace_start(trace_id_t trid);
int posix_trace_stop (trace_id_t trid);
```

The *posix_trace_start()* and *posix_trace_stop()* functions, respectively, start and stop the trace stream identified by the argument *trid*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000. These functions are part of the Tracing Option Group and may not be available on all implementations.

IEEE PASC Interpretation 1003.1 #123 is applied.

posix_typed_mem_get_info

Query typed memory information (**ADVANCED REALTIME**)

```
TYM #include <sys/mman.h>

int posix_typed_mem_get_info(int fildes,
    struct posix_typed_mem_info *info);
```

Typed memory objects are pools of specialized storage, different from the main memory resource normally used by a processor to hold code and data, that can be mapped into the address space of one or more processes. For more information see XRAT, Section B.2.8.3.

The *posix_typed_mem_get_info()* function returns, in the *posix_tmi_length* field of the **posix_typed_mem_info** structure pointed to by *info*, the maximum length which can be successfully allocated by the typed memory object designated by *fildes*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

posix_typed_mem_open

Open a typed memory object (**ADVANCED REALTIME**)

```
TYM #include <sys/mman.h>

int posix_typed_mem_open(const char *name, int oflag, int tflag);
```

The *posix_typed_mem_open()* function establishes a connection between the typed memory object specified by the string pointed to by *name* and a file descriptor.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

pow, powf, powl

Power function

```
#include <math.h>

double pow(double x, double y);
float powf(float x, float y);
long double powl(long double x, long double y);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *powf()* and *powl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

pselect, select

Synchronous I/O multiplexing

```
#include <sys/select.h>

int pselect(int nfds, fd_set *restrict readfds,
            fd_set *restrict writefds, fd_set *restrict errorfds,
            const struct timespec *restrict timeout,
            const sigset_t *restrict sigmask);
int select(int nfds, fd_set *restrict readfds,
           fd_set *restrict writefds, fd_set *restrict errorfds,
           struct timeval *restrict timeout);
void FD_CLR(int fd, fd_set *fdset);
int FD_ISSET(int fd, fd_set *fdset);
void FD_SET(int fd, fd_set *fdset);
void FD_ZERO(fd_set *fdset);
```

The *pselect()* function examines the file descriptor sets whose addresses are passed in the *readfds*, *writefds*, and *errorfds* parameters to see whether some of their descriptors are ready for reading, are ready for writing, or have an exceptional condition pending, respectively.

The *select()* function is equivalent to the *pselect()* function, except as follows:

- For the *select()* function, the timeout period is given in seconds and microseconds in an argument of type **struct timeval**, whereas for the *pselect()* function the timeout period is given in seconds and nanoseconds in an argument of type **struct timespec**.
- The *select()* function has no *sigmask* argument; it behaves as *pselect()* does when *sigmask* is a null pointer.
- Upon successful completion, the *select()* function may modify the object pointed to by the *timeout* argument.

- Derivation First released in Issue 4, Version 2.
- Issue 6 The Open Group Corrigendum U026/6 is applied, changing the occurrences of *readfs* and *writefs* in the *select()* DESCRIPTION to be *readfds* and *writfds*.
Text referring to sockets is added to the DESCRIPTION.
The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.
The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:
- These functions are now mandatory.
- The *pselect()* function is added for alignment with IEEE Std 1003.1g-2000 and additional detail related to sockets semantics is added to the DESCRIPTION.
The *select()* function now requires inclusion of **<sys/select.h>**.
The **restrict** keyword is added to the *select()* prototype for alignment with the ISO/IEC 9899:1999 standard.
IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/70 is applied, updating the DESCRIPTION to reference the signal mask in terms of the calling thread rather than the process.

pthread_atfork

Register fork handlers

```
THR #include <pthread.h>

int pthread_atfork(void (*prepare)(void), void (*parent)(void),
void (*child)(void));
```

- Derivation First released in Issue 5. Derived from the POSIX Threads Extension.
- Issue 6 The *pthread_atfork()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.
IEEE PASC Interpretation 1003.1c #4 is applied, clarifying that the function should be declared in the **<pthread.h>** header.
The **<pthread.h>** header is added to the SYNOPSIS.

pthread_attr_destroy, pthread_attr_init

Destroy and initialize threads attributes object

```
THR #include <pthread.h>

int pthread_attr_destroy(pthread_attr_t *attr);
int pthread_attr_init(pthread_attr_t *attr);
```

- Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.
- Issue 6 The *pthread_attr_destroy()* and *pthread_attr_init()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE PASC Interpretation 1003.1 #107 is applied, noting that the effect of initializing an already initialized thread attributes object is undefined.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/71 is applied, updating the ERRORS section to add the optional [EINVAL] error for the *pthread_attr_destroy()* function, and the optional [EBUSY] error for the *pthread_attr_init()* function.

pthread_attr_getdetachstate, pthread_attr_setdetachstate

Get and set detachstate attribute

```
THR #include <pthread.h>

int pthread_attr_getdetachstate(const pthread_attr_t *attr,
    int *detachstate);
int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_attr_setdetachstate()* and *pthread_attr_getdetachstate()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/73 is applied, updating the ERRORS section to include the optional [EINVAL] error.

pthread_attr_getguardsize, pthread_attr_setguardsize

Get and set the thread guardsize attribute

```
XSI #include <pthread.h>

int pthread_attr_getguardsize(const pthread_attr_t *restrict attr,
    size_t *restrict guardsize);
int pthread_attr_setguardsize(pthread_attr_t *attr,
    size_t guardsize);
```

Derivation First released in Issue 5.

Issue 6 In the ERRORS section, a third [EINVAL] error condition is removed as it is covered by the second error condition.

The **restrict** keyword is added to the *pthread_attr_getguardsize()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/74 is applied, updating the ERRORS section to remove the [EINVAL] error ("The attribute *attr* is invalid."), and replacing it with the optional [EINVAL] error.

pthread_attr_getinheritsched, pthread_attr_setinheritsched

Get and set inheritsched attribute (**REALTIME THREADS**)

```
THR TPS #include <pthread.h>

int pthread_attr_getinheritsched(const pthread_attr_t *restrict attr,
int *restrict inheritsched);
int pthread_attr_setinheritsched(pthread_attr_t *attr,
int inheritsched);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6 The *pthread_attr_getinheritsched()* and *pthread_attr_setinheritsched()* functions are marked as part of the Threads and Thread Execution Scheduling options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The **restrict** keyword is added to the *pthread_attr_getinheritsched()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/75 is applied, clarifying the values of *inheritsched* in the DESCRIPTION and adding two optional [EINVAL] errors to the ERRORS section for checking when *attr* refers to an uninitialized thread attribute object.

pthread_attr_getschedparam, pthread_attr_setschedparam

Get and set schedparam attribute

```
THR #include <pthread.h>

int pthread_attr_getschedparam(const pthread_attr_t *restrict attr,
struct sched_param *restrict param);
int pthread_attr_setschedparam(pthread_attr_t *restrict attr,
const struct sched_param *restrict param);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread_attr_getschedparam()* and *pthread_attr_setschedparam()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/78 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

pthread_attr_getschedpolicy, pthread_attr_setschedpolicy

Get and set schedpolicy attribute (**REALTIME THREADS**)

```
THR TPS #include <pthread.h>

int pthread_attr_getschedpolicy(const pthread_attr_t *restrict attr,
    int *restrict policy);
int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6 The *pthread_attr_getschedpolicy()* and *pthread_attr_setschedpolicy()* functions are marked as part of the Threads and Thread Execution Scheduling options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread_attr_getschedpolicy()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/80 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

pthread_attr_getscope, pthread_attr_setscope

Get and set contentionscope attribute (**REALTIME THREADS**)

```
THR TPS #include <pthread.h>

int pthread_attr_getscope(const pthread_attr_t *restrict attr,
    int *restrict contentionscope);
int pthread_attr_setscope(pthread_attr_t *attr, int contentionscope);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6 The *pthread_attr_getscope()* and *pthread_attr_setscope()* functions are marked as part of the Threads and Thread Execution Scheduling options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The **restrict** keyword is added to the `pthread_attr_getscope()` prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/82 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

pthread_attr_getstack, pthread_attr_setstack

Get and set stack attributes

```
THR      #include <pthread.h>
TSA TSS  int pthread_attr_getstack(const pthread_attr_t *restrict attr,
        void **restrict stackaddr, size_t *restrict stacksize);
        int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr,
        size_t stacksize);
```

The `pthread_attr_getstack()` and `pthread_attr_setstack()` functions, respectively, get and set the thread creation stack attributes *stackaddr* and *stacksize* in the *attr* object.

These functions are appropriate for use by applications in an environment where the stack for a thread must be placed in some particular region of memory.

While it might seem that an application could detect stack overflow by providing a protected page outside the specified stack region, this cannot be done portably. Implementations are free to place the thread's initial stack pointer anywhere within the specified region to accommodate the machine's stack pointer behavior and allocation requirements. Furthermore, on some architectures, such as the IA-64, "overflow" might mean that two separate stack pointers allocated within the region will overlap somewhere in the middle of the region.

After a successful call to `pthread_attr_setstack(),` the storage area specified by the *stackaddr* parameter is under the control of the implementation, as described in Section 2.9.8, Use of Application-Managed Thread Stacks.

Derivation First released in Issue 6. Developed as an XSI extension and brought into the BASE as a response to IEEE PASC Interpretation 1003.1 #101.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC/D6/84 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

pthread_attr_getstackaddr, pthread_attr_setstackaddr

Get and set stackaddr attribute

```
THR TSA  #include <pthread.h>
OB       int pthread_attr_getstackaddr(const pthread_attr_t *restrict attr,
        void **restrict stackaddr);
        int pthread_attr_setstackaddr(pthread_attr_t *attr, void *stackaddr);
```

The specification of the *stackaddr* attribute presents several ambiguities that make portable use of these interfaces impossible, hence they have been marked obsolescent. The description of the single address parameter as a "stack" does not specify a particular relationship between the

address and the “stack” implied by that address. For example, the address may be taken as the low memory address of a buffer intended for use as a stack, or it may be taken as the address to be used as the initial stack pointer register value for the new thread. These two are not the same except for a machine on which the stack grows “up” from low memory to high, and on which a “push” operation first stores the value in memory and then increments the stack pointer register. Further, on a machine where the stack grows “down” from high memory to low, interpretation of the address as the “low memory” address requires a determination of the intended size of the stack. The Single UNIX Specification, Version 3 introduces the new interfaces *pthread_attr_setstack()* and *pthread_attr_getstack()* to resolve these ambiguities.

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_attr_getstackaddr()* and *pthread_attr_setstackaddr()* functions are marked as part of the Threads and Thread Stack Address Attribute options.

The **restrict** keyword is added to the *pthread_attr_getstackaddr()* prototype for alignment with the ISO/IEC 9899:1999 standard.

These functions are marked obsolescent.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/86 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

pthread_attr_getstacksize, pthread_attr_setstacksize

Get and set stacksize attribute

```
THR TSS #include <pthread.h>
int pthread_attr_getstacksize(const pthread_attr_t *restrict attr,
    size_t *restrict stacksize);
int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_attr_getstacksize()* and *pthread_attr_setstacksize()* functions are marked as part of the Threads and Thread Stack Address Attribute options. These functions are mandatory on systems supporting the Single UNIX Specification.

The **restrict** keyword is added to the *pthread_attr_getstacksize()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/87 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

pthread_barrier_destroy, pthread_barrier_init

Destroy and initialize a barrier object (**ADVANCED REALTIME THREADS**)

```
THR BAR #include <pthread.h>

int pthread_barrier_destroy(pthread_barrier_t *barrier);
int pthread_barrier_init(pthread_barrier_t *restrict barrier,
    const pthread_barrierattr_t *restrict attr, unsigned count);
```

Barriers are typically used in parallel DO/FOR loops to ensure that all threads have reached a particular stage in a parallel computation before allowing any to proceed to the next stage.

The *pthread_barrier_destroy()* function destroys the barrier referenced by *barrier* and releases any resources used by the barrier.

The *pthread_barrier_init()* function allocates any resources required to use the barrier referenced by *barrier* and initializes the barrier with attributes referenced by *attr*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

These functions are part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

pthread_barrier_wait

Synchronize at a barrier (**ADVANCED REALTIME THREADS**)

```
THR BAR #include <pthread.h>

int pthread_barrier_wait(pthread_barrier_t *barrier);
```

The *pthread_barrier_wait()* function synchronizes participating threads at the barrier referenced by *barrier*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

pthread_barrierattr_destroy, pthread_barrierattr_init

Destroy and initialize barrier attributes object (**ADVANCED REALTIME THREADS**)

```
THR BAR #include <pthread.h>

int pthread_barrierattr_destroy(pthread_barrierattr_t *attr);
int pthread_barrierattr_init(pthread_barrierattr_t *attr);
```

The *pthread_barrierattr_destroy()* function destroys a barrier attributes object.

The *pthread_barrierattr_init()* function initializes a barrier attributes object *attr* with the default value for all of the attributes defined by the implementation.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

These functions are part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

pthread_barrierattr_getpshared, pthread_barrierattr_setpshared

Get and set process-shared attribute of barrier attributes object (**ADVANCED REALTIME THREADS**)

```
THR #include <pthread.h>
BAR TSH int pthread_barrierattr_getpshared(
        const pthread_barrierattr_t *restrict attr,
        int *restrict pshared);
int pthread_barrierattr_setpshared(pthread_barrierattr_t *attr,
        int pshared);
```

The *pthread_barrierattr_getpshared()* function obtains the value of the *process-shared* attribute from the attributes object referenced by *attr*. The *pthread_barrierattr_setpshared()* function sets the *process-shared* attribute in an initialized attributes object referenced by *attr*.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000

These functions are part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

pthread_cancel

Cancel execution of a thread

```
THR #include <pthread.h>
int pthread_cancel(pthread_t thread);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_cancel()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_cleanup_pop, pthread_cleanup_push

Establish cancelation handlers

```
THR #include <pthread.h>
void pthread_cleanup_pop(int execute);
void pthread_cleanup_push(void (*routine)(void*), void *arg);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_cleanup_pop()* and *pthread_cleanup_push()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/88 is applied, updating the DESCRIPTION to describe the consequences of prematurely leaving a code block defined by the *pthread_cleanup_push()* and *pthread_cleanup_pop()* functions.

pthread_cond_broadcast, pthread_cond_signal

Broadcast or signal a condition

```
THR #include <pthread.h>

int pthread_cond_broadcast(pthread_cond_t *cond);
int pthread_cond_signal(pthread_cond_t *cond);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_cond_broadcast()* and *pthread_cond_signal()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_cond_destroy, pthread_cond_init

Destroy and initialize condition variables

```
THR #include <pthread.h>

int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_init(pthread_cond_t *restrict cond,
    const pthread_condattr_t *restrict attr);
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_cond_destroy()* and *pthread_cond_init()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE PASC Interpretation 1003.1c #34 is applied, updating the DESCRIPTION.

The **restrict** keyword is added to the *pthread_cond_init()* prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_cond_timedwait, pthread_cond_wait

Wait on a condition

```
THR #include <pthread.h>

int pthread_cond_timedwait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
int pthread_cond_wait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_cond_timedwait()* and *pthread_cond_wait()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The Open Group Corrigendum U021/9 is applied, correcting the prototype for the *pthread_cond_wait()* function from:

```
int pthread_cond_wait(pthread_cond_t *);
```

to:

```
int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
```

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics for the Clock Selection option.

The ERRORS section has an additional case for [EPERM] in response to IEEE PASC Interpretation 1003.1c #28.

The **restrict** keyword is added to the *pthread_cond_timedwait()* and *pthread_cond_wait()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/89 is applied, updating the DESCRIPTION for consistency with the *pthread_cond_destroy()* function that states it is safe to destroy an initialized condition variable upon which no threads are currently blocked.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/90 is applied, updating words in the DESCRIPTION from “the cancelability enable state” to “the cancelability type”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/91 is applied, updating the ERRORS section to remove the error case related to *abstime* from the *pthread_cond_wait()* function, and to make the error case related to *abstime* mandatory for *pthread_cond_timedwait()* for consistency with other functions.

pthread_condattr_destroy, pthread_condattr_init

Destroy and initialize condition variable attributes object

```
THR #include <pthread.h>

int pthread_condattr_destroy(pthread_condattr_t *attr);
int pthread_condattr_init(pthread_condattr_t *attr);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_condattr_destroy()* and *pthread_condattr_init()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_condattr_getclock, pthread_condattr_setclock

Get and set the clock selection condition variable attribute (**ADVANCED REALTIME**)

```
THR CS #include <pthread.h>

int pthread_condattr_getclock(const pthread_condattr_t *restrict attr,
clockid_t *restrict clock_id);
int pthread_condattr_setclock(pthread_condattr_t *attr,
clockid_t clock_id);
```

The `pthread_condattr_getclock()` function obtains the value of the `clock` attribute from the attributes object referenced by `attr`. The `pthread_condattr_setclock()` function sets the `clock` attribute in an initialized attributes object referenced by `attr`.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

pthread_condattr_getpshared, pthread_condattr_setpshared

Get and set the process-shared condition variable attributes

```
THR TSH #include <pthread.h>

int pthread_condattr_getpshared(const pthread_condattr_t *restrict attr,
                               int *restrict pshared);
int pthread_condattr_setpshared(pthread_condattr_t *attr,
                               int pshared);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_condattr_getpshared()` and `pthread_condattr_setpshared()` functions are marked as part of the Threads and Thread Process-Shared Synchronization options.

The **restrict** keyword is added to the `pthread_condattr_getpshared()` prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_create

Thread creation

```
THR #include <pthread.h>

int pthread_create(pthread_t *restrict thread,
                  const pthread_attr_t *restrict attr,
                  void *(*start_routine)(void*), void *restrict arg);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_create()` function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EPERM] mandatory error condition is added.

The thread CPU-time clock semantics are added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the `pthread_create()` prototype for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION is updated to make it explicit that the floating-point environment is inherited from the creating thread.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/44 is applied, adding text that the alternate stack is not inherited.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/93 is applied, updating the ERRORS section to remove the mandatory [EINVAL] error (“The value specified by *attr* is invalid”), and adding the optional [EINVAL] error (“The attributes specified by *attr* are invalid”).

pthread_detach

Detach a thread

```
THR #include <pthread.h>
int pthread_detach(pthread_t thread);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_detach()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/95 is applied, updating the ERRORS section so that the [EINVAL] and [ESRCH] error cases become optional.

pthread_equal

Compare thread IDs

```
THR #include <pthread.h>
int pthread_equal(pthread_t t1, pthread_t t2);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_equal()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_exit

Thread termination

```
THR #include <pthread.h>
void pthread_exit(void *value_ptr);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_exit()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_getconcurrency, pthread_setconcurrency

Get and set level of concurrency

```
XSI #include <pthread.h>

int pthread_getconcurrency(void);
int pthread_setconcurrency(int new_level);
```

Derivation First released in Issue 5.

Issue 6 No functional changes are made in this issue.

pthread_getcpuclockid

Access a thread CPU-time clock (**ADVANCED REALTIME THREADS**)

```
THR TCT #include <pthread.h>
#include <time.h>

int pthread_getcpuclockid(pthread_t thread_id, clockid_t *clock_id);
```

The *pthread_getcpuclockid()* function returns in *clock_id* the clock ID of the CPU-time clock of the thread specified by *thread_id*, if the thread specified by *thread_id* exists.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

This function is part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

pthread_getschedparam, pthread_setschedparam

Dynamic thread scheduling parameters access (**REALTIME THREADS**)

```
THR TPS #include <pthread.h>

int pthread_getschedparam(pthread_t thread, int *restrict policy,
    struct sched_param *restrict param);
int pthread_setschedparam(pthread_t thread, int policy,
    const struct sched_param *param);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_getschedparam()* and *pthread_setschedparam()* functions are marked as part of the Threads and Thread Execution Scheduling options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the *pthread_setschedparam()* function so that its second argument is of type **int**.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the `pthread_getschedparam()` prototype for alignment with the ISO/IEC 9899:1999 standard.

The Open Group Corrigendum U047/1 is applied, making it clear that any temporary adjustments to priority are not reflected as a result of any priority inheritance or the ceiling functions.

IEEE PASC Interpretation 1003.1 #96 is applied, noting that priority values can also be set by a call to the `pthread_setschedprio()` function.

pthread_getspecific, pthread_setspecific

Thread-specific data management

```
THR #include <pthread.h>
void *pthread_getspecific(pthread_key_t key);
int pthread_setspecific(pthread_key_t key, const void *value);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_getspecific()` and `pthread_setspecific()` functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE PASC Interpretation 1003.1c #3 (Part 6) is applied, updating the DESCRIPTION.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/96 is applied, updating the ERRORS section so that the [ENOMEM] error case is changed from “to associate the value with the key” to “to associate the non-NULL value with the key”.

pthread_join

Wait for thread termination

```
THR #include <pthread.h>
int pthread_join(pthread_t thread, void **value_ptr);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_join()` function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE Std 1003.1-2001/Cor 2-2004, XSH/TC2/D6/97 is applied, updating the ERRORS section so that the [EINVAL] error is made optional and the words “the implementation has detected” are removed from it.

pthread_key_create

Thread-specific data key creation

```
THR #include <pthread.h>
int pthread_key_create(pthread_key_t *key, void (*destructor)(void*));
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_key_create()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE PASC Interpretation 1003.1c #8 is applied, updating the DESCRIPTION.

pthread_key_delete

Thread-specific data key deletion

```
THR #include <pthread.h>
int pthread_key_delete(pthread_key_t key);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_key_delete()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_kill

Send a signal to a thread

```
THR #include <signal.h>
int pthread_kill(pthread_t thread, int sig);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_kill()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_mutex_destroy, pthread_mutex_init

Destroy and initialize a mutex

```
THR #include <pthread.h>
int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
    const pthread_mutexattr_t *restrict attr);
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_mutex_destroy()* and *pthread_mutex_init()* functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE PASC Interpretation 1003.1c #34 is applied, updating the DESCRIPTION.

The **restrict** keyword is added to the *pthread_mutex_init()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

pthread_mutex_getprioceiling, pthread_mutex_setprioceiling

Get and set the priority ceiling of a mutex (**REALTIME THREADS**)

```
THR TPP #include <pthread.h>

int pthread_mutex_getprioceiling(const pthread_mutex_t *restrict mutex,
int *restrict prioceiling);
int pthread_mutex_setprioceiling(pthread_mutex_t *restrict mutex,
int prioceiling, int *restrict old_ceiling);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6 The *pthread_mutex_getprioceiling()* and *pthread_mutex_setprioceiling()* functions are marked as part of the Threads and Thread Priority Protection options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Priority Protection option.

The [ENOSYS] error denoting non-support of the priority ceiling protocol for mutexes has been removed. This is since if the implementation provides the functions (regardless of whether `_POSIX_PTHREAD_PRIO_PROTECT` is defined), they must function as in the DESCRIPTION and therefore the priority ceiling protocol for mutexes is supported.

The **restrict** keyword is added to the *pthread_mutex_getprioceiling()* and *pthread_mutex_setprioceiling()* prototypes for alignment with the ISO/IEC 9899: 1999 standard.

pthread_mutex_lock, pthread_mutex_trylock, pthread_mutex_unlock

Lock and unlock a mutex

```
THR #include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_mutex_lock()`, `pthread_mutex_trylock()`, and `pthread_mutex_unlock()` functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The behavior when attempting to relock a mutex is defined.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/98 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_mutex_timedlock

Lock a mutex (**ADVANCED REALTIME**)

```
THR TMO #include <pthread.h>
#include <time.h>

int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abs_timeout);
```

The `pthread_mutex_timedlock()` function locks the mutex object referenced by `mutex`. If the mutex is already locked, the calling thread blocks until the mutex becomes available as in the `pthread_mutex_lock()` function. If the mutex cannot be locked without waiting for another thread to unlock the mutex, this wait terminates when the specified timeout expires.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. This function is part of the Advanced Realtime Option Group and may not be available on all implementations.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/99 is applied, marking the last paragraph in the DESCRIPTION as part of the Thread Priority Inheritance option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/100 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_mutexattr_destroy, pthread_mutexattr_init

Destroy and initialize mutex attributes object

```
THR #include <pthread.h>

int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
int pthread_mutexattr_init(pthread_mutexattr_t *attr);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_mutexattr_destroy()` and `pthread_mutexattr_init()` functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

IEEE PASC Interpretation 1003.1c #27 is applied, updating the ERRORS section.

pthread_mutexattr_getprioceiling, pthread_mutexattr_setprioceiling

Get and set prioceiling attribute of mutex attributes object (**REALTIME THREADS**)

THR TPP `#include <pthread.h>`

```
int pthread_mutexattr_getprioceiling(
    const pthread_mutexattr_t *restrict attr,
    int *restrict prioceiling);
int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,
    int prioceiling);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6 The `pthread_mutexattr_getprioceiling()` and `pthread_mutexattr_setprioceiling()` functions are marked as part of the Threads and Thread Priority Protection options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Priority Protection option.

The [ENOTSUP] error condition has been removed since these functions do not have a `protocol` argument.

The **restrict** keyword is added to the `pthread_mutexattr_getprioceiling()` prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_mutexattr_getprotocol, pthread_mutexattr_setprotocol

Get and set protocol attribute of mutex attributes object (**REALTIME THREADS**)

THR `#include <pthread.h>`

TPPITPI

```
int pthread_mutexattr_getprotocol(
    const pthread_mutexattr_t *restrict attr,
    int *restrict protocol);
int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,
    int protocol);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

Issue 6 The `pthread_mutexattr_getprotocol()` and `pthread_mutexattr_setprotocol()` functions are marked as part of the Threads option and either the Thread Priority Protection or Thread Priority Inheritance options.

These functions are part of the Realtime Threads Option Group and may not be available on all implementations.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Priority Protection or Thread Priority Inheritance options.

The **restrict** keyword is added to the `pthread_mutexattr_getprotocol()` prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_mutexattr_getpshared, pthread_mutexattr_setpshared

Get and set process-shared attribute

```
THR TSH #include <pthread.h>

int pthread_mutexattr_getpshared(
    const pthread_mutexattr_t *restrict attr,
    int *restrict pshared);
int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
    int pshared);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_mutexattr_getpshared()` and `pthread_mutexattr_setpshared()` functions are marked as part of the Threads and Thread Process-Shared Synchronization options. These functions are mandatory on systems supporting the Single UNIX Specification.

The **restrict** keyword is added to the `pthread_mutexattr_getpshared()` prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_mutexattr_gettype, pthread_mutexattr_settype

Get and set a mutex type attribute

```
XSI #include <pthread.h>

int pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict attr,
    int *restrict type);
int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type);
```

Derivation First released in Issue 5.

Issue 6 The Open Group Corrigendum U033/3 is applied. The SYNOPSIS for `pthread_mutexattr_gettype()` is updated so that the first argument is of type **const pthread_mutexattr_t***.

The **restrict** keyword is added to the `pthread_mutexattr_gettype()` prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_once

Dynamic package initialization

```
THR #include <pthread.h>

int pthread_once(pthread_once_t *once_control,
                 void (*init_routine)(void));
pthread_once_t once_control = PTHREAD_ONCE_INIT;
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The *pthread_once()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The [EINVAL] error is added as a may fail case for if either argument is invalid.

pthread_rwlock_destroy, pthread_rwlock_init

Destroy and initialize a read-write lock object

```
THR #include <pthread.h>

int pthread_rwlock_destroy(pthread_rwlock_t *rwlock);
int pthread_rwlock_init(pthread_rwlock_t *restrict rwlock,
                       const pthread_rwlockattr_t *restrict attr);
```

Derivation First released in Issue 5.

Issue 6 These functions are mandatory on systems supporting the Single UNIX Specification.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension). The initializer macro is also deleted from the SYNOPSIS.
- The DESCRIPTION is updated as follows:
 - It explicitly notes allocation of resources upon initialization of a read-write lock object.
 - A paragraph is added specifying that copies of read-write lock objects may not be used.
- An [EINVAL] error is added to the ERRORS section for *pthread_rwlock_init()*, indicating that the *rwlock* value is invalid.

The **restrict** keyword is added to the *pthread_rwlock_init()* prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_rwlock_rdlock, pthread_rwlock_tryrdlock

Lock a read-write lock object for reading

```
THR #include <pthread.h>

int pthread_rwlock_rdlock(pthread_rwlock_t *rwlock);
int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwlock);
```

Derivation First released in Issue 5.

Issue 6 These functions are mandatory on systems supporting the Single UNIX Specification.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION is updated as follows:
 - Conditions under which writers have precedence over readers are specified.
 - Failure of *pthread_rwlock_tryrdlock()* is clarified.
 - A paragraph on the maximum number of read locks is added.
- In the ERRORS sections, [EBUSY] is modified to take into account write priority, and [EDEADLK] is deleted as a *pthread_rwlock_tryrdlock()* error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/101 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_rwlock_timedrdlock

Lock a read-write lock for reading

```
THR TMO #include <pthread.h>
#include <time.h>

int pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict rwlock,
    const struct timespec *restrict abs_timeout);
```

The function is part of the Advanced Realtime Option Group and need not be supported on all implementations.

The *pthread_rwlock_timedrdlock()* function applies a read lock to the read-write lock referenced by *rwlock* as in the *pthread_rwlock_rdlock()* function. However, if the lock cannot be acquired without waiting for other threads to unlock the lock, this wait terminates when the specified timeout expires.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/102 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_rwlock_timedwrlock

Lock a read-write lock for writing

```
THR TMO #include <pthread.h>
#include <time.h>

int pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict rwlock,
    const struct timespec *restrict abs_timeout);
```

The function is part of the Advanced Realtime option group and need not be supported on all implementations.

The *pthread_rwlock_timedwrlock()* function applies a write lock to the read-write lock referenced by *rwlock* as in the *pthread_rwlock_wrlock()* function. However, if the lock cannot be acquired without waiting for other threads to unlock the lock, this wait terminates when the specified timeout expires.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/103 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_rwlock_trywrlock, pthread_rwlock_wrlock

Lock a read-write lock object for writing

```
THR #include <pthread.h>

int pthread_rwlock_trywrlock(pthread_rwlock_t *rwlock);
int pthread_rwlock_wrlock(pthread_rwlock_t *rwlock);
```

Derivation First released in Issue 5.

Issue 6 These functions are mandatory on systems supporting the Single UNIX Specification.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The [EDEADLK] error is deleted as a *pthread_rwlock_trywrlock()* error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/104 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_rwlock_unlock

Unlock a read-write lock object

```
THR #include <pthread.h>
int pthread_rwlock_unlock(pthread_rwlock_t *rwlock);
```

Derivation First released in Issue 5.

Issue 6 These functions are mandatory on systems supporting the Single UNIX Specification.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION is updated as follows:
 - The conditions under which writers have precedence over readers are specified.
 - The concept of read-write lock owner is deleted.

pthread_rwlockattr_destroy, pthread_rwlockattr_init

Destroy and initialize read-write lock attributes object

```
THR #include <pthread.h>
int pthread_rwlockattr_destroy(pthread_rwlockattr_t *attr);
int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);
```

Derivation First released in Issue 5.

Issue 6 These functions are mandatory on systems supporting the Single UNIX Specification.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).

pthread_rwlockattr_getpshared, pthread_rwlockattr_setpshared

Get and set process-shared attribute of read-write lock attributes object

```
THR TSH #include <pthread.h>
int pthread_rwlockattr_getpshared(
    const pthread_rwlockattr_t *restrict attr,
    int *restrict pshared);
int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr,
    int pshared);
```

Derivation First released in Issue 5.

Issue 6 These functions are mandatory on systems supporting the Single UNIX Specification.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR TSH to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION notes that additional attributes are implementation-defined.

The **restrict** keyword is added to the `pthread_rwlockattr_getpshared()` prototype for alignment with the ISO/IEC 9899:1999 standard.

pthread_self

Get calling thread's ID

```
THR #include <pthread.h>
pthread_t pthread_self(void);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_self()` function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_setcancelstate, pthread_setcanceltype, pthread_testcancel

Set cancelability state

```
THR #include <pthread.h>
int pthread_setcancelstate(int state, int *oldstate);
int pthread_setcanceltype(int type, int *oldtype);
void pthread_testcancel(void);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The `pthread_setcancelstate()`, `pthread_setcanceltype()`, and `pthread_testcancel()` functions are marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pthread_setschedprio

Dynamic thread scheduling parameters access (**REALTIME THREADS**)

```
THR TPS #include <pthread.h>
int pthread_setschedprio(pthread_t thread, int prio);
```

The `pthread_setschedprio()` function sets the scheduling priority for the thread whose thread ID is given by `thread` to the value given by `prio`.

Derivation First released in Issue 6. Included as a response to IEEE PASC Interpretation 1003.1 #96. This function was added since previously there was no way for a thread to lower its own priority without going to the tail of the threads list for its new priority. This capability is required to bound the duration of the priority inversion encountered by the thread.

This function is marked as part of the Realtime Threads Option Group and may not be available on all implementations.

pthread_sigmask, sigprocmask

Examine and change blocked signals

```
#include <signal.h>
```

```
THR int pthread_sigmask(int how, const sigset_t *restrict set,  
sigset_t *restrict oset);  
CX int sigprocmask(int how, const sigset_t *restrict set,  
sigset_t *restrict oset);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The *pthread_sigmask()* function is marked as part of the Threads option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The SYNOPSIS for *sigprocmask()* is marked as a CX extension to note that the presence of this function in the **<signal.h>** header is an extension to the ISO C standard.

The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The DESCRIPTION is updated to explicitly state the functions which may generate the signal.

The **restrict** keyword is added to the *pthread_sigmask()* and *sigprocmask()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/105 is applied, updating “process’ signal mask” to “thread’s signal mask” in the DESCRIPTION and RATIONALE sections.

pthread_spin_destroy, pthread_spin_init

Destroy or initialize a spin lock object (**ADVANCED REALTIME THREADS**)

```
THR SPI #include <pthread.h>  
  
int pthread_spin_destroy(pthread_spinlock_t *lock);  
int pthread_spin_init(pthread_spinlock_t *lock, int pshared);
```

Spin locks represent an extremely low-level synchronization mechanism suitable primarily for use on shared memory multi-processors. It is typically an atomically modified Boolean value that is set to one when the lock is held and to zero when the lock is freed.

When a caller requests a spin lock that is already held, it typically spins in a loop testing whether the lock has become available. Such spinning wastes processor cycles so the lock should only

be held for short durations and not across sleep/block operations. Callers should unlock spin locks before calling sleep operations.

The `pthread_spin_destroy()` function destroys the spin lock referenced by `lock` and release any resources used by the lock.

The `pthread_spin_init()` function allocates any resources required to use the spin lock referenced by `lock` and initializes the lock to an unlocked state.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

These functions are part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

pthread_spin_lock, pthread_spin_trylock

Lock a spin lock object (**ADVANCED REALTIME THREADS**)

```
THR SPI #include <pthread.h>

int pthread_spin_lock(pthread_spinlock_t *lock);
int pthread_spin_trylock(pthread_spinlock_t *lock);
```

The `pthread_spin_lock()` function locks the spin lock referenced by `lock`.

The `pthread_spin_trylock()` function locks the spin lock referenced by `lock` if it is not held by any thread. Otherwise, the function fails.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

These functions are part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/107 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

pthread_spin_unlock

Unlock a spin lock object (**ADVANCED REALTIME THREADS**)

```
THR SPI #include <pthread.h>

int pthread_spin_unlock(pthread_spinlock_t *lock);
```

The `pthread_spin_unlock()` function releases the spin lock referenced by `lock` which was locked via the `pthread_spin_lock()` or `pthread_spin_trylock()` functions.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

These functions are part of the Advanced Realtime Threads Option Group and may not be available on all implementations.

In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

ptsname

Get name of the slave pseudo-terminal device

```
xSI #include <stdlib.h>
char *ptsname(int fildes);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

putc

Put byte on a stream

```
#include <stdio.h>
int putc(int c, FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

putchar

Put byte on stdout stream

```
#include <stdio.h>
int putchar(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

putenv

Change or add a value to environment

```
xSI #include <stdlib.h>
int putenv(char *string);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

putmsg, putpmsg

Send a message on a STREAM (**STREAMS**)

```
xSR #include <stropts.h>
int putmsg(int fildes, const struct strbuf *ctlptr,
           const struct strbuf *dataptr, int flags);
int putpmsg(int fildes, const struct strbuf *ctlptr,
            const struct strbuf *dataptr, int band, int flags);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked as part of the XSI STREAMS Option Group and need not be supported on all implementations supporting the Single UNIX Specification.

puts

Put a string on standard output

```
#include <stdio.h>
int puts(const char *s);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

putwc

Put a wide character on a stream

```
#include <stdio.h>
#include <wchar.h>
wint_t putwc(wchar_t wc, FILE *stream);
```

Derivation First released as a World-wide Portability Interface in Issue 4.

Issue 6 No functional changes are made in this issue.

putwchar

Put a wide character on stdout stream

```
#include <wchar.h>
wint_t putwchar(wchar_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

qsort

Sort a table of data

```
#include <stdlib.h>
void qsort(void *base, size_t nel, size_t width,
           int (*compar)(const void *, const void *));
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/49 is applied, adding the last sentence to the first non-shaded paragraph in the DESCRIPTION, and the following two paragraphs. These changes are for alignment with the ISO C standard.

raise

Send a signal to the executing process

```
#include <signal.h>
int raise(int sig);
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EINVAL] error condition is added.

rand, rand_r, srand

Pseudo-random number generator

```
#include <stdlib.h>
int rand(void);
int rand_r(unsigned *seed);
void srand(unsigned seed);
```

TSF

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *rand_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

pread, read

Read from a file

```
#include <unistd.h>
ssize_t pread(int fildes, void *buf, size_t nbyte, off_t offset);
ssize_t read(int fildes, void *buf, size_t nbyte);
```

XSI

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION now states that if *read()* is interrupted by a signal after it has successfully read some data, it returns the number of bytes read. In Issue 3, it was optional whether *read()* returned the number of bytes read, or whether it returned *-1* with *errno* set to [EINTR]. This is a FIPS requirement.
- In the DESCRIPTION, text is added to indicate that for regular files, no data transfer occurs past the offset maximum established in the open file description associated with *fildes*. This change is to support large files.
- The [EOVERFLOW] mandatory error condition is added.

- The [ENXIO] optional error condition is added.

Text referring to sockets is added to the DESCRIPTION.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The effect of reading zero bytes is clarified.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *read()* results are unspecified for typed memory objects.

New RATIONALE is added to explain the atomicity requirements for input and output operations.

The following error conditions are added for operations on sockets: [EAGAIN], [ECONNRESET], [ENOTCONN], and [ETIMEDOUT].

The [EIO] error is changed to “may fail”.

The following error conditions are added for operations on sockets: [ENOBUFS] and [ENOMEM].

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/108 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

readdir, readdir_r

Read directory

```
#include <dirent.h>
```

```
struct dirent *readdir(DIR *dirp);
```

```
TSF int readdir_r(DIR *restrict dirp, struct dirent *restrict entry,
    struct dirent **restrict result);
```

Derivation First released in Issue 2.

Issue 6 The *readdir_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The Open Group Corrigendum U026/7 is applied, correcting the prototype for *readdir_r()*.

The Open Group Corrigendum U026/8 is applied, clarifying the wording of the successful return for the *readdir_r()* function.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- A statement is added to the DESCRIPTION indicating the disposition of certain fields in **struct dirent** when an entry refers to a symbolic link.
- The [EOVERFLOW] mandatory error condition is added. This change is to support large files.

- The [ENOENT] optional error condition is added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *readdir_r()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

readlink

Read the contents of a symbolic link

```
#include <unistd.h>

ssize_t readlink(const char *restrict path, char *restrict buf,
                 size_t bufsize);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The return type is changed to **ssize_t**, to align with the IEEE P1003.1a draft standard.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- This function is made mandatory.
- In this function it is possible for the return value to exceed the range of the type **ssize_t** (since **size_t** has a larger range of positive values than **ssize_t**). A sentence restricting the size of the **size_t** object is added to the description to resolve this conflict.

The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The FUTURE DIRECTIONS section is changed to None.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

The **restrict** keyword is added to the *readlink()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

readv

Read a vector

```
xsi #include <sys/uio.h>

     ssize_t readv(int fildes, const struct iovec *iov, int iovcnt);
```

Derivation First released in Issue 4, Version 2.

Issue 6 Split out from the *read()* reference page.

realloc

Memory reallocator

```
#include <stdlib.h>

void *realloc(void *ptr, size_t size);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, if there is not enough available memory, the setting of *errno* to [ENOMEM] is added.
- The [ENOMEM] error condition is added.

realpath

Resolve a pathname

```
xsi #include <stdlib.h>

char *realpath(const char *restrict file_name,
               char *restrict resolved_name);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The **restrict** keyword is added to the *realpath()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/51 is applied, adding new text to the DESCRIPTION for the case when *resolved_name* is a null pointer, and changing the [EINVAL] error text.

recv

Receive a message from a connected socket

```
#include <sys/socket.h>

ssize_t recv(int socket, void *buffer, size_t length, int flags);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 No functional changes since the XNS, Issue 5.2 specification.

recvfrom

Receive a message from a socket

```
#include <sys/socket.h>

ssize_t recvfrom(int socket, void *restrict buffer, size_t length,
                 int flags, struct sockaddr *restrict address,
                 socklen_t *restrict address_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

recvmsg

Receive a message from a socket

```
#include <sys/socket.h>

ssize_t recvmsg(int socket, struct msghdr *message, int flags);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

regcomp, regerror, regex, regfree

Regular expression matching

```
#include <regex.h>

int regcomp(regex_t *restrict preg, const char *restrict pattern,
            int cflags);
size_t regerror(int errcode, const regex_t *restrict preg,
               char *restrict errbuf, size_t errbuf_size);
int regex(const regex_t *restrict preg, const char *restrict string,
          size_t nmatch, regmatch_t pmatch[restrict], int eflags);
void regfree(regex_t *preg);
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.
The REG_ENOSYS constant is removed.

The **restrict** keyword is added to the *regcomp()*, *regerror()*, and *regex()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

remainder, remainderf, remainderl

Remainder function

```
#include <math.h>

double remainder(double x, double y);
float remainderf(float x, float y);
long double remainderl(long double x, long double y);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *remainder()* function is no longer marked as an extension.

The *remainderf()* and *remainderl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

remove

Remove a file

```
#include <stdio.h>

int remove(const char *path);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1) and the ISO C standard.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION, RETURN VALUE, and ERRORS sections are updated so that if *path* is not a directory, *remove()* is equivalent to *unlink()*, and if it is a directory, it is equivalent to *rmdir()*.

remquo, remquof, remquol

Remainder functions

```
#include <math.h>

double remquo(double x, double y, int *quo);
float remquof(float x, float y, int *quo);
long double remquol(long double x, long double y, int *quo);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

rename

Rename a file

```
#include <stdio.h>

int rename(const char *old, const char *new);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] mandatory error condition is added.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Details are added regarding the treatment of symbolic links.
- The [ELOOP] optional error condition is added.

rewind

Reset file position indicator in a stream

```
#include <stdio.h>
void rewind(FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

rewinddir

Reset position of directory stream to the beginning of a directory

```
#include <dirent.h>
void rewinddir(DIR *dirp);
```

Derivation First released in Issue 2.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

rindex

Character string operations (**LEGACY**)

```
xSI #include <strings.h>
char *rindex(const char *s, int c);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations.
The *strchr()* function is preferred over this function.

For maximum portability, it is recommended to replace the function call to *rindex()* as follows:

```
#define rindex(a,b) strchr((a),(b))
```

rint, rintf, rintl

Round-to-nearest integral value

```
#include <math.h>
double rint(double x);
float rintf(float x);
long double rintl(long double x);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *rintf()* and *rintl()* functions are added.
- The *rint()* function is no longer marked as an extension.
- The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

rmdir

Remove a directory

```
#include <unistd.h>
int rmdir(const char *path);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to indicate the results of naming a symbolic link in *path*.
- The [EIO] mandatory error condition is added.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

round, roundf, roundl

Round to nearest integer value in floating-point format

```
#include <math.h>
double round(double x);
float roundf(float x);
long double roundl(long double x);
```

These functions round their argument to the nearest integer value in floating-point format, rounding halfway cases away from zero, regardless of the current rounding direction.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

scalb

Load exponent of a radix-independent floating-point number

```
OB XSI #include <math.h>
double scalb(double x, double n);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked obsolescent.

Although this function is not part of the ISO/IEC 9899:1999 standard, the RETURN VALUE and ERROR sections are updated to align with the error handling in the ISO/IEC 9899:1999 standard.

scalbln, scalblnf, scalblnl, scalbn, scalbnf, scalbnl

Compute exponent using FLT_RADIX

```
#include <math.h>

double scalbln(double x, long n);
float scalblnf(float x, long n);
long double scalblnl(long double x, long n);
double scalbn(double x, int n);
float scalbnf(float x, int n);
long double scalbnl(long double x, int n);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

sched_get_priority_max, sched_get_priority_min

Get priority limits (**REALTIME**)

```
PSITPS #include <sched.h>

int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 These functions are marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

The [ESRCH] error condition has been removed since these functions do not take a *pid* argument.

sched_getparam

Get scheduling parameters (**REALTIME**)

```
PS #include <sched.h>

int sched_getparam(pid_t pid, struct sched_param *param);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sched_getparam()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

sched_getschedulerGet scheduling policy (**REALTIME**)

PS

```
#include <sched.h>

int sched_getscheduler(pid_t pid);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sched_getscheduler()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

sched_rr_get_intervalGet execution time limits (**REALTIME**)

PSITPS

```
#include <sched.h>

int sched_rr_get_interval(pid_t pid, struct timespec *interval);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sched_rr_get_interval()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

sched_setparamSet scheduling parameters (**REALTIME**)

PS

```
#include <sched.h>

int sched_setparam(pid_t pid, const struct sched_param *param);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sched_setparam()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the effect of this function on a thread's scheduling parameters is added.

- Sections describing two-level scheduling and atomicity of the function are added.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #100 is applied, changing text about the target process in the DESCRIPTION to be described in terms of the tail of the thread list for its priority.

sched_setscheduler

Set scheduling policy and parameters (**REALTIME**)

PS

```
#include <sched.h>

int sched_setscheduler(pid_t pid, int policy,
    const struct sched_param *param);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sched_setscheduler()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the effect of this function on a thread's scheduling parameters is added.
- Sections describing two-level scheduling and atomicity of the function are added.

The SCHED_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

sched_yield

Yield processor

PSITHR

```
#include <sched.h>

int sched_yield(void);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Issue 6 The *sched_yield()* function is marked as part of the Process Scheduling and Threads options.

seekdir

Set position of directory stream

```
XSI #include <dirent.h>
void seekdir(DIR *dirp, long loc);
```

Derivation First released in Issue 2.

Issue 6 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

sem_close

Close a named semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_close(sem_t *sem);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_close()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/113 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

sem_destroy

Destroy an unnamed semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_destroy(sem_t *sem);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_destroy()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/113 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

sem_getvalue

Get the value of a semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_getvalue(sem_t *restrict sem, int *restrict sval);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_getvalue()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

The **restrict** keyword is added to the *sem_getvalue()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/115 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

sem_init

Initialize an unnamed semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_init(sem_t *sem, int pshared, unsigned value);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_init()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/116 is applied, updating the DESCRIPTION to add the *sem_timedwait()* function for alignment with IEEE Std 1003.1d-1999.

sem_open

Initialize and open a named semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
sem_t *sem_open(const char *name, int oflag, ...);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_open()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/118 is applied, updating the DESCRIPTION to describe the conditions to return the same semaphore address on a call to *sem_open()*. The words “and at least one previous successful *sem_open()* call for this semaphore has not been matched with a *sem_close()* call” are added.

sem_post

Unlock a semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_post(sem_t *sem);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_post()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

SCHED_SPORADIC is added to the list of scheduling policies for which the thread that is to be unblocked is specified for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/119 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

sem_timedwait

Lock a semaphore (**ADVANCED REALTIME**)

```
SEM TMO #include <semaphore.h>
#include <time.h>
int sem_timedwait(sem_t *restrict sem,
const struct timespec *restrict abs_timeout);
```

The *sem_timedwait()* function locks the semaphore referenced by *sem* as in the *sem_wait()* function. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore by performing a *sem_post()* function, this wait is terminated when the specified timeout expires.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1d-1999. These functions are part of the Advanced Realtime Option Group and may not be available on all implementations.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/120 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

sem_trywait, sem_wait

Lock a semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_trywait(sem_t *sem);
int sem_wait(sem_t *sem);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_trywait()* and *sem_wait()* functions are marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/121 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

sem_unlink

Remove a named semaphore (**REALTIME**)

```
SEM #include <semaphore.h>
int sem_unlink(const char *name);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *sem_unlink()* function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

semctl

XSI semaphore control operations

```
XSI #include <sys/sem.h>
int semctl(int semid, int semnum, int cmd, ...);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

semget

Get set of XSI semaphores

```
XSI #include <sys/sem.h>
int semget(key_t key, int nsems, int semflg);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/122 is applied, updating the DESCRIPTION from “each semaphore in the set shall not be initialized” to “each semaphore in the set need not be initialized”.

semop

XSI semaphore operations

```
XSI #include <sys/sem.h>
int semop(int semid, struct sembuf *sops, size_t nsops);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

send

Send a message on a socket

```
#include <sys/socket.h>
ssize_t send(int socket, const void *buffer, size_t length, int flags);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

sendmsg

Send a message on a socket using a message structure

```
#include <sys/socket.h>
ssize_t sendmsg(int socket, const struct msghdr *message, int flags);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

sendto

Send a message on a socket

```
#include <sys/socket.h>
ssize_t sendto(int socket, const void *message, size_t length,
               int flags, const struct sockaddr *dest_addr,
               socklen_t dest_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

setbuf

Assign buffering to a stream

```
#include <stdio.h>
```

```
void setbuf(FILE *restrict stream, char *restrict buf);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The prototype for *setbuf()* is updated for alignment with the ISO/IEC 9899:1999 standard.

setegid

Set effective group ID

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

```
int setegid(gid_t gid);
```

If *gid* is equal to the real group ID or the saved set-group-ID, or if the process has appropriate privileges, *setegid()* sets the effective group ID of the calling process to *gid*; the real group ID, saved set-group-ID, and any supplementary group IDs remain unchanged, and the supplementary group list is not affected in any way.

Derivation First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

setenv

Add or change environment variable

CX

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

```
int setenv(const char *envname, const char *envval, int overwrite);
```

The *setenv()* function updates or adds a variable in the environment of the calling process.

Derivation First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

seteuid

Set effective user ID

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

```
int seteuid(uid_t uid);
```

If *uid* is equal to the real user ID or the saved set-user-ID, or if the process has appropriate privileges, *seteuid()* sets the effective user ID of the calling process to *uid*; the real user ID and saved set-user-ID remain unchanged and the supplementary group list is not affected in any way.

Derivation First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

setgid

Set-group-ID

```
#include <unistd.h>
int setgid(gid_t gid);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Functionality associated with `_POSIX_SAVED_IDS` is now mandated. This is a FIPS requirement.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The effects of `setgid()` in processes without appropriate privileges are changed.
- A requirement that the supplementary group list is not affected is added.

setjmp

Set jump point for a non-local goto

```
#include <setjmp.h>
int setjmp(jmp_buf env);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

setkeySet encoding key (**CRYPT**)

```
xsi #include <stdlib.h>
void setkey(const char *key);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

setlocale

Set program locale

```
#include <locale.h>
char *setlocale(int category, const char *locale);
```

Derivation First released in Issue 3.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/124 is applied, updating the DESCRIPTION to clarify the behavior of:

```
setlocale(LC_ALL, "");
```


setpgid

Set process group ID for job control

```
#include <unistd.h>
int setpgid(pid_t pid, pid_t pgid);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed. The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The *setpgid()* function is mandatory since `_POSIX_JOB_CONTROL` is required to be defined in this issue. This is a FIPS requirement.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/56 is applied, changing the wording in the DESCRIPTION from “the process group ID of the indicated process shall be used” to “the process ID of the indicated process shall be used”. This change reverts the wording to as in ISO/IEC 9945-1:1996 (POSIX-1); it appeared to be an unintentional change.

setpgrp

Set process group ID

```
xSI #include <unistd.h>
pid_t setpgrp(void);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

setregid

Set real and effective group IDs

```
xSI #include <unistd.h>
int setregid(gid_t rgid, gid_t egid);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

setreuid

Set real and effective user IDs

```
xSI #include <unistd.h>
int setreuid(uid_t ruid, uid_t euid);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

setsid

Create session and set process group ID

```
#include <unistd.h>
pid_t setsid(void);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

setsockopt

Set the socket options

```
#include <sys/socket.h>
int setsockopt(int socket, int level, int option_name,
               const void *option_value, socklen_t option_len);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/125 is applied, updating the SO_LINGER option in the DESCRIPTION to refer to the calling thread rather than the process.

setuid

Set user ID

```
#include <unistd.h>
int setuid(uid_t uid);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The functionality associated with `_POSIX_SAVED_IDS` is now mandatory. This is a FIPS requirement.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The effects of `setuid()` in processes without appropriate privileges are changed.
- A requirement that the supplementary group list is not affected is added.

setvbuf

Assign buffering to a stream

```
#include <stdio.h>
int setvbuf(FILE *restrict stream, char *restrict buf, int type,
            size_t size);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `setvbuf()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

shm_open

Open a shared memory object (**REALTIME**)

```
SHM #include <sys/mman.h>
int shm_open(const char *name, int oflag, mode_t mode);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The `shm_open()` function is marked as part of the Shared Memory Objects option. The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Shared Memory Objects option.

shm_unlink

Remove a shared memory object (**REALTIME**)

```
SHM #include <sys/mman.h>
int shm_unlink(const char *name);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The `shm_unlink()` function is marked as part of the Shared Memory Objects option.

In the DESCRIPTION, text is added to clarify that reusing the same name after a `shm_unlink()` will not attach to the old shared memory object.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Shared Memory Objects option.

shmat

XSI shared memory attach operation

```
XSI #include <sys/shm.h>
void *shmat(int shmid, const void *shmaddr, int shmflg);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 The Open Group Corrigendum U021/13 is applied, changing the expression:

`(shmaddr - ((ptrdiff_t)shmaddr % SHMLBA))`

to:

`(shmaddr - ((uintptr_t)shmaddr % SHMLBA))`

shmctl

XSI shared memory control operations

```
xSI #include <sys/shm.h>
int shmctl(int shmid, int cmd, struct shmids *buf);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

shmdt

XSI shared memory detach operation

```
xSI #include <sys/shm.h>
int shmdt(const void *shmaddr);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

shmget

Get XSI shared memory segment

```
xSI #include <sys/shm.h>
int shmget(key_t key, size_t size, int shmflg);
```

Derivation First released in Issue 2. Derived from Issue 2 of the SVID.

Issue 6 No functional changes are made in this issue.

shutdown

Shut down socket send and receive operations

```
#include <sys/socket.h>
int shutdown(int socket, int how);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

sigaction

Examine and change signal action

```
cx #include <signal.h>
int sigaction(int sig, const struct sigaction *restrict act,
struct sigaction *restrict oact);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The Open Group Corrigendum U028/7 is applied. In the paragraph entitled "Signal Effects on Other Functions", a reference to *sigpending()* is added.

In the DESCRIPTION, the text “Signal Generation and Delivery”, “Signal Actions”, and “Signal Effects on Other Functions” are moved to a separate section.

Text describing functionality from the Realtime Signals option is marked.

The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The [ENOTSUP] error condition is added.

The **restrict** keyword is added to the *sigaction()* prototype for alignment with the ISO/IEC 9899:1999 standard.

References to the *wait3()* function are removed.

The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sigaddset

Add a signal to a signal set

```
CX #include <signal.h>
int sigaddset(sigset_t *set, int signo);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sigaltstack

Set and get signal alternate stack context

```
XSI #include <signal.h>
int sigaltstack(const stack_t *restrict ss, stack_t *restrict oss);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The **restrict** keyword is added to the *sigaltstack()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/58 is applied, updating the first sentence to include “for the current thread”.

sigdelset

Delete a signal from a signal set

```
CX #include <signal.h>
int sigdelset(sigset_t *set, int signo);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sigemptyset

Initialize and empty a signal set

```
CX #include <signal.h>
int sigemptyset(sigset_t *set);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sigfillset

Initialize and fill a signal set

```
CX #include <signal.h>
int sigfillset(sigset_t *set);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sighold, sigignore, sigpause, sigrelse, sigset

Signal management

```
XSI #include <signal.h>
int sighold(int sig);
int sigignore(int sig);
int sigpause(int sig);
int sigrelse(int sig);
void (*sigset(int sig, void (*disp)(int)))(int);
```

Derivation First released in Issue 4, Version 2.

Issue 6 References to the *wait3()* function are removed.

The XSI functions are split out into their own reference page.

siginterrupt

Allow signals to interrupt functions

```
xSI #include <signal.h>
int siginterrupt(int sig, int flag);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

sigismember

Test for a signal in a signal set

```
CX #include <signal.h>
int sigismember(const sigset_t *set, int signo);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

siglongjmp

Non-local goto with signal handling

```
CX #include <setjmp.h>
void siglongjmp(sigjmp_buf env, int val);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The DESCRIPTION is rewritten in terms of *longjmp()*.

The SYNOPSIS is marked CX since the presence of this function in the **<setjmp.h>** header is an extension over the ISO C standard.

signal

Signal management

```
#include <signal.h>
void (*signal(int sig, void (*func)(int)))(int);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

References to the *wait3()* function are removed.

The *sighold()*, *sigignore()*, *sigrelse()*, and *sigset()* functions are split out onto their own reference page.

signbit

Test sign

```
#include <math.h>

int signbit(real-floating x);
```

The *signbit()* macro determines whether the sign of its argument value is negative.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

sigpending

Examine pending signals

```
CX #include <signal.h>

int sigpending(sigset_t *set);
```

Derivation First released in Issue 3.

Issue 6 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sigqueue

Queue a signal to a process (**REALTIME**)

```
RTS #include <signal.h>

int sigqueue(pid_t pid, int signo, const union sigval value);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Issue 6 The *sigqueue()* function is marked as part of the Realtime Signals Extension option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Realtime Signals Extension option.

sigsetjmp

Set jump point for a non-local goto

```
CX #include <setjmp.h>

int sigsetjmp(sigjmp_buf env, int savemask);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The DESCRIPTION is reworded in terms of *setjmp()*.

The SYNOPSIS is marked CX since the presence of this function in the **<setjmp.h>** header is an extension over the ISO C standard.

sigsuspend

Wait for a signal

```
CX #include <signal.h>
int sigsuspend(const sigset_t *sigmask);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The text in the RETURN VALUE section has been changed from “suspends process execution” to “suspends thread execution”. This reflects IEEE PASC Interpretation 1003.1c #40.

Text in the APPLICATION USAGE section has been replaced.

The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

sigtimedwait, sigwaitinfo

Wait for queued signals (**REALTIME**)

```
RTS #include <signal.h>
int sigtimedwait(const sigset_t *restrict set,
    siginfo_t *restrict info,
    const struct timespec *restrict timeout);
int sigwaitinfo(const sigset_t *restrict set,
    siginfo_t *restrict info);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Issue 6 These functions are marked as part of the Realtime Signals Extension option.

The Open Group Corrigendum U035/3 is applied. The SYNOPSIS of the *sigwaitinfo()* function has been corrected so that the second argument is of type **siginfo_t***.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Realtime Signals Extension option.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that the CLOCK_MONOTONIC clock, if supported, is used to measure timeout intervals.

The **restrict** keyword is added to the *sigtimedwait()* and *sigwaitinfo()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/130 is applied, restoring wording in the RETURN VALUE section to that in the original base document (“An implementation should only check for this error if no signal is pending in *set* and it is necessary to wait”).

sigwait

Wait for queued signals

cx

```
#include <signal.h>

int sigwait(const sigset_t *restrict set, int *restrict sig);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Issue 6 The **restrict** keyword is added to the *sigwait()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/131 is applied, updating the DESCRIPTION section to state that if more than a single thread is blocked in *sigwait()*, it is unspecified which of the waiting threads returns, and that if a signal is generated for a specific thread only that thread shall return.

sin, sinf, sinl

Sine function

```
#include <math.h>

double sin(double x);
float sinf(float x);
long double sinl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *sinf()* and *sinl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

sinh, sinhf, sinhl

Hyperbolic sine function

```
#include <math.h>

double sinh(double x);
float sinhf(float x);
long double sinhl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *sinhf()* and *sinhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

sleep

Suspend execution for an interval of time

```
#include <unistd.h>

unsigned sleep(unsigned seconds);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

socketmark

Determine whether a socket is at the out-of-band mark

```
#include <sys/socket.h>

int socketmark(int s);
```

The *socketmark()* function determines whether the socket specified by the descriptor *s* is at the out-of-band data mark .

The *socketmark()* function replaces the historical SIOCATMARK command to *ioctl()* which implemented the same functionality on many implementations. Using a wrapper function follows the adopted conventions to avoid specifying commands to the *ioctl()* function, other than those now included to support XSI STREAMS. The *socketmark()* function could be implemented as follows:

```
#include <sys/ioctl.h>

int socketmark(int s)
{
    int val;
    if (ioctl(s, SIOCATMARK, &val) == -1)
        return(-1);
    return(val);
}
```

The use of [ENOTTY] to indicate an incorrect descriptor type matches the historical behavior of SIOCATMARK.

Derivation First released in Issue 6. Derived from IEEE Std 1003.1g-2000.

socket

Create an endpoint for communication

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

socketpair

Create a pair of connected sockets

```
#include <sys/socket.h>

int socketpair(int domain, int type, int protocol,
               int socket_vector[2]);
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

sqrt, sqrtf, sqrtl

Square root function

```
#include <math.h>

double sqrt(double x);
float sqrtf(float x);
long double sqrtl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *sqrtf()* and *sqrtl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

stat

Get file status

```
#include <sys/stat.h>

int stat(const char *restrict path, struct stat *restrict buf);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] mandatory error condition is added.
- The [ELOOP] mandatory error condition is added.
- The [E_OVERFLOW] mandatory error condition is added. This change is to support large files.
- The [ENAMETOOLONG] and the second [E_OVERFLOW] optional error conditions are added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Details are added regarding the treatment of symbolic links.
- The [ELOOP] optional error condition is added.

The **restrict** keyword is added to the *stat()* prototype for alignment with the ISO/IEC 9899:1999 standard.

stderr, stdin, stdout

Standard I/O streams

```
#include <stdio.h>
extern FILE *stderr, *stdin, *stdout;
```

Derivation First released in Issue 1.

Issue 6 A note that *stderr* is expected to be open for reading and writing is added to the DESCRIPTION.

strcasecmp, strncasecmp

Case-insensitive string comparisons

```
xsi #include <strings.h>
int strcasecmp(const char *s1, const char *s2);
int strncasecmp(const char *s1, const char *s2, size_t n);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

strcat

Concatenate two strings

```
#include <string.h>
char *strcat(char *restrict s1, const char *restrict s2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *strcat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

strchr

String scanning operation

```
#include <string.h>
char *strchr(const char *s, int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strcmp

Compare two strings

```
#include <string.h>
int strcmp(const char *s1, const char *s2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strcoll

String comparison using collating information

```
#include <string.h>
```

```
int strcoll(const char *s1, const char *s2);
```

Derivation First released in Issue 3.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] optional error condition is added.

An example is added.

strcpy

Copy a string

```
#include <string.h>
```

```
char *strcpy(char *restrict s1, const char *restrict s2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *strcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

strcspn

Get length of a complementary substring

```
#include <string.h>
```

```
size_t strcspn(const char *s1, const char *s2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The Open Group Corrigendum U030/1 is applied. The text of the RETURN VALUE section is updated to indicate that the computed segment length is returned, not the *s1* length.

strdup

Duplicate a string

```
xsi #include <string.h>
```

```
char *strdup(const char *s1);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

strerror, strerror_r

Get error message string

```
#include <string.h>
```

```
char *strerror(int errnum);
```

```
TSF int strerror_r(int errnum, char *strerrbuf, size_t buflen);
```

Derivation First released in Issue 3.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the fact that *errno* may be set is added.
- The [EINVAL] optional error condition is added.

The *strerror_r()* function is added in response to IEEE PASC Interpretation 1003.1c #39.

The *strerror_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

strfmon

Convert monetary value to a string

```
XSI #include <monetary.h>
```

```
ssize_t strfmon(char *restrict s, size_t maxsize,  
const char *restrict format, ...);
```

Derivation First released in Issue 4.

Issue 6 The **restrict** keyword is added to the *strfmon()* prototype for alignment with the ISO/IEC 9899:1999 standard.

strftime

Convert date and time to a string

```
#include <time.h>
```

```
size_t strftime(char *restrict s, size_t maxsize,  
const char *restrict format, const struct tm *restrict timeptr);
```

Derivation First released in Issue 3.

Issue 6 The Open Group Corrigendum U033/8 is applied. The %v conversion specifier is changed from “Otherwise, it is week 53 of the previous year, and the next week is week 1” to “Otherwise, it is the last week of the previous year, and the next week is week 1”.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The %C, %D, %e, %h, %n, %r, %R, %t, and %T conversion specifiers are added.

- The modified conversion specifiers are added for consistency with the ISO/IEC 9945-2: 1993 (POSIX-2) *date* utility.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strptime()* prototype is updated.
- The DESCRIPTION is extensively revised.
- The *%z* conversion specifier is added.

A new example is added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/60 is applied, describing the consequences of creating or modifying broken-down time structures.

strlen

Get string length

```
#include <string.h>
size_t strlen(const char *s);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strncat

Concatenate a string with part of another

```
#include <string.h>
char *strncat(char *restrict s1, const char *restrict s2, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *strncat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

strncmp

Compare part of two strings

```
#include <string.h>
int strncmp(const char *s1, const char *s2, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strncpy

Copy part of a string

```
#include <string.h>
char *strncpy(char *restrict s1, const char *restrict s2, size_t n);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `strncpy()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

strpbrk

Scan string for byte

```
#include <string.h>
```

```
char *strpbrk(const char *s1, const char *s2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strptime

Date and time conversion

```
xsl #include <time.h>
char *strptime(const char *restrict buf, const char *restrict format,
               struct tm *restrict tm);
```

Derivation First released in Issue 4.

Issue 6 The Open Group Corrigendum U033/5 is applied. The `%r` specifier description is reworded.

The **restrict** keyword is added to the `strptime()` prototype for alignment with the ISO/IEC 9899:1999 standard.

The Open Group Corrigendum U047/2 is applied, adding APPLICATION USAGE that states that the effect of multiple calls to `strptime()` using the same `tm` structure is unspecified.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” for consistency with `strptime()`.

strchr

String scanning operation

```
#include <string.h>
```

```
char *strchr(const char *s, int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strspn

Get length of a substring

```
#include <string.h>
```

```
size_t strspn(const char *s1, const char *s2);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

strstr

Find a substring

```
#include <string.h>

char *strstr(const char *s1, const char *s2);
```

Derivation First released in Issue 3. Included for alignment with ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 No functional changes are made in this issue.

strtod, strtod, strtold

Convert string to a double-precision number

```
#include <stdlib.h>

double strtod(const char *restrict nptr, char **restrict endptr);
float strtod(const char *restrict nptr, char **restrict endptr);
long double strtold(const char *restrict nptr, char **restrict endptr);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strtod()* function is updated.
- The *strtod()* and *strtold()* functions are added.
- The DESCRIPTION is extensively revised.

ISO/IEC 9899:1999 standard, Technical Corrigendum No. 1 is incorporated.

strtoimax, strtoumax

Convert string to integer type

```
#include <inttypes.h>

intmax_t strtoimax(const char *restrict nptr, char **restrict endptr,
                  int base);
uintmax_t strtoumax(const char *restrict nptr, char **restrict endptr,
                   int base);
```

These functions are equivalent to the *strtol()*, *strtoll()*, *strtoul()*, and *strtoull()* functions, except that the initial portion of the string is converted to **intmax_t** and **uintmax_t** representation, respectively.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

strtok, strtok_r

Split string into tokens

```
#include <string.h>
```

```
char *strtok(char *restrict s1, const char *restrict s2);  
TSF char *strtok_r(char *restrict s, const char *restrict sep,  
char **restrict lastp);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *strtok_r()* function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety. A portable multi-threaded application may only safely call the function when access to the function is serialized.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *strtok()* and *strtok_r()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

strtol, strtoll

Convert string to a long integer

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

```
long strtol(const char *restrict str, char **restrict endptr, int base);  
long long strtoll(const char *restrict str, char **restrict endptr,  
int base)
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strtol()* prototype is updated.
- The *strtoll()* function is added.

strtoul, strtoull

Convert string to an unsigned long

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

```
unsigned long strtoul(const char *restrict str,  
char **restrict endptr, int base);  
unsigned long long strtoull(const char *restrict str,  
char **restrict endptr, int base);
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] error condition is added for when the value of *base* is not supported.

In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strtoul()* prototype is updated.
- The *strtoull()* function is added.

strxfrm

String transformation

```
#include <string.h>
```

```
size_t strxfrm(char *restrict s1, const char *restrict s2, size_t n);
```

Derivation First released in Issue 3. Included for alignment with the ISO C standard.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The *strxfrm()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

swab

Swap bytes

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

```
void swab(const void *restrict src, void *restrict dest,
          ssize_t nbytes);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The **restrict** keyword is added to the *swab()* prototype for alignment with the ISO/IEC 9899:1999 standard.

symlink

Make symbolic link to a file

```
#include <unistd.h>

int symlink(const char *path1, const char *path2);
```

Derivation First released in Issue 4, Version 2.

Issue 6 The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION text is updated.
- The [ELOOP] optional error condition is added.

sync

Schedule file system updates

```
xsi #include <unistd.h>

void sync(void);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

sysconf

Get configurable system variables

```
#include <unistd.h>

long sysconf(int name);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The symbol CLK_TCK is obsolescent and removed. It is replaced with the phrase “clock ticks per second”.

The symbol {PASS_MAX} is removed.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Table entries are added for the following variables: _SC_REGEX, _SC_SHELL, _SC_REGEX_VERSION, _SC_SYMLINK_MAX.

The following *sysconf()* variables and their associated names are added for alignment with IEEE Std 1003.1d-1999:

```
_POSIX_ADVISORY_INFO
_POSIX_CPUTIME
_POSIX_SPAWN
_POSIX_SPORADIC_SERVER
_POSIX_THREAD_CPUTIME
_POSIX_THREAD_SPORADIC_SERVER
_POSIX_TIMEOUTS
```

The following changes are made to the DESCRIPTION for alignment with IEEE Std 1003.1j-2000:

- A statement expressing the dependency of support for some system variables on implementation options is added.
- The following system variables are added:

```
_POSIX_BARRIERS
_POSIX_CLOCK_SELECTION
_POSIX_MONOTONIC_CLOCK
_POSIX_READER_WRITER_LOCKS
_POSIX_SPIN_LOCKS
_POSIX_TYPED_MEMORY_OBJECTS
```

The following system variables are added for alignment with IEEE Std 1003.2d-1994:

```
_POSIX2_PBS
_POSIX2_PBS_ACCOUNTING
_POSIX2_PBS_LOCATE
_POSIX2_PBS_MESSAGE
_POSIX2_PBS_TRACK
```

The following *sysconf()* variables and their associated names are added for alignment with IEEE Std 1003.1q-2000:

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_INHERIT
_POSIX_TRACE_LOG
```

The macros associated with the *c89* programming models are marked LEGACY, and new equivalent macros associated with *c99* are introduced.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/62 is applied, updating the DESCRIPTION to denote that the *_PC** and *_SC** symbols are now required to be supported. A corresponding change has been made in the Base Definitions volume of IEEE Std 1003.1-2001. The deletion in the second paragraph removes some duplicated text. Additional symbols that were erroneously omitted from this reference page have been added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/63 is applied, making it clear in the RETURN VALUE section that the value returned for *sysconf(_SC_OPEN_MAX)* may change if a call to *setrlimit()* adjusts the RLIMIT_NOFILE soft limit.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/134 is applied, updating the DESCRIPTION to remove an erroneous entry for *_POSIX_SYMLINK_MAX*. This corrects an error in IEEE Std 1003.1-2001/Cor 1-2002.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/135 is applied, removing *_POSIX_FILE_LOCKING*, *_POSIX_MULTI_PROCESS*, *_POSIX2_C_VERSION*, and *_XOPEN_XCU_VERSION* (and their associated *_SC_** variables) from the DESCRIPTION and APPLICATION USAGE sections.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/136 is applied, adding *_POSIX_SS_REPL_MAX*, *_POSIX_TRACE_EVENT_NAME_MAX*, *_POSIX_TRACE_NAME_MAX*, *_POSIX_TRACE_SYS_MAX*, and *_POSIX_TRACE_USER_EVENT_MAX* (and their associated *_SC_** variables) to the DESCRIPTION. The RETURN VALUE and APPLICATION USAGE sections are updated to note that if variables are dependent on unsupported options, the results are unspecified.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/137 is applied, removing `_REGEX_VERSION` and `_SC_REGEX_VERSION`.

system

Issue a command

```
#include <stdlib.h>

int system(const char *command);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION is adjusted to reflect the behavior on systems that do not support the Shell option.

tan, tanf, tanl

Tangent function

```
#include <math.h>

double tan(double x);
float tanf(float x);
long double tanl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `tanf()` and `tanl()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/64 is applied, correcting the last paragraph in the RETURN VALUE section.

tanh, tanhf, tanhl

Hyperbolic tangent functions

```
#include <math.h>

double tanh(double x);
float tanhf(float x);
long double tanhl(long double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `tanhf()` and `tanhl()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

tcdrain

Wait for transmission of output

```
#include <termios.h>
int tcdrain(int fildev);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the final paragraph is no longer conditional on `_POSIX_JOB_CONTROL`. This is a FIPS requirement.
- The [EIO] error is added.

tcflow

Suspend or restart the transmission or reception of data

```
#include <termios.h>
int tcflow(int fildev, int action);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] error is added.

tcflush

Flush non-transmitted output data, non-read input data, or both

```
#include <termios.h>
int tcflush(int fildev, int queue_selector);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The Open Group Corrigendum U035/1 is applied. In the ERRORS and APPLICATION USAGE sections, references to `tcflow()` are replaced with `tcflush()`.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the final paragraph is no longer conditional on `_POSIX_JOB_CONTROL`. This is a FIPS requirement.
- The [EIO] error is added.

tcgetattr

Get the parameters associated with the terminal

```
#include <termios.h>
```

```
int tcgetattr(int fildev, struct termios *termios_p);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the DESCRIPTION, the rate returned as the input baud rate shall be the output rate. Previously, the number zero was also allowed but was obsolescent.

tcgetpgrp

Get the foreground process group ID

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

```
pid_t tcgetpgrp(int fildev);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.
The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text previously conditional on support for `_POSIX_JOB_CONTROL` is now mandatory. This is a FIPS requirement.

tcgetsid

Get process group ID for session leader for controlling terminal

XSI

```
#include <termios.h>
```

```
pid_t tcgetsid(int fildev);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

tcsendbreak

Send a “break” for a specific duration

```
#include <termios.h>
```

```
int tcsendbreak(int fildev, int duration);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text previously conditional on `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.

- The [EIO] error is added.

tcsetattr

Set the parameters associated with the terminal

```
#include <termios.h>

int tcsetattr(int fildev, int optional_actions,
              const struct termios *termios_p);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text previously conditional on `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.
- The [EIO] error is added.

In the DESCRIPTION, the text describing use of `tcsetattr()` from a process which is a member of a background process group is clarified.

tcsetpgrp

Set the foreground process group ID

```
#include <unistd.h>

int tcsetpgrp(int fildev, pid_t pgid_id);
```

Derivation First released in Issue 3. Included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION and ERRORS sections, text previously conditional on `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.

The Open Group Corrigendum U047/4 is applied, adding a new paragraph to the end of the DESCRIPTION. This change is for consistency with IEEE Std 1003.1-1996 (POSIX.1), Section 7.2.

tdelete, tfind, tsearch, twalk

Manage a binary search tree

XSI

```
#include <search.h>

void *tdelete(const void *restrict key, void **restrict rootp,
              int (*compar)(const void *, const void *));
void *tfind(const void *key, void *const *rootp,
            int (*compar)(const void *, const void *));
void *tsearch(const void *key, void **rootp,
              int (*compar)(const void *, const void *));
void twalk(const void *root,
           void (*action)(const void *, VISIT, int));
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The **restrict** keyword is added to the *tdelete()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

telldir

Current location of a named directory stream

```
xSI #include <dirent.h>
long telldir(DIR *dirp);
```

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

tempnam

Create a name for a temporary file

```
xSI #include <stdio.h>
char *tempnam(const char *dir, const char *pfx);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

tgamma, tgammaf, tgamma

Compute gamma function

```
#include <math.h>
double tgamma(double x);
float tgammaf(float x);
long double tgamma(long double x);
```

These functions compute the *gamma()* function of *x*.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/65 is applied, correcting the third paragraph in the RETURN VALUE section.

time

Get time

```
#include <time.h>
time_t time(time_t *tloc);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The EXAMPLES, RATIONALE, and FUTURE DIRECTIONS sections are added.

timer_createCreate a per-process timer (**REALTIME**)

```
TMR #include <signal.h>
#include <time.h>

int timer_create(clockid_t clockid, struct sigevent *restrict evp,
                timer_t *restrict timerid);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *timer_create()* function is marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

CPU-time clocks are added for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding the requirement for the CLOCK_MONOTONIC clock under the Monotonic Clock option.

The **restrict** keyword is added to the *timer_create()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/138 is applied, updating the DESCRIPTION and APPLICATION USAGE sections to describe the case when a timer is created with the notification method set to SIGEV_THREAD.

timer_deleteDelete a per-process timer (**REALTIME**)

```
TMR #include <time.h>

int timer_delete(timer_t timerid);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. This is part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *timer_delete()* function is marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/139 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

timer_getoverrun, timer_gettime, timer_settime

Per-process timers (**REALTIME**)

TMR

```
#include <time.h>

int timer_getoverrun(timer_t timerid);
int timer_gettime(timer_t timerid, struct itimerspec *value);
int timer_settime(timer_t timerid, int flags,
    const struct itimerspec *restrict value,
    struct itimerspec *restrict ovalue);
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension. These functions are part of the Realtime Option Group and may not be available on all implementations.

Issue 6 The *timer_getoverrun()*, *timer_gettime()*, and *timer_settime()* functions are marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

The [EINVAL] error condition is updated to include the following: “and the *it_value* member of that structure did not specify zero seconds and nanoseconds.” This change is for IEEE PASC Interpretation 1003.1 #89.

The DESCRIPTION for *timer_getoverrun()* is updated to clarify that “If no expiration signal has been delivered for the timer, or if the Realtime Signals Extension is not supported, the return value of *timer_getoverrun()* is unspecified”.

The **restrict** keyword is added to the *timer_settime()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/141 is applied, updating the ERRORS section to include an optional [EINVAL] error for the case when a timer is created with the notification method set to SIGEV_THREAD. APPLICATION USAGE text is also added.

times

Get process and waited-for child process times

```
#include <sys/times.h>

clock_t times(struct tms *buffer);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

tmpfile

Create a temporary file

```
#include <stdio.h>

FILE *tmpfile(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the ERRORS section, the [Eoverflow] condition is added. This change is to support large files.
- The [EMFILE] optional error condition is added.

tmpnam

Create a name for a temporary file

```
#include <stdio.h>
```

```
char *tmpnam(char *s);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The DESCRIPTION is expanded for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/142 is applied, updating the DESCRIPTION to allow implementations of the *tmpnam()* function to call *tmpnam()*.

toascii

Translate integer to a 7-bit ASCII character

```
xSI #include <ctype.h>
```

```
int toascii(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

tolower

Transliterate uppercase characters to lowercase

```
#include <ctype.h>
```

```
int tolower(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

toupper

Transliterate lowercase characters to uppercase

```
#include <ctype.h>
```

```
int toupper(int c);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

towctrans

Wide-character transliteration

```
#include <wctype.h>
```

```
wint_t towctrans(wint_t wc, wctrans_t desc);
```

Derivation First released in Issue 5. Derived from the ISO/IEC 9899: 1990 standard.

Issue 6 No functional changes are made in this issue.

towlower

Transliterate uppercase wide-character code to lowercase

```
#include <wctype.h>
```

```
wint_t tolower(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

toupper

Transliterate lowercase wide-character code to uppercase

```
#include <wctype.h>
```

```
wint_t toupper(wint_t wc);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

trunc, truncf, trunc1

Round to truncated integer value

```
#include <math.h>
```

```
double trunc(double x);  
float truncf(float x);  
long double trunc1(long double x);
```

These functions round their argument to the integer value, in floating format, nearest to but no larger in magnitude than the argument.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

truncate

Truncate a file to a specified length

XSI

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

```
int truncate(const char *path, off_t length);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This reference page is split out from the *ftruncate()* reference page.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

ttyname, ttyname_r

Find pathname of a terminal

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

```
char *ttyname(int fildev);
```

```
TSF int ttyname_r(int fildev, char *name, size_t namesize);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `ttyname_r()` function is marked as part of the Thread-Safe Functions option. Support for this option is mandatory on systems supporting the Single UNIX Specification.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The statement that `errno` is set on error is added.
- The [EBADF] and [ENOTTY] optional error conditions are added.

daylight, timezone, tzname, tzset

Set timezone conversion information

```
#include <time.h>
```

```
XSI extern int daylight;
```

```
extern long timezone;
```

```
CX extern char *tzname[2];
```

```
void tzset(void);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The example is corrected.

ualarm

Set the interval timer

```
OB XSI #include <unistd.h>
```

```
useconds_t ualarm(useconds_t useconds, useconds_t interval);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked obsolescent.

ulimit

Get and set process limits

```
xSI #include <ulimit.h>
long ulimit(int cmd, ...);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

umask

Set and get file mode creation mask

```
#include <sys/stat.h>
mode_t umask(mode_t cmask);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

uname

Get name of current system

```
#include <sys/utsname.h>
int uname(struct utsname *name);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

ungetc

Push byte back into input stream

```
#include <stdio.h>
int ungetc(int c, FILE *stream);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes are made in this issue.

ungetwc

Push wide-character code back into input stream

```
#include <stdio.h>
#include <wchar.h>
wint_t ungetwc(wint_t wc, FILE *stream);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The [EILSEQ] optional error condition is marked CX.

unlink

Remove a directory entry

```
#include <unistd.h>
int unlink(const char *path);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the effect is specified if *path* specifies a symbolic link.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

unlockpt

Unlock a pseudo-terminal master/slave pair

```
XSI #include <stdlib.h>
int unlockpt(int fildes);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

unsetenv

Remove environment variable

```
CX #include <stdlib.h>
int unsetenv(const char *name);
```

Derivation First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

usleep

Suspend execution for an interval

```
OB XSI #include <unistd.h>
int usleep(useconds_t useconds);
```

Applications are recommended to use *nanosleep()* if the Timers option is supported, or *setitimer()*, *timer-create()*, *timer-delete()*, *timer_getoverrun()*, *timer_gettime()*, or *timer_settime()* instead of this function.

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked obsolescent.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/144 is applied, updating the DESCRIPTION from “process’ signal mask” to “thread’s signal mask”, and adding a statement that the *usleep()* function need not be reentrant.

utime

Set file access and modification times

```
#include <utime.h>
```

```
int utime(const char *path, const struct utimbuf *times);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

utimes

Set file access and modification times (**LEGACY**)

xsi

```
#include <sys/time.h>
```

```
int utimes(const char *path, const struct timeval times[2]);
```

Derivation First released in Issue 4, Version 2.

Issue 6 This function is marked LEGACY and may not be available on all implementations.

For applications portability, the *utime()* function should be used to set file access and modification times instead of *utimes()*.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

va_arg, va_copy, va_end, va_start

Handle variable argument list

```
#include <stdarg.h>
```

```
type va_arg(va_list ap, type);  
void va_copy(va_list dest, va_list src);  
void va_end(va_list ap);  
void va_start(va_list ap, argN);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 The *va_copy()* function is added to copy a variable argument list. This is added for alignment with the ISO/IEC 9899:1999 standard.

vfork

Create new process; share virtual memory

```
OB XSI #include <unistd.h>
pid_t vfork(void);
```

Conforming applications are recommended not to depend on *vfork()*, but to use *fork()* instead. The *vfork()* function may be withdrawn in a future version.

The use of *vfork()* for any purpose except as a prelude to an immediate call to a function from the *exec* family, or to *_exit()*, is not advised.

Derivation First released in Issue 4, Version 2.

Issue 6 Marked obsolescent.

vfprintf, vprintf, vsnprintf, vsprintf

Format output of a stdarg argument list

```
#include <stdarg.h>
#include <stdio.h>

int vfprintf(FILE *restrict stream, const char *restrict format,
             va_list ap);
int vprintf(const char *restrict format, va_list ap);
int vsnprintf(char *restrict s, size_t n, const char *restrict format,
             va_list ap);
int vsprintf(char *restrict s, const char *restrict format, va_list ap);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *vfprintf()*, *vprintf()*, *vsnprintf()*, and *vsprintf()* functions are updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

vfscanf, vscanf, vsscanf

Format input of a stdarg list

```
#include <stdarg.h>
#include <stdio.h>

int vfscanf(FILE *restrict stream, const char *restrict format,
            va_list arg);
int vscanf(const char *restrict format, va_list arg);
int vsscanf(const char *restrict s, const char *restrict format,
            va_list arg);
```

The *vscanf()*, *vfscanf()*, and *vsscanf()* functions are equivalent to the *scanf()*, *fscanf()*, and *sscanf()* functions, respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined in the **<stdarg.h>** header.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

vfwprintf, vswprintf, vwprintf

Wide-character formatted output of a stdarg argument list

```
#include <stdarg.h>
#include <stdio.h>
#include <wchar.h>

int vfwprintf(FILE *restrict stream, const wchar_t *restrict format,
              va_list arg);
int vswprintf(wchar_t *restrict ws, size_t n,
              const wchar_t *restrict format, va_list arg);
int vwprintf(const wchar_t *restrict format, va_list arg);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The *vfwprintf()*, *vswprintf()*, and *vwprintf()* prototypes are updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

vfwscanf, vswscanf, vwscanf

Wide-character formatted input of a stdarg list

```
#include <stdarg.h>
#include <stdio.h>
#include <wchar.h>

int vfwscanf(FILE *restrict stream, const wchar_t *restrict format,
             va_list arg);
int vswscanf(const wchar_t *restrict ws, const wchar_t *restrict format,
             va_list arg);
int vwscanf(const wchar_t *restrict format, va_list arg);
```

The *vfwscanf()*, *vswscanf()*, and *vwscanf()* functions are equivalent to the *fwscanf()*, *swscanf()*, and *wscanf()* functions, respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined in the **<stdarg.h>** header.

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

wait, waitpid

Wait for a child process to stop or terminate

```
#include <sys/wait.h>

pid_t wait(int *stat_loc);
pid_t waitpid(pid_t pid, int *stat_loc, int options);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The processing of the SIGCHLD signal and the [ECHILD] error is clarified.

The semantics of *WIFSTOPPED(stat_val)*, *WIFEXITED(stat_val)*, and *WIFSIGNALED(stat_val)* are defined with respect to *posix_spawn()* or *posix_spawnnp()* for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

waitid

Wait for a child process to change state

```
xSI #include <sys/wait.h>
int waitid(idtype_t idtype, id_t id, siginfo_t *infop, int options);
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes are made in this issue.

wcrtomb

Convert a wide-character code to a character (restartable)

```
#include <stdio.h>
size_t wcrtomb(char *restrict s, wchar_t wc, mbstate_t *restrict ps);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 In the DESCRIPTION, a note on using this function in a threaded application is added.

The *wcrtomb()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcscat

Concatenate two wide-character strings

```
#include <wchar.h>
wchar_t *wcscat(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The Open Group Corrigendum U040/2 is applied. In the RETURN VALUE section, *s1* is changed to *ws1*.

The *wcscat()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcschr

Wide-character string scanning operation

```
#include <wchar.h>
wchar_t *wcschr(const wchar_t *ws, wchar_t wc);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcscmp

Compare two wide-character strings

```
#include <wchar.h>
```

```
int wcscmp(const wchar_t *ws1, const wchar_t *ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcscoll

Wide-character string comparison using collating information

```
#include <wchar.h>
```

```
int wcscoll(const wchar_t *ws1, const wchar_t *ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcscpy

Copy a wide-character string

```
#include <wchar.h>
```

```
wchar_t *wcscpy(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The *wcscpy()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcscspn

Get length of a complementary wide substring

```
#include <wchar.h>
```

```
size_t wcscspn(const wchar_t *ws1, const wchar_t *ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcsftime

Convert date and time to a wide-character string

```
#include <wchar.h>
```

```
size_t wcsftime(wchar_t *restrict wcs, size_t maxsize,  
               const wchar_t *restrict format, const struct tm *restrict timeptr);
```

Derivation First released in Issue 4.

Issue 6 The *wcsftime()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcslen

Get wide-character string length

```
#include <wchar.h>

size_t wcslen(const wchar_t *ws);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcsncat

Concatenate a wide-character string with part of another

```
#include <wchar.h>

wchar_t *wcsncat(wchar_t *restrict ws1, const wchar_t *restrict ws2,
                size_t n);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The *wcsncat()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcsncmp

Compare part of two wide-character strings

```
#include <wchar.h>

int wcsncmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcsncpy

Copy part of a wide-character string

```
#include <wchar.h>

wchar_t *wcsncpy(wchar_t *restrict ws1, const wchar_t *restrict ws2,
                size_t n);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The *wcsncpy()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcspbrk

Scan wide-character string for a wide-character code

```
#include <wchar.h>

wchar_t *wcspbrk(const wchar_t *ws1, const wchar_t *ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcsrchr

Wide-character string scanning operation

```
#include <wchar.h>
```

```
wchar_t *wcsrchr(const wchar_t *ws, wchar_t wc);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcsrtombs

Convert a wide-character string to a character string (restartable)

```
#include <wchar.h>
```

```
size_t wcsrtombs(char *restrict dst, const wchar_t **restrict src,  
                size_t len, mbstate_t *restrict ps);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 In the DESCRIPTION, a note on using this function in a threaded application is added.

The *wcsrtombs()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcsspn

Get length of a wide substring

```
#include <wchar.h>
```

```
size_t wcsspn(const wchar_t *ws1, const wchar_t *ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 No functional changes are made in this issue.

wcsstr

Find a wide-character substring

```
#include <wchar.h>
```

```
wchar_t *wcsstr(const wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The *wcsstr()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcstod, wcstof, wcstold

Convert a wide-character string to a double-precision number

```
#include <wchar.h>

double wcstod(const wchar_t *restrict nptr, wchar_t **restrict endptr);
float wcstof(const wchar_t *restrict nptr, wchar_t **restrict endptr);
long double wcstold(const wchar_t *restrict nptr,
    wchar_t **restrict endptr);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *wcstod()* prototype is updated to include the **restrict** keyword.
- The *wcstof()* and *wcstold()* functions are added.
- If the correct value for *wcstod()* would cause underflow, the return value changed from 0 (as specified in Issue 5) to the smallest normalized positive number.
- The DESCRIPTION, RETURN VALUE, and APPLICATION USAGE sections are extensively updated.

ISO/IEC 9899:1999 standard, Technical Corrigendum No. 1 is incorporated.

wcstoimax, wcstoumax

Convert wide-character string to integer type

```
#include <stddef.h>
#include <inttypes.h>

intmax_t wcstoimax(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
uintmax_t wcstoumax(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
```

Derivation First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

wcstok

Split wide-character string into tokens

```
#include <wchar.h>

wchar_t *wcstok(wchar_t *restrict ws1, const wchar_t *restrict ws2,
    wchar_t **restrict ptr);
```

Derivation First released in Issue 4.

Issue 6 The *wcstok()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcstol, wcstoll

Convert a wide-character string to a long integer

```
#include <wchar.h>

long wcstol(const wchar_t *restrict nptr, wchar_t **restrict endptr,
            int base);
long long wcstoll(const wchar_t *restrict nptr,
                  wchar_t **restrict endptr, int base);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *wcstol()* prototype is updated to include the **restrict** keyword.
- The *wcstoll()* function is added.

wcstombs

Convert a wide-character string to a character string

```
#include <stdlib.h>

size_t wcstombs(char *restrict s, const wchar_t *restrict pwcs,
                size_t n);
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION states the effect of when *s* is a null pointer.
- The [EILSEQ] error condition is added.

The *wcstombs()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wcstoul, wcstoull

Convert a wide-character string to an unsigned long

```
#include <wchar.h>

unsigned long wcstoul(const wchar_t *restrict nptr,
                     wchar_t **restrict endptr, int base);
unsigned long long wcstoull(const wchar_t *restrict nptr,
                             wchar_t **restrict endptr, int base);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] error condition is added for when the value of *base* is not supported.

In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *wcstoul()* prototype is updated to include the **restrict** keyword.
- The *wcstoull()* function is added.

wcswcs

Find a wide substring (**LEGACY**)

```
xSI #include <wchar.h>
wchar_t *wcswcs(const wchar_t *ws1, const wchar_t *ws2);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 This function is marked LEGACY and may not be available on all implementations. This function was not included in the final ISO C Amendment 1 (MSE). Application developers are strongly encouraged to use the *wcsstr()* function instead.

wcswidth

Number of column positions of a wide-character string

```
xSI #include <wchar.h>
int wcswidth(const wchar_t *pwcs, size_t n);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 The Open Group Corrigendum U021/11 is applied. The function is marked as an extension.

wcsxfrm

Wide-character string transformation

```
#include <wchar.h>
size_t wcsxfrm(wchar_t *restrict ws1, const wchar_t *restrict ws2,
               size_t n);
```

Derivation First released in Issue 4. Derived from the MSE working draft.

Issue 6 In previous versions, this function was required to return -1 on error.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The *wcsxfrm()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wctob

Wide-character to single-byte conversion

```
#include <stdio.h>
#include <wchar.h>

int wctob(wint_t c);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

wctomb

Convert a wide-character code to a character

```
#include <stdlib.h>

int wctomb(char *s, wchar_t wchar);
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 In the DESCRIPTION, a note about reentrancy and thread-safety is added. A portable multi-threaded application may only safely call the function when access to the function is serialized.

wctrans

Define character mapping

```
#include <wctype.h>

wctrans_t wctrans(const char *charclass);
```

Derivation First released in Issue 5. Derived from the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

wctype

Define character class

```
#include <wctype.h>

wctype_t wctype(const char *property);
```

Derivation First released in Issue 4.

Issue 6 No functional changes are made in this issue.

wcwidth

Number of column positions of a wide-character code

```
xSI #include <wchar.h>
int wcwidth(wchar_t wc);
```

Derivation First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working draft.

Issue 6 The Open Group Corrigendum U021/12 is applied, noting that this function was removed from the final ISO C Amendment 1 (MSE). This function is marked as an extension since this function was not included in the ISO/IEC 9899:1999 standard.

wmemchr

Find a wide character in memory

```
#include <wchar.h>
```

```
wchar_t *wmemchr(const wchar_t *ws, wchar_t wc, size_t n);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

wmemcmp

Compare wide characters in memory

```
#include <wchar.h>
```

```
int wmemcmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

wmemcpy

Copy wide characters in memory

```
#include <wchar.h>
```

```
wchar_t *wmemcpy(wchar_t *restrict ws1, const wchar_t *restrict ws2,
                 size_t n);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 The *wmemcpy()* prototype is updated to include the **restrict** keyword for alignment with the ISO/IEC 9899:1999 standard.

wmemmove

Copy wide characters in memory with overlapping areas

```
#include <wchar.h>
```

```
wchar_t *wmemmove(wchar_t *ws1, const wchar_t *ws2, size_t n);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

wmemset

Set wide characters in memory

```
#include <wchar.h>
```

```
wchar_t *wmemset(wchar_t *ws, wchar_t wc, size_t n);
```

Derivation First released in Issue 5. Included for alignment with the ISO/IEC 9899:1990 standard.

Issue 6 No functional changes are made in this issue.

wordexp, wordfree

Perform word expansions

```
#include <wordexp.h>
```

```
int wordexp(const char *restrict words, wordexp_t *restrict pwordexp,  
            int flags);
```

```
void wordfree(wordexp_t *pwordexp);
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 The **restrict** keyword is added to the *wordexp()* prototype for alignment with the ISO/IEC 9899:1999 standard.

pwrite, write

Write on a file

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

```
XSI ssize_t pwrite(int fildes, const void *buf, size_t nbyte,  
                off_t offset);
```

```
ssize_t write(int fildes, const void *buf, size_t nbyte);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The DESCRIPTION states that the *write()* function does not block the thread. Previously this said “process” rather than “thread”.

The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION now states that if *write()* is interrupted by a signal after it has successfully written some data, it returns the number of bytes written. In IEEE Std 1003.1-1988 (POSIX.1), it was optional whether *write()* returned the number of bytes written, or whether it returned -1 with *errno* set to [EINTR]. This is a FIPS requirement.
- The following changes are made to support large files:
 - For regular files, no data transfer occurs past the offset maximum established in the open file description associated with the *fildes*.
 - A second [EFBIG] error condition is added.

- The [EIO] error condition is added.
- The [EPIPE] error condition is added for when a pipe has only one end open.
- The [ENXIO] optional error condition is added.

Text referring to sockets is added to the DESCRIPTION.

The following changes are made to align with the IEEE P1003.1a draft standard:

- The effect of reading zero bytes is clarified.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *write()* results are unspecified for typed memory objects.

The following error conditions are added for operations on sockets: [EAGAIN], [EWOULDBLOCK], [ECONNRESET], [ENOTCONN], and [EPIPE].

The [EIO] error is changed to “may fail”.

The [ENOBUFS] error is added for sockets.

The following error conditions are added for operations on sockets: [EACCES], [ENETDOWN], and [ENETUNREACH].

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/146 is applied, updating text in the ERRORS section from “a SIGPIPE signal is generated to the calling process” to “a SIGPIPE signal shall also be sent to the thread”.

writew

Write a vector

```
xSI #include <sys/uio.h>
    ssize_t writew(int fildev, const struct iovec *iov, int iovcnt);
```

Derivation First released in Issue 4, Version 2.

Issue 6 Split out from the *write()* reference page.

y0, y1, yn

Bessel functions of the second kind

```
xSI #include <math.h>
    double y0(double x);
    double y1(double x);
    double yn(int n, double x);
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The RETURN VALUE and ERRORS sections are reworked for alignment of the error handling with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/148 is applied, updating the RETURN VALUE and ERRORS sections. The changes are made for consistency with the general rules stated in “Treatment of Error Conditions for Mathematical Functions” in the Base Definitions volume of IEEE Std 1003.1-2001.

Utilities Migration

This chapter contains a section for each interface defined in XCU, Issue 6. Each section contains the SYNOPSIS and identifies syntax and semantic changes made to the interface in Issue 6 (if any), complete with examples where appropriate. Only changes that might affect an application programmer are identified.

ISO/IEC Terminology Changes

In order for the XCU document to also be an ISO/IEC standard, a number of terminology changes in the normative text were made, notably use of the term “shall” instead of the term “will” for requirements, and avoiding the term “must” for application requirements. Although these terminology changes affected most utilities, no functional change is intended and these changes are not further noted in this chapter.

Eight-Bit Bytes

This issue of the specification mandates that a byte has eight bits. As such, text describing behavior on systems with bytes consisting of more than eight bits has been removed. This change is not further noted in this chapter.

admin

Create and administer SCCS files (**DEVELOPMENT**)

```
XSI  admin -i[name][-n][-a login][-d flag][-e login][-f flag][-m mrlist]
      [-r rel][-t[name][-y[comment]]] newfile
      admin -n[-a login][-d flag][-e login][-f flag][-m mrlist][-t[name]]
      [-y[comment]] newfile ...
      admin [-a login][-d flag][-m mrlist][-r rel][-t[name]] file ...
      admin -h file ...
      admin -z file ...
```

Derivation First released in Issue 2.

Issue 6 The grammar is updated.

The Open Group Base Resolution bwg2001-007 is applied, adding new text to the INPUT FILES section warning that the maximum lines recorded in the file is 99999.

The Open Group Base Resolution bwg2001-009 is applied, replacing the text related to the **-h** option for the *admin* utility with the following:

Check the structure of the SCCS file and compare the newly-computed checksum with the checksum that is stored in the SCCS file. If the newly-computed checksum does not match the checksum in the SCCS file, a diagnostic message shall be written.

alias

Define or display aliases

```
UP  alias [alias-name[=string] ...]
```

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

ar

Create and maintain library archives

```
SD  ar -d[-v] archive file ...
```

```
XSI  ar -m[-abiv][posname] archive file ...
```

```
XSI  ar -p[-v] [-s] archive [file ...]
```

```
XSI  ar -q[-cv] archive file ...
```

XSI `ar -r[-cuv] [-abi] [posname] archive file ...`

XSI `ar -t[-v] [-s] archive [file ...]`

XSI `ar -x[-v] [-sCT] archive [file ...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the Software Development Utilities option.

The STDOUT description is changed for the `-v` option to align with the IEEE P1003.2b draft standard.

The `TZ` entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #198 is applied, changing the description to consistently use “file” to refer to a file in the file system hierarchy, “archive” to refer to the archive being operated upon by the `ar` utility, and “file in the archive” to refer to a copy of a file that is contained in the archive.

asa

Interpret carriage-control characters

FR `asa [file ...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the FORTRAN Runtime Utilities option.

No functional changes in this issue.

at

Execute commands at a later time

UP `at [-m] [-f file] [-q queuename] -t time_arg`

`at [-m] [-f file] [-q queuename] timespec ...`

`at -r at_job_id ...`

`at -l -q queuename`

`at -l [at_job_id ...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- If `-m` is not used, the job’s standard output and standard error are provided to the user by mail.

The effects of using the `-q` and `-t` options as defined in the IEEE P1003.2b draft standard are specified.

awk

Pattern scanning and processing language

```
awk [-F ERE] [-v assignment] ... program [argument ...]
```

```
awk [-F ERE] -f progfile ... [-v assignment] ... [argument ...]
```

Derivation First released in Issue 2.

Issue 6 The *awk* utility is aligned with the IEEE P1003.2b draft standard.

IEEE PASC Interpretation 1003.2 #211 is applied, adding the sentence “An occurrence of two consecutive backslashes shall be interpreted as just a single literal backslash character.” into the description of the **sub** string function.

basename

Return non-directory portion of a pathname

```
basename string [suffix]
```

Derivation First released in Issue 2.

Issue 6 IEEE PASC Interpretation 1003.2 #164 is applied, adding a new step 1. as follows:

1. If *string* is a null string, it is unspecified whether the resulting string is '.' or a null string. In either case, skip steps 2 through 6.

batch

Schedule commands to be executed in a batch queue

UP

```
batch
```

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The NAME is changed to align with the IEEE P1003.2b draft standard.

bc

Arbitrary-precision arithmetic language

```
bc [-l] [file ...]
```

Derivation First released in Issue 4.

Issue 6 Updated to align with the IEEE P1003.2b draft standard, which included resolution of several interpretations of ISO/IEC 9945-2: 1993 (POSIX-2).

bg

Run jobs in the background

UP `bg [job_id ...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The JC margin marker on the SYNOPSIS is removed since support for Job Control is mandatory in this issue. This is a FIPS requirement.

No functional changes in this issue.

c99

Compile standard C programs

CD `c99 [-c] [-D name[=value]]... [-E] [-g] [-I directory] ... [-L directory] ... [-o outfile] [-Ooptlevel] [-s] [-U name]... operand ...`

Derivation First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

Issue 6 This is a new utility created in Issue 6 to allow applications a front end to a compiler conforming to the ISO/IEC 9899:1999 standard.

On systems providing POSIX Conformance (see XBD, Chapter 2, Conformance), *c99* is required only with the C-Language Development option; XSI-conformant systems always provide *c99*.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/12 is applied, correcting the EXTENDED DESCRIPTION of **-I c** and **-I m**. Previously, the text did not take into account the presence of the *c99* math headers.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/13 is applied, changing the reference to the **libxnet** library to **libxnet.a**.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/5 is applied, updating the OPTIONS section, so that the names of files contained in the directory specified by the **-L** option are not assumed to end in the **.a** suffix. The set of library prefixes is also updated.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/6 is applied, removing the lead underscore from the `POSIX_V6_WIDTH_RESTRICTED_ENVS` variable in the EXTENDED DESCRIPTION and the EXAMPLES sections.

cal

Print a calendar

XSI `cal [[month] year]`

Derivation First released in Issue 2.

Issue 6 The DESCRIPTION is updated to allow for traditional behavior for years before the adoption of the Gregorian calendar.

cat

Concatenate and print files

`cat [-u][file ...]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

cd

Change the working directory

`cd [-L] [-P] [directory]`

`cd -`

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The `cd -`, `PWD`, and `OLDPWD` are added.

The `-L` and `-P` options are added to align with the IEEE P1003.2b draft standard. This also includes the introduction of a new description to include the effect of these options.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/14 is applied, changing the SYNOPSIS to make it clear that the `-L` and `-P` options are mutually-exclusive.

cflow

Generate a C-language flowgraph (**DEVELOPMENT**)

XSI `cflow [-r][-d num][-D name[=def]] ... [-i incl][-I dir] ...
[-U dir] ... file ...`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

chgrp

Change the file group ownership

```
chgrp -hR group file ...
```

```
chgrp -R [-H | -L | -P ] group file ...
```

Derivation First released in Issue 2.

Issue 6 New options **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These options affect the processing of symbolic links.

IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS section to “Default.”.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/15 is applied, changing the SYNOPSIS to make it clear that **-h** and **-R** are optional.

chmod

Change the file modes

```
chmod [-R] mode file ...
```

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Octal modes have been kept and made mandatory despite being marked obsolescent in ISO/IEC 9945-2: 1993 (POSIX-2).

IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS section to “Default.”.

The Open Group Base Resolution bwg2001-010 is applied (as XSI shaded text) as follows:

- The table is extended to include the entry 1000 S_ISVTX.
- The **t perm** symbol is added.
- The following text is added to the EXTENDED DESCRIPTION:

The **perm** symbol **t** shall specify the S_ISVTX bit and shall apply to directories only. The effect when using it with any other file type is unspecified. It can be used with the **who** symbols **o**, **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u** or **g** in conjunction with the **perm** symbol **t**; it shall be ignored for **u** and **g**.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/16 is applied, changing the XSI shaded text in the EXTENDED DESCRIPTION from:

“The **perm** symbol **t** shall specify the S_ISVTX bit and shall apply to directories only. The effect when using it with any other file type is unspecified. It can be used with the **who** symbols **o**, **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u** or **g** in conjunction with the **perm** symbol **t**; it shall be ignored for **u** and **g**.”

to:

“The **perm** symbol **t** shall specify the S_ISVTX bit. When used with a file of type directory, it can be used with the **who** symbol **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u**, **g**, or **o** in conjunction with the **perm** symbol **t**, but the meaning of these combinations is unspecified. The effect when using the **perm** symbol **t** with any file type other than directory is unspecified.”

This change is to permit historical behavior.

chown

Change the file ownership

```
chown -hR owner[:group] file ...
```

```
chown -R [-H | -L | -P ] owner[:group] file ...
```

Derivation First released in Issue 2.

Issue 6 New options **-h**, **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These options affect the processing of symbolic links.

IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS section is changed to “Default.”.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/17 is applied, changing the SYNOPSIS to make it clear that **-h** and **-R** are optional.

cksum

Write file checksums and sizes

```
cksum [file ...]
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

cmp

Compare two files

```
cmp [ -l | -s ] file1 file2
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

comm

Select or reject lines common to two files

```
comm [-123] file1 file2
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

command

Execute a simple command

```
command [-p] command_name [argument ...]
```

UP

```
command [ -v | -V ] command_name
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

compress

Compress data

XSI

```
compress [-fv][-b bits][file ...]
```

```
compress [-cfv][-b bits][file]
```

Derivation First released in Issue 4.

Issue 6 An error case is added for systems not supporting adaptive Lempel-Ziv coding.

cp

Copy files

```
cp [-fip] source_file target_file
```

```
cp [-fip] source_file ... target
```

```
cp -R [-H | -L | -P][-fip] source_file ... target
```

OB

```
cp -r [-H | -L | -P][-fip] source_file ... target
```

Derivation First released in Issue 2.

Issue 6 The `-r` option is marked obsolescent.

The new options `-H`, `-L`, and `-P` are added to align with the IEEE P1003.2b draft standard. These options affect the processing of symbolic links.

IEEE PASC Interpretation 1003.2 #194 is applied, adding a description of the `-P` option.

crontab

Schedule periodic background work

UP

```
crontab [file]
```

```
crontab [ -e | -l | -r ]
```

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

csplit

Split files based on context

UP `csplit [-ks][-f prefix][-n number] file arg1 ...argn`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

The description of regular expression operands is changed to align with the IEEE P1003.2b draft standard.

ctags

Create a tags file (**DEVELOPMENT, FORTRAN**)

UP `ctags [-a][-f tagsfile] pathname ...`

`ctags -x pathname ...`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The OUTPUT FILES section is changed to align with the IEEE P1003.2b draft standard.

IEEE PASC Interpretation 1003.2 #168 is applied, changing “create” to “write” in the DESCRIPTION.

cut

Cut out selected fields of each line of a file

`cut -b list [-n] [file ...]`

`cut -c list [file ...]`

`cut -f list [-d delim][-s][file ...]`

Derivation First released in Issue 2.

Issue 6 The OPTIONS section is changed to align with the IEEE P1003.2b draft standard.

cxref

Generate a C-language program cross-reference table (**DEVELOPMENT**)

XSI `cxref [-cs][-o file][-w num] [-D name[=def]]...[-I dir]...
[-U name]... file ...`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

date

Write the date and time

date [-u] [+format]

xSI date [-u] *mmddhhmm* [cc]yy

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The setting of system date and time is described, including how to interpret two-digit year values if a century is not given.
- The %EX modified conversion specification is added.

The Open Group Corrigendum U048/2 is applied, correcting the examples.

The DESCRIPTION is updated to refer to conversion specifications, instead of field descriptors for consistency with the *LC_TIME* category.

A clarification is made such that the current year is the default if the yy argument is omitted when setting the system date and time.

dd

Convert and copy a file

dd [operand ...]

Derivation First released in Issue 2.

Issue 6 Changes are made to *swab* conversion and SIGINT handling to align with the IEEE P1003.2b draft standard.

IEEE PASC Interpretation 1003.2 #209 is applied, clarifying the interaction between *dd of=file* and **conv=notrunc**.

delta

Make a delta (change) to an SCCS file (**DEVELOPMENT**)

xSI delta [-nps] [-g list] [-m mrlist] [-r SID] [-y[comment]] file...

Derivation First released in Issue 2.

Issue 6 The APPLICATION USAGE section is added.

The Open Group Base Resolution bwg2001-007 is applied as follows:

- The use of '-' as a file argument is clarified.
- The use of STDIN is added.
- The ASYNCHRONOUS EVENTS section is updated to remove the implicit requirement that implementations re-signal themselves when catching a normally fatal signal.

- New text is added to the INPUT FILES section warning that the maximum lines recorded in the file is 99 999.

New text is added to the EXTENDED DESCRIPTION and APPLICATION USAGE sections regarding how the system date and time may be taken into account, and the *TZ* environment variable is added to the ENVIRONMENT VARIABLES section as per The Open Group Base Resolution bwg2001-007.

df

Report free disk space

UP XSI `df [-k] [-P|-t] [file...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

diff

Compare two files

`diff [-c|-e|-f|-C n][-br] file1 file2`

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The `-f` option is added.

The output format for `-c` or `-C` format is changed to align with changes to the IEEE P1003.2b draft standard resulting from IEEE PASC Interpretation 1003.2 #71.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/20 is applied, changing the STDOUT section. This changes the specification of `diff -c` so that it agrees with existing practice when contexts contain zero lines or one line.

dirname

Return the directory portion of pathname

`dirname string`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

du

Estimate file space usage

UP `du [-a | -s] [-kx] [-H | -L] [file ...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

This utility is reinstated, as the LEGACY marking was incorrect in Issue 5.

The obsolescent `-r` option has been removed.The Open Group Corrigendum U025/3 is applied. The `du` utility had incorrectly been marked LEGACY.The `-H` and `-L` options for symbolic links are added as described in the IEEE P1003.2b draft standard.**echo**

Write arguments to standard output

`echo [string ...]`

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- A set of character sequences is defined as *string* operands.
- `LC_CTYPE` is added to the list of environment variables affecting `echo`.
- In the OPTIONS section, implementations shall not support any options.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/21 is applied, so that the `echo` utility can accommodate historical BSD behavior.**ed**

Edit text

`ed [-p string] [-s] [file]`

Derivation First released in Issue 2.

Issue 6 The obsolescent single-minus form has been removed.

A second APPLICATION USAGE note has been added.

The Open Group Corrigendum U025/2 is applied, correcting the description of the Edit section.

The `ed` utility is updated to align with the IEEE P1003.2b draft standard. This includes addition of the treatment of the SIGQUIT signal, changes to `ed` addressing, and changes to processing when end-of-file is detected and when terminal disconnect is detected.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/22 is applied, adding the text: “Any line modified by the *command list* shall be unmarked.” to the **G** command. This change corresponds to a similar change made to the **g** command in the first version of IEEE Std 1003.1-2001.1-2001.

env

Set the environment for command invocation

```
env [-i][name=value]... [utility [argument...]]
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

ex

Text editor

UP `ex [-rR][-l][-s | -v][-c command][-t tagstring][-w size][file ...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The obsolescent SYNOPSIS is removed, removing the **+command** and **-** options.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the *map* command description, the sequence *#digit* is added.
- The **directory**, **edcompatible**, **redraw**, and **slowopen** edit options are added.

The *ex* utility is extensively changed for alignment with the IEEE P1003.2b draft standard. This includes changes as a result of the IEEE PASC Interpretations 1003.2 #31, #38, #49, #50, #51, #52, #55, #56, #57, #61, #62, #63, #64, #65, and #78.

The **-I** option is removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/23 is applied, correcting a URL.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/8 is applied, making an editorial correction in the EXTENDED DESCRIPTION.

expand

Convert tabs to spaces

UP `expand [-t tablist][file ...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

The obsolescent SYNOPSIS is removed.

The *LC_CTYPE* environment variable description is updated to align with the IEEE P1003.2b draft standard.

expr

Evaluate arguments as an expression

expr operand

Derivation First released in Issue 2.

Issue 6 The *expr* utility is aligned with the IEEE P1003.2b draft standard, to include resolution of IEEE PASC Interpretation 1003.2 #104.

false

Return false value

false

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

fc

Process the command history list

UP `fc [-r][-e editor] [first[last]]`

`fc -l[-nr] [first[last]]`

`fc -s[old=new] [first]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to “directory referred to by the *HOME* environment variable”.

fg

Run jobs in the foreground

UP `fg [job_id]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

The JC marking is removed from the SYNOPSIS since job control is mandatory in this issue.

No functional changes in this issue.

file

Determine file type

UP `file [-dhi][-M file][-m file] file ...`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

Options and an EXTENDED DESCRIPTION are added as specified in the IEEE P1003.2b draft standard.

IEEE PASC Interpretation 1003.2 #178 is applied, adding the following text to the EXTENDED DESCRIPTION:

x The test shall succeed if the file is large enough to contain a value of the type specified starting at the offset specified.

IEEE PASC Interpretation 1003.2 #192 is applied, replacing the second paragraph in the EXTENDED DESCRIPTION within the *value* item with:

If the specifier from the type field is *s* or **string**, then interpret the value as a string. Otherwise, interpret it as a number. If the value is a string, then the test shall succeed only when a string value exactly matches the bytes from the file.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/25 is applied, making major changes to address ambiguities raised in defect reports.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/26 is applied, making it clear in the OPTIONS section that the **-m**, **-d**, and **-M** options do not comply with Guideline 11 of the Utility Syntax Guidelines.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/10 is applied, clarifying the specification characters.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/11 is applied, allowing application writers to create portable magic files that can match characters in strings, and allowing common extensions found in existing implementations.

find

Find files

`find [-H | -L] path ... [operand_expression ...]`

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-perm [-]onum** primary is supported.

The *find* utility is aligned with the IEEE P1003.2b draft standard, to include processing of symbolic links and changes to the description of the **atime**, **ctime**, and **mtime** operands.

IEEE PASC Interpretation 1003.2 #210 is applied, extending the **-exec** operand.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/13 is applied, updating the RATIONALE section to be consistent with the normative text.

fold

Filter for folding lines

```
fold [-bs] [-w width] [file...]
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

fort77

FORTRAN compiler (**FORTRAN**)

```
FD fort77 [-c] [-g] [-L directory]... [-O optlevel] [-o outfile] [-s] [-w]
  operand...
```

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the FORTRAN Development Utilities option. This utility may not be present on all systems supporting the Single UNIX Specification.

No functional changes in this issue.

fuser

List process IDs of all processes that have one or more files open

```
XSI fuser [ -cfu ] file ...
```

Derivation First released in Issue 5.

Issue 6 No functional changes in this issue.

gencat

Generate a formatted message catalog

```
XSI gencat catfile msgfile...
```

Derivation First released in Issue 3.

Issue 6 No functional changes in this issue.

get

Get a version of an SCCS file (**DEVELOPMENT**)

```
XSI get [-begkmnlIpst] [-c cutoff] [-i list] [-r SID] [-x list] file...
```

Derivation First released in Issue 2.

Issue 6 The obsolescent SYNOPSIS is removed, removing the `-lp` option.

The Open Group Corrigendum U025/5 is applied, correcting text in the OPTIONS section.

The Open Group Corrigendum U048/1 is applied.

The EXTENDED DESCRIPTION section is updated to make partial SID handling unspecified, reflecting common usage, and to clarify SID ranges as per The Open Group Base Resolution bwg2001-007.

The Open Group Interpretation PIN4C.00014 is applied, adding the string "lines" to the required output on *stdout*.

New text is added to the EXTENDED DESCRIPTION and APPLICATION USAGE sections regarding how the system date and time may be taken into account, and the *TZ* environment variable is added to the ENVIRONMENT VARIABLES section as per The Open Group Base Resolution bwg2001-007.

getconf

Get configuration values

```
getconf [ -v specification ] system_var
```

```
getconf [ -v specification ] path_var pathname
```

Derivation First released in Issue 4.

Issue 6 The Open Group Corrigendum U029/4 is applied, correcting the example command in the last paragraph of the OPTIONS section.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Operands are added to determine supported programming environments.

This reference page is updated for alignment with the ISO/IEC 9899:1999 standard. Specifically, new macros for *c99* programming environments are introduced.

XSI marked *system_var* (XBS5_*) values are marked LEGACY.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/27 is applied, correcting the descriptions of *path_var* and *system_var* in the OPERANDS section.

getopts

Parse utility options

```
getopts optstring name [arg...]
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

grep

Search a file for a pattern

```
grep [-E| -F][-c| -l| -q][-insvx] -e pattern_list...
    [-f pattern_file...][file...]
```

```
grep [-E| -F][-c| -l| -q][-insvx][-e pattern_list...]
    -f pattern_file... [file...]
```

```
grep [-E| -F][-c| -l| -q][-insvx] pattern_list[file...]
```

Derivation First released in Issue 2.

Issue 6 The Open Group Corrigendum U029/5 is applied, correcting the SYNOPSIS.

No functional changes in this issue.

hash

Remember or report utility locations

XSI `hash [utility...]`

`hash -r`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

head

Copy the first part of files

```
head [-n number][file...]
```

Derivation First released in Issue 4.

Issue 6 The obsolescent **-number** form is withdrawn.

iconv

Codeset conversion

```
iconv [-cs] -f fromcode -t to code [file ...]
```

```
iconv -l
```

Application writers should note that the *iconv* utility can be used portably only when it is provided with two charmap files as option-arguments. This is because a single charmap provided by the user cannot reliably be joined with the names in a system-provided character set description.

Derivation First released in Issue 3.

Issue 6 This utility has been rewritten to align with the IEEE P1003.2b draft standard. Specifically, the ability to use charmap files for conversion has been added.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/29 is applied, making changes to address inconsistencies with the *iconv()* function in the System Interfaces volume of IEEE Std 1003.1-2001.

id

Return user identity

```
id [user]
```

```
id -G[-n] [user]
```

```
id -g[-nr] [user]
```

```
id -u[-nr] [user]
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

ipcrm

Remove an XSI message queue, semaphore set, or shared memory segment identifier

```
XSI ipcrm [ -q msgid | -Q msgkey | -s semid | -S semkey |  
      -m shmid | -M shmkey ] ...
```

Derivation First released in Issue 5.

Issue 6 No functional changes in this issue.

ipcs

Report XSI interprocess communication facilities status

```
XSI ipcs [-qms] [-a | -bcopt]
```

Derivation First released in Issue 5.

Issue 6 The Open Group Corrigendum U020/1 is applied, correcting the SYNOPSIS.

The Open Group Corrigenda U032/1 and U032/2 are applied, clarifying the output format.

The Open Group Base Resolution bwg98-004 is applied, changing NSEMS (**a,t**) to NSEMS (**a,b**) in the STDOUT section.

jobs

Display status of jobs in the current session

```
UP jobs [-l | -p] [job_id...]
```

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the

Single UNIX Specification.

The JC shading is removed as job control is mandatory in this issue.

No functional changes in this issue.

join

Relational database operator

```
join [-a file_number | -v file_number][-e string][-o list][-t char]  
    [-1 field][-2 field] file1 file2
```

Derivation First released in Issue 2.

Issue 6 The obsolescent `-j` options and the multi-argument `-o` option are withdrawn in this issue.

kill

Terminate or signal processes

```
kill -s signal_name pid ...
```

```
kill -l [exit_status]
```

XSI `kill [-signal_name] pid ...`

`kill [-signal_number] pid ...`

Derivation First released in Issue 2.

Issue 6 The obsolescent versions of the SYNOPSIS were turned into non-obsolescent features of the XSI option, corresponding to a similar change in the `trap` special built-in.

lex

Generate programs for lexical tasks (**DEVELOPMENT**)

CD `lex [-t][-n|-v][file ...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the C-Language Development Utilities option.
The obsolescent `-c` option is withdrawn in this issue.

link

Call `link()` function

XSI `link file1 file2`

Derivation First released in Issue 5.

Issue 6 No functional changes in this issue.

ln

Link files

```
ln [-fs] source_file target_file
```

```
ln [-fs] source_file ... target_dir
```

Derivation First released in Issue 2.

Issue 6 The *ln* utility is updated to include symbolic link processing as defined in the IEEE P1003.2b draft standard.

locale

Get locale-specific information

```
locale [-a| -m]
```

```
locale [-ck] name...
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

localedef

Define locale environment

```
localedef [-c][-f charmap][-i sourcefile][-u code_set_name] name
```

Derivation First released in Issue 4.

Issue 6 The *-u* option is added, as specified in the IEEE P1003.2b draft standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC2/D6/15 is applied, rewording text in the OPERANDS section describing the ability to create public locales.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/16 is applied, making the text consistent with the descriptions of **WIDTH** and **WIDTH_DEFAULT** in the Base Definitions volume of IEEE Std 1003.1-2001.

logger

Log messages

```
logger string ...
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

logname

Return the user's login name

```
logname
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

lp

Send files to a printer

```
lp [-c] [-d dest] [-n copies] [-msw] [-o option]... [-t title] [file...]
```

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the requirement to associate a unique request ID, and the normal generation of a banner page is added.
- In the OPTIONS section:
 - The **-d** *dest* description is expanded, but references to *lpstat* are removed.
 - The **-m**, **-o**, **-s**, **-t**, and **-w** options are added.
- In the ENVIRONMENT VARIABLES section, *LC_TIME* may now affect the execution.
- The STDOUT section is added.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

ls

List directory contents

```
xsi ls [-CFRacdilqrtul] [-H | -L ] [-fgmnopsx] [file...]
```

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the **-F** option, other symbols are allowed for other file types.

Treatment of symbolic links is added, as defined in the IEEE P1003.2b draft standard.

The Open Group Base Resolution bwg2001-010 is applied (as XSH shaded text) as follows:

T If in *<other permissions>* and the file is a directory, search permission is not granted to others, and the restricted deletion flag is set.

t If in *<other permissions>* and the file is a directory, search permission is granted to others, and the restricted deletion flag is set.

m4

Macro processor (**DEVELOPMENT**)

```
xsi m4 [-s] [-D name[=val]]... [-U name]... file...
```

Derivation First released in Issue 2.

Issue 6 In the EXTENDED DESCRIPTION, the **eval** text is updated to include a **'&'** character in the excepted list.

The EXTENDED DESCRIPTION of **divert** is updated to clarify that there are only nine diversion buffers.

The Open Group Base Resolution bwg2000-006 is applied, adding the following text to the end of the last paragraph before the list of built-in macros in the EXTENDED DESCRIPTION:

Except for the first argument to the **eval** built-in macro, all numeric built-in macro arguments shall be interpreted as decimal values. The string values produced as the defining text of the **decr**, **divnum**, **incr**, **index**, **len**, and **sysval** built-in macros shall be in the form of a decimal-constant as defined in the C language.

mailx

Process messages

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-F** option is added.
- The **allnet**, **debug**, and **sendwait** internal variables are added.
- The **C**, **ec**, **fo**, **F**, and **S** *mailx* commands are added.

In the DESCRIPTION and ENVIRONMENT VARIABLES sections, text stating “*HOME* directory” is replaced by “directory referred to by the *HOME* environment variable”.

The *mailx* utility is aligned with the IEEE P1003.2b draft standard, which included various clarifications to resolve IEEE PASC Interpretations submitted for ISO/IEC 9945-2:1993 (POSIX-2). In particular, the changes here address IEEE PASC Interpretations 1003.2 #10, #11, #103, #106, #108, #114, #115, #122, and #129.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/32 is applied, applying a change to the EXTENDED DESCRIPTION, raised by IEEE PASC Interpretation 1003.2 #122, which was overlooked in the first version of IEEE Std 1003.1-2001.1-2001.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/17 is applied, updating the EXTENDED DESCRIPTION (Internal Variables in *mailx*). The system start-up file is changed from “implementation-defined” to “unspecified” for consistency with other text in the EXTENDED DESCRIPTION.

make

Maintain, update, and regenerate groups of programs (**DEVELOPMENT**)

```
SD make [-einqrst][-f makefile...] [-k | -S][macro=value]...  
    [target_name...]
```

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the Software Development Utilities option.
The Open Group Corrigendum U029/1 is applied, correcting a typographical error in the SPECIAL TARGETS section.

In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.

It is specified whether the command line is related to the makefile or to the *make* command, and the macro processing rules are updated to align with the IEEE P1003.2b draft standard.

PASC Interpretation 1003.2 #193 is applied, clarifying the wording regarding macro expansion and evaluation of macros.

man

Display system documentation

`man [-k] name...`

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

mesg

Permit or deny messages

UP `mesg [y|n]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

mkdir

Make directories

`mkdir [-p][-m mode] dir...`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

mkfifo

Make FIFO special files

`mkfifo [-m mode] file...`

Derivation First released in Issue 3.

Issue 6 No functional changes in this issue.

more

Display files on a page-by-page basis

UP `more [-ceisu][-n number][-p command][-t tagstring][file ...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The obsolescent SYNOPSIS is removed.

The utility has been extensively reworked for alignment with the IEEE P1003.2b draft standard:

- Changes have been made as a result of IEEE PASC Interpretations 1003.2 #37 and #109.
- The *more* utility should be able to handle underlined and emboldened displays of characters that are wider than a single column position.

mv

Move files

`mv [-fi] source_file target_file`

`mv [-fi] source_file... target_file`

Derivation First released in Issue 2.

Issue 6 The *mv* utility is changed to describe processing of symbolic links as specified in the IEEE P1003.2b draft standard.

The APPLICATION USAGE section is added.

newgrp

Change to a new group

UP `newgrp [-l][group`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The obsolescent SYNOPSIS is removed.

The text describing supplemental groups is no longer conditional on {NGROUPS_MAX} being greater than 1. This is because {NGROUPS_MAX} now has a minimum value of 8. This is a FIPS requirement.

nice

Invoke a utility with an altered nice value

UP `nice [-n increment] utility [argument...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The obsolescent SYNOPSIS is removed.

nl

Line numbering filter

XSI `nl [-p][-b type][-d delim][-f type][-h type][-i incr][-l num][-n format]
[-s sep][-v startnum][-w width][file]`

Derivation First released in Issue 2.

Issue 6 The obsolescent behavior allowing the options to be intermingled with the optional *file* operand is removed.

nm

Write the name list of an object file (**DEVELOPMENT**)

UP SD XSI `nm [-APv][-efox][-g | -u][-t format] file...`

Derivation First released in Issue 2.

Issue 6 This utility is marked as supported when both the User Portability Utilities option and the Software Development Utilities option are supported.

nohup

Invoke a utility immune to hangups

`nohup utility [argument...]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

od

Dump files in various formats

`od [-v][-A address_base][-j skip][-N count][-t type_string]
[file...]`

XSI `od [-bcdosx][file] [[+]offset[.][b]]`

Derivation First released in Issue 2.

Issue 6 The *od* utility is changed to remove the assumption that **short** was a two-byte entity, as per the revisions in the IEEE P1003.2b draft standard.

paste

Merge corresponding or subsequent lines of files

```
paste [-s][-d list] file...
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

patch

Apply changes to files

```
UP patch [-blNR][ -c| -e| -n][-d dir][-D define][-i patchfile]
      [-o outfile][-p num][-r rejectfile][file]
```

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The description of the **-D** option and the steps in Filename Determination (in the EXTENDED DESCRIPTION) are changed to match historical practice as defined in the IEEE P1003.2b draft standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/34 is applied, clarifying the way that the *patch* utility performs **ifdef** selection for the **-D** option.

pathchk

Check pathnames

```
pathchk [-p] pathname...
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

pax

Portable archive interchange

```
pax [-cdnv][-H|-L][-f archive][-s replstr]...[pattern...]
```

```
pax -r[-cdiknuv][-H|-L][-f archive][-o options]...[-p string]...
    [-s replstr]...[pattern...]
```

```
pax -w[-dituvX][-H|-L][-b blocksize][[-a][-f archive][-o options]...
    [-s replstr]...[-x format][file...]
```

```
pax -r -w[-diklntuvX][-H|-L][-p string]...[-s replstr]...
    [file...] directory
```

Derivation First released in Issue 4.

Issue 6 The *pax* utility is aligned with the IEEE P1003.2b draft standard:

- Support has been added for symbolic links in the options and interchange formats.
- A new format has been devised, based on extensions to **ustar**.
- References to the “extended” *tar* and *cpio* formats derived from IEEE Std 1003.1-1990 (POSIX.1) have been changed to remove the “extended” adjective because this could cause confusion with the extended *tar* header added in this revision. (All references to *tar* are actually to **ustar**).

IEEE PASC Interpretation 1003.2 #168 is applied, clarifying that *mkdir()* and *mkfifo()* calls can ignore an [EEXIST] error when extracting an archive.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #180 is applied, clarifying how extracted files are created when in **read** mode.

IEEE PASC Interpretation 1003.2 #181 is applied, clarifying the description of the **-t** option.

IEEE PASC Interpretation 1003.2 #195 is applied.

IEEE PASC Interpretation 1003.2 #206 is applied, clarifying the handling of links for the **-H**, **-L**, and **-I** options.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/35 is applied, adding the process ID of the *pax* process into certain fields. This change provides a method for the implementation to ensure that different instances of *pax* extracting a file named **/a/b/foo** will not collide when processing the extended header information associated with **foo**.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/36 is applied, changing **-xB** to **-x pax** in the OPTIONS section.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/20 is applied, updating the SYNOPSIS to be consistent with the normative text.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/21 is applied, updating the DESCRIPTION to describe the behavior when files to be linked are symbolic links and the system is not capable of making hard links to symbolic links.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/22 is applied, updating the OPTIONS section to describe the behavior for how multiple **-odelete=pattern** options are to be handled.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/23 is applied, updating the **write** option within the OPTIONS section.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/24 is applied, adding a paragraph into the OPTIONS section that states that specifying more than one of the mutually-exclusive options (**-H** and **-L**) is not considered an error and that the last option specified will determine the behavior of the utility.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/25 is applied, removing the *ctime* paragraph within the EXTENDED DESCRIPTION. There is a contradiction in the definition of the *ctime* keyword for the *pax* extended header, in that the *st_ctime* member of the **stat** structure does not refer to a file creation time. No field in the standard **stat** structure from **<sys/stat.h>** includes a file creation time.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/26 is applied, making it clear that *typeflag* 1 (**ustar** Interchange Format) applies not only to files that are hard-linked, but also to files that are aliased via symlinks.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/27 is applied, clarifying the *cpio c_nlink* field.

pr

Print files

```
xSI pr [+page] [-column] [-adFmrt] [-e[char][gap]] [-h header] [-i[char][gap]]
      [-l lines] [-n[char][width]] [-o offset] [-s[char]] [-w width] [-fp]
      [file...]
```

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-p** option is added.

printf

Write formatted output

```
printf format[argument...]
```

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

prs

Print an SCCS file (**DEVELOPMENT**)

```
xSI prs [-a] [-d dataspec] [-r[SID]] file...
```

```
xSI prs [ -e | -l ] -c cutoff [-d dataspec] file...
```

```
xSI prs [ -e | -l ] -r[SID] [-d dataspec] file...
```

Derivation First released in Issue 2.

Issue 6 The normative text is reworded to emphasize the term “shall” for implementation requirements. No functional change is intended by this change.

The Open Group Base Resolution bwg2001-007 is applied, updating the table in STDOUT with a note that line statistics are capped at 99 999 for the **:Li:**, **:Ld:**, **:Lu:**, and **:DL:** keywords.

The Open Group Interpretation PIN4C.00009 is applied, clarifying that under Data Keywords in the STDOUT section, the *dataspec* for each selected delta does not include a trailing <newline>. The *dataspec* is thus now:

```
:Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:
```

ps

Report process status

UP XSI `ps [-aA][-defl][-G grouplist][-o format]...[-p proclist][-t termlist]
[-U userlist][-g grouplist][-n namelist][-u userlist]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

pwd

Return working directory name

`pwd [-L | -P]`

Derivation First released in Issue 2.

Issue 6 The **-P** and **-L** options are added to describe actions relating to symbolic links as specified in the IEEE P1003.2b draft standard.

qalter

Alter batch job

BE `qalter [-a date_time][-A account_string][-c interval][-e path_name]
[-h hold_list][-j join_list][-k keep_list][-l resource_list]
[-m mail_options][-M mail_list][-N name][-o path_name]
[-p priority][-r y|n][-S path_name_list][-u user_list]
job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #182 is applied, clarifying the description of the **-a** option.

qdel

Delete batch jobs

BE `qdel job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qhold

Hold batch jobs

BE `qhold [-h hold_list] job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qmove

Move batch jobs

BE `qmove destination job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qmsg

Send message to batch jobs

BE `qmsg [-E][-O] message_string job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qrerun

Rerun batch jobs

BE `qrerun job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qrIs

Release batch jobs

BE `qrIs [-h hold_list] job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qselect

Select batch jobs

BE `qselect [-a [op]date_time][-A account_string][-c [op]interval]
[-h hold_list][-l resource_list][-N name][-p [op]priority]
[-q destination][-r y|n][-s states][-u user_list]`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 No functional changes in this issue.

qsig

Signal batch jobs

BE `qsig [-s signal] job_identifier ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *LC_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

qstat

Show status of batch jobs

BE `qstat [-f] job_identifier ...``qstat -Q [-f] destination ...``qstat -B [-f] server_name ...`

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Utilities Migration

Issue 6 IEEE PASC Interpretation 1003.2 #191 is applied, removing the following ENVIRONMENT VARIABLES listed as affecting *qstat*: *COLUMNS*, *LINES*, *LOGNAME*, *TERM*, and *TZ*.

The *LC_TIME* entry is also removed from the ENVIRONMENT VARIABLES section.

qsub

Submit a script

```
BE qsub [-a date_time] [-A account_string] [-c interval]
    [-C directive_prefix] [-e path_name] [-h] [-j join_list] [-k keep_list]
    [-m mail_options] [-M mail_list] [-N name]
    [-o path_name] [-p priority] [-q destination] [-r y|n]
    [-S path_name_list] [-u user_list] [-v variable_list] [-V]
    [-z] [script]
```

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

Derivation Derived from IEEE Std 1003.2d-1994.

Issue 6 The *-I* option has been removed as there is no portable description of the resources that are allowed or required by the batch job.

read

Read a line from standard input

```
read [-r] var...
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

renice

Set nice values of running processes

```
UP renice -n increment [-g | -p | -u] ID ...
```

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

The obsolescent forms of the SYNOPSIS are removed.

Text previously conditional on *POSIX_SAVED_IDS* is mandatory in this issue. This is a FIPS requirement.

rm

Remove directory entries

```
rm [-fiRr] file...
```

Derivation First released in Issue 2.

Issue 6 Text is added to clarify actions relating to symbolic links as specified in the IEEE P1003.2b draft standard.

rmdel

Remove a delta from an SCCS file (**DEVELOPMENT**)

```
xSI rmdel -r SID file...
```

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

rmdir

Remove directories

```
rmdir [-p] dir...
```

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

sact

Print current SCCS file-editing activity (**DEVELOPMENT**)

```
xSI sact file...
```

Derivation First released in Issue 2.

Issue 6 The normative text is reworded to emphasize the term “shall” for implementation requirements. No functional change is intended by this change.

sccs

Front end for the SCCS subsystem (**DEVELOPMENT**)

```
xSI sccs [-r][-d path][-p path] command [options...][operands...]
```

Derivation First released in Issue 4.

Issue 6 In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.

sed

Stream editor

```
sed [-n] script[file...]
```

```
sed [-n][-e script]...[-f script_file]...[file...]
```

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Implementations are required to support at least ten *wfile* arguments in an editing command.

The EXTENDED DESCRIPTION is changed to align with the IEEE P1003.2b draft standard.

IEEE PASC Interpretation 1003.2 #190 is applied.

IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the meaning of the backslash escape sequences in a replacement string for a BRE.

sh

Shell, the standard command language interpreter

```
sh [-abCefhimnuvx][-o option][+abCefhimnuvx][+o option]  
[command_file [argument...]]
```

```
sh -c[-abCefhimnuvx][-o option][+abCefhimnuvx][+o option]command_string  
[command_name [argument...]]
```

```
sh -s[-abCefhimnuvx][-o option][+abCefhimnuvx][+o option][argument]
```

Derivation First released in Issue 2.

Issue 6 The Open Group Corrigendum U029/2 is applied, correcting the second SYNOPSIS.

The Open Group Corrigendum U027/3 is applied, correcting a typographical error.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The option letters derived from the *set* special built-in are also accepted with a leading plus sign ('+').
- Large file extensions are added:
 - Pathname expansion does not fail due to the size of a file.
 - Shell input and output redirections have an implementation-defined offset maximum that is established in the open file description.

In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to “directory referred to by the *HOME* environment variable”.

Descriptions for the *ENV* and *PWD* environment variables are included to align with the IEEE P1003.2b draft standard.

sleep

Suspend execution for an interval

```
sleep time
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

sort

Sort, merge, or sequence check text files

```
sort [-m] [-o output] [-bdfinru] [-t char] [-k keydef]... [file...]
```

```
sort -c [-bdfinru] [-t char] [-k keydef] [file]
```

Derivation First released in Issue 2.

Issue 6 IEEE PASC Interpretation 1003.2 #174 is applied, updating the DESCRIPTION of comparisons.

IEEE PASC Interpretation 1003.2 #168 is applied.

split

Split files into pieces

UP `split [-l line_count] [-a suffix_length] [file[name]]`

`split -b n[k|m] [-a suffix_length] [file[name]]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

The obsolescent SYNOPSIS is removed.

strings

Find printable strings in files

UP `strings [-a] [-t format] [-n number] [file...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The obsolescent SYNOPSIS is removed.

strip

Remove unnecessary information from executable files (**DEVELOPMENT**)

SD `strip file...`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the Software Development Utilities option. This utility may not be present on all systems supporting the Single UNIX Specification.

No functional changes in this issue.

stty

Set the options for a terminal

`stty [-a | -g]`

stty operands

Derivation First released in Issue 2.

Issue 6 The legacy items **iuclc(-iuclc)**, **xcase**, **olcuc(-olcuc)**, **lcase(-lcase)**, and **LCASE(-LCASE)**, are removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/37 is applied, applying IEEE PASC Interpretation 1003.2 #133, fixing an error in the OPERANDS section for the Combination Modes **nl(-nl)**.

tabs

Set terminal tabs

UP XSI `tabs [-n | -a | -a2 | -c | -c2 | -c3 | -f | -p | -s | -u][+m[n]] [-T type]`

`tabs [-T type] [+[n]] n1[,n2,...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

tail

Copy the last part of a file

`tail [-f] [-c number | -n number] [file]`

Derivation First released in Issue 2.

Issue 6 The obsolescent SYNOPSIS lines and associated text are removed.

talk

Talk to another user

UP `talk address [terminal]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

tee

Duplicate standard input

`tee [-ai] [file...]`

Derivation First released in Issue 2.

Issue 6 IEEE PASC Interpretation 1003.2 #168 is applied.

test

Evaluate expression

`test [expression]``[[expression]]`

Derivation First released in Issue 2.

Issue 6 The `-h` operand is added for symbolic links, and access permission requirements are clarified for the `-r`, `-w`, and `-x` operands to align with the IEEE P1003.2b draft standard.The `-L` and `-S` operands are added for symbolic links and sockets.**time**

Time a simple command

UP `time [-p] utility [argument...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

touch

Change file access and modification times

```
touch [-acm] [ -r ref_file | -t time] file...
```

Derivation First released in Issue 2.

Issue 6 The obsolescent *date_time* operand is removed.

The Open Group Corrigendum U027/1 is applied. This extends the range of valid time past the Epoch to at least the time 0 hours, 0 minutes, 0 seconds, January 1, 2038, Coordinated Universal Time. This is a new requirement on POSIX implementations.

The range for double leap seconds is changed from [00,61] to [00,60] to align with the ISO/IEC 9899:1999 standard.

tput

Change terminal characteristics

UP

```
tput [-T type] operand...
```

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

tr

Translate characters

```
tr [-c | -C] [-s] string1 string2
```

```
tr -s [-c | -C] string1
```

```
tr -d [-c | -C] string1
```

```
tr -ds [-c | -C] string1 string2
```

Derivation First released in Issue 2.

Issue 6 The **-C** operand is added, and the description of the **-c** operand is changed to align with the IEEE P1003.2b draft standard.

true

Return true value

```
true
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

tsort

Topological sort

xSI `tsort [file]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

tty

Return user's terminal name

`tty`

Derivation First released in Issue 2.

Issue 6 The obsolescent `-s` option is removed.**type**

Write a description of command type

xSI `type name...`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

ulimit

Set or report file size limit

xSI `ulimit [-f] [blocks]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

umask

Get or set the file mode creation mask

`umask [-S] [mask]`

Derivation First released in Issue 2.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The octal mode is supported.

unalias

Remove alias definitions

UP `unalias alias-name...`

`unalias -a`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

No functional changes in this issue.

uname

Return system name

`uname [-snrvma]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

uncompress

Expand compressed data

XSI `uncompress [-cfv] [file...]`

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

unexpand

Convert spaces to tabs

UP `unexpand [-a | -t tablist] [file...]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The definition of the *LC_CTYPE* environment variable is changed to align with the IEEE P1003.2b draft standard.

unget

Undo a previous get of an SCCS file (**DEVELOPMENT**)

XSI `unget [-ns][-r SID] file...`

Derivation First released in Issue 2.

Issue 6 No functional changes are made in this issue.

uniq

Report or filter out repeated lines in a file

`uniq [-c|-d|-u][-f fields][-s char][input_file [output_file]]`

Derivation First released in Issue 2.

Issue 6 The obsolescent SYNOPSIS and associated text are removed.

unlink

Call the `unlink()` function

XSI `unlink file`

Derivation First released in Issue 5.

Issue 6 No functional changes in this issue.

uucp

System-to-system copy

XSI `uucp [-cCdfjmr][-n user] source-file... destination-file`

Derivation First released in Issue 2.

Issue 6 The `LC_TIME` and `TZ` entries are removed from the ENVIRONMENT VARIABLES section.

The UN margin codes and associated shading are removed from the `-C`, `-f`, `-j`, `-n`, and `-r` options in response to The Open Group Base Resolution bwg2001-003.

uudecode

Decode a binary file

UP `uudecode [-o outfile][file]`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The `-o outfile` option is added, as specified in the IEEE P1003.2b draft standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/35 is applied, clarifying in the DESCRIPTION that the initial mode used if either of the `op` characters is '+'

or '-' is unspecified.

uuencode

Encode a binary file

UP `uuencode [-m] [file] decode_pathname`

Derivation First released in Issue 4.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The Base64 algorithm and the ability to output to `/dev/stdout` are added as specified in the IEEE P1003.2b draft standard.

uustat

uucp status inquiry and job control

XSI `uustat [-q| -k jobid| -r jobid]`

`uustat [-s system] [-u user]`

Derivation First released in Issue 2.

Issue 6 The `LC_TIME` and `TZ` entries are removed from the ENVIRONMENT VARIABLES section.

The UN margin code and associated shading are removed from the `-q` option in response to The Open Group Base Resolution bwg2001-003.

uux

Remote command execution

XSI `uux [-np] command-string`

`uux [-jnp] command-string`

Derivation First released in Issue 2.

Issue 6 The obsolescent SYNOPSIS is removed.

The UN margin code and associated shading are removed from the `-j` option in response to The Open Group Base Resolution bwg2001-003.

val

Validate SCCS files (**DEVELOPMENT**)

XSI `val -`

`val [-s] [-m name] [-r SID] [-y type] file...`

Derivation First released in Issue 2.

Issue 6 The Open Group Corrigendum U025/4 is applied, correcting a typographical error in the EXIT STATUS.

The normative text is reworded to emphasize the term “shall” for implementation requirements. No functional change is intended by this change.

vi

Screen-oriented (visual) display editor

UP `vi [-rR] [-l] [-c command] [-t tagstring] [-w size] [file ...]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The APPLICATION USAGE section is added.

The obsolescent SYNOPSIS is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **reindent** command description is added.

The *vi* utility has been extensively rewritten for alignment with the IEEE P1003.2b draft standard.

IEEE PASC Interpretations 1003.2 #57, #62, #63, #64, #78, and #188 are applied.

IEEE PASC Interpretation 1003.2 #207 is applied, clarifying the description of the **R** command in a manner similar to the descriptions of other text input mode commands such as **i**, **o**, and **O**.

wait

Await process completion

`wait [pid...]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

wc

Word, line, and byte or character count

`wc [-c|-m] [-lw] [file...]`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

what

Identify SCCS files (**DEVELOPMENT**)

XSI `what [-s] file...`

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

who

Display who is on the system

UP `who [-mTu]`

XSI `who [-mu]-s[-bHlprt] [file]`

`who [-mTu] [-abdHlprt] [file]`

`who -q [file]`

`who am i`

`who am I`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

The TZ entry is added to the ENVIRONMENT VARIABLES section.

write

Write to another user

UP `write user_name [terminal]`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the User Portability Utilities option. Support for the User Portability Utilities option is mandatory on implementations supporting the Single UNIX Specification.

xargs

Construct argument lists and invoke utility

XSI `xargs [-t] [-p]] [-E eofstr] [-I replstr] [-L number] [-n number [-x]] [-s size] [utility [argument...]]`

Derivation First released in Issue 2.

Issue 6 The obsolescent `-e`, `-i`, and `-l` options are removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-p** option is added.
- In the INPUT FILES section, the file **/dev/tty** is used to read responses required by the **-p** option.
- The STDERR section is updated to describe the **-p** option.

The description of the **-E** option is aligned with ISO/IEC 9945-2: 1993 (POSIX-2).

yacc

Yet another compiler compiler (**DEVELOPMENT**)

CD `yacc [-dltv] [-b file_prefix] [-p sym_prefix] grammar`

Derivation First released in Issue 2.

Issue 6 This utility is marked as part of the C-Language Development Utilities option. Minor changes have been added to align with the IEEE P1003.2b draft standard. IEEE PASC Interpretation 1003.2 #177 is applied, changing the comment on **RCURL** from the **}%** token to the **%}**.

zcat

Expand and concatenate data

XSI `zcat [file...]`

Derivation First released in Issue 4.

Issue 6 No functional changes in this issue.

Headers Migration

This chapter contains a section for each header defined in XBD, Issue 6. Each section contains the SYNOPSIS and identifies syntax and semantic changes made to the header in Issue 6 (if any), complete with examples where appropriate. Only changes that might affect an application programmer are identified.

aio.h

Asynchronous input and output (**REALTIME**)

AIO

```
#include <aio.h>
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6 The **<aio.h>** header is marked as part of the Asynchronous Input and Output option and need not be available in all implementations.

The description of the constants is expanded.

No functional changes in this issue except that the **restrict** keyword is added to the prototype for *lio_listio()*:

```
int lio_listio(int, struct aiocb *restrict const[restrict], int,
              struct sigevent *restrict);
```

arpa/inet.h

Definitions for internet operations

```
#include <arpa/inet.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification. It contains definitions to support internet operations.

This header defines the types **in_port_t** and **in_addr_t** and the **in_addr** structure.

The INET_ADDRSTRLEN and INET6_ADDRSTRLEN macros are defined.

The following are defined as macros or functions or both: *htonl()*, *htons()*, *ntohl()*, *ntohs()*.

This header includes declarations for the following functions: *inet_addr()*, *inet_ntoa()*, *inet_ntop()*, and *inet_pton()*.

A change over the XNS, Issue 5.2 specification is that the **restrict** keyword is added to the prototypes for *inet_ntop()* and *inet_pton()*:

```
const char *inet_ntop(int, const void *restrict,
                     char *restrict, socklen_t);
int inet_pton(int, const char *restrict, void *restrict);
```

assert.h

Verify program assertion

```
#include <assert.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The definition of the *assert()* macro is changed for alignment with the ISO/IEC 9899:1999 standard.

complex.h

Complex arithmetic

```
#include <complex.h>
```

Derivation First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

Issue 6 This is a new header included for alignment with the ISO/IEC 9899:1999 standard. Complex types have been added to the ISO/IEC 9899:1999 standard as part of the effort to make C suitable and attractive for general numerical programming. Complex arithmetic is used heavily in certain important application areas.

This header defines the macros `complex` and `_Complex_I`.

This header includes declarations for the following functions: `cabs()`, `cabsf()`, `cabsl()`, `cacos()`, `cacosf()`, `cacosh()`, `cacoshf()`, `cacoshl()`, `cacosl()`, `carg()`, `cargf()`, `cargl()`, `casin()`, `casinf()`, `casinh()`, `casinhf()`, `casinhl()`, `casinl()`, `catan()`, `catanf()`, `catanh()`, `catanhf()`, `catanhl()`, `catanl()`, `ccos()`, `ccosf()`, `ccosh()`, `ccoshf()`, `ccoshl()`, `ccosl()`, `cexp()`, `cexpf()`, `cexpl()`, `cimag()`, `cimagf()`, `cimagl()`, `clog()`, `clogf()`, `clogl()`, `conj()`, `conjf()`, `conjl()`, `cpow()`, `cpowf()`, `cpowl()`, `cproj()`, `cprojf()`, `cprojl()`, `creal()`, `crealf()`, `creall()`, `csin()`, `csinf()`, `csinh()`, `csinhf()`, `csinhl()`, `csinl()`, `csqrt()`, `csqrtf()`, `csqrtl()`, `ctan()`, `ctanf()`, `ctanh()`, `ctanhf()`, `ctanhl()`, `ctanl()`.

cpio.h

Cpio archive values

```
xsi #include <cpio.h>
```

Derivation First released in Issue 3. Derived from IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 No functional changes in this issue.

The SEE ALSO is updated to refer to `pax`, since the `cpio` utility is not included in XCU, Issue 6.

ctype.h

Character types

```
#include <ctype.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes in this issue.

Extensions beyond the ISO C standard are marked.

dirent.h

Format of directory entries

```
#include <dirent.h>
```

Derivation First released in Issue 2.

Issue 6 The Open Group Corrigendum U026/7 is applied, correcting the prototype for *readdir_r()*.

The **restrict** keyword is added to the prototype for *readdir_r()*:

```
int readdir_r(DIR *restrict, struct dirent *restrict,
              struct dirent **restrict);
```

dlfcn.h

Dynamic linking

```
xsi #include <dlfcn.h>
```

Derivation First released in Issue 5.

Issue 6 No functional changes in this issue except that the **restrict** keyword is added to the prototype for *dlsym()*:

```
void *dlsym(void *restrict, const char *restrict);
```

errno.h

System error numbers

```
#include <errno.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The majority of the error conditions previously marked as extensions are now mandatory, except for the STREAMS-related error conditions.

Values for *errno* are now required to be distinct positive values rather than non-zero values. This change is for alignment with the ISO/IEC 9899:1999 standard.

fcntl.h

File control options

```
#include <fcntl.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- O_DSYNC and O_RSYNC are marked as part of the Synchronized Input and Output option and need not be available in all implementations.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The definition of the **mode_t**, **off_t**, and **pid_t** types is mandated.

The F_GETOWN and F_SETOWN values are added for sockets.

The *posix_fadvise()*, *posix_fallocate()*, and *posix_madvise()* functions are added for alignment with IEEE Std 1003.1d-1999. Note however that IEEE PASC Interpretation 1003.1 #102 is applied, moving the prototype for *posix_madvise()* to **<sys/mman.h>**:

```
int posix_fadvise(int, off_t, off_t, int);
int posix_fallocate(int, off_t, off_t);
```

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/18 is applied, updating the prototypes for *posix_fadvise()* and *posix_fallocate()* to be large file-aware, using **off_t** instead of **size_t**.

fenv.h

Floating-point environment

```
#include <fenv.h>
```

Derivation First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

Issue 6 This is a new header included for alignment with the ISO/IEC 9899:1999 standard. This header provides definitions for the floating-point environment.

This header defines the data types **fenv_t** and **fexcept_t**.

This header defines the following macros and constants:

```
FE_DIVBYZERO
FE_INEXACT
FE_INVALID
FE_OVERFLOW
FE_UNDERFLOW
FE_ALL_EXCEPT
FE_DOWNWARD
FE_TONEAREST
FE_TOWARDZERO
FE_UPWARD
FE_DFL_ENV
```

This header includes declarations for the following functions: *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*, *fesetexceptflag()*, *fetestexcept()*, *fegetround()*, *fesetround()*, *fegetenv()*, *feholdexcept()*, *fesetenv()*, *feupdateenv()*.

The following change has been made for alignment with the ISO/IEC 9899:1999 standard, Defect Report 202:

- The return types for *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*, *fesetexceptflag()*, *fegetenv()*, *fesetenv()*, and *feupdateenv()* are changed from **void** to **int**.

float.h

Floating types

```
#include <float.h>
```

Derivation First released in Issue 4. Derived from the ISO C standard.

Issue 6 No functional changes in this issue, although the description of the operations with floating-point values has been reworked more closely to align with the description and layout of the ISO/IEC 9899: 1999 standard.

fmtmsg.h

Message display structures

```
xSI #include <fmtmsg.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes in this issue.

fnmatch.h

Filename-matching types

```
#include <fnmatch.h>
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2: 1993 (POSIX-2).

Issue 6 The constant FNM_NOSYS is marked obsolescent.

ftw.h

File tree traversal

```
xSI #include <ftw.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes in this issue.

glob.h

Pathname pattern-matching types

```
#include <glob.h>
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2: 1993 (POSIX-2).

Issue 6 The **restrict** keyword is added to the prototype for *glob()*:

```
int glob(const char *restrict, int,
        int (*)(const char *, int), glob_t *restrict);
```

The constant GLOB_NOSYS is marked obsolescent.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/8 is applied, correcting the *glob()* prototype definition by removing the **restrict** qualifier from the function pointer argument.

grp.h

Group structure

```
#include <grp.h>
```

Derivation First released in Issue 1.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The definition of **gid_t** is mandated.
- The *getgrgid_r()* and *getgrnam_r()* functions are marked as part of the Thread-Safe Functions option and need not be available in all implementations.

iconv.h

Codeset conversion facility

```
xsi #include <iconv.h>
```

Derivation First released in Issue 4.

Issue 6 The **restrict** keyword is added to the prototype for *iconv()*:

```
size_t iconv(iconv_t, char **restrict, size_t *restrict,  
             char **restrict, size_t *restrict);
```

inttypes.h

Fixed size integer types

```
#include <inttypes.h>
```

Derivation First released in Issue 5.

Issue 6 The purpose of this header is to provide a set of integer types whose definitions are consistent across machines and independent of operating systems and other implementation idiosyncrasies. It defines, via **typedef**, integer types of various sizes.

The Open Group Base Resolution bwg97-006 is applied.

This reference page is updated to align with the ISO/IEC 9899: 1999 standard.

iso646.h

Alternative spellings

```
#include <iso646.h>
```

Derivation First released in Issue 5. Derived from ISO C Amendment 1 (MSE).

Issue 6 No functional changes in this issue.

langinfo.h

Language information constants

```
#include <langinfo.h>
```

Derivation First released in Issue 2.

Issue 6 The constants YESSTR and NOSTR are removed. These were previously marked LEGACY in Issue 5.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/9 is applied, adding a sentence to the “Meaning” column entry for the CRNCYSTR constant. This change is to accommodate historic practice.

libgen.h

Definitions for pattern matching functions

```
xsi #include <libgen.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 The `__loc1` symbol and the `regcmp()` and `regex()` functions are removed. These were previously marked LEGACY in Issue 5.

limits.h

Implementation-defined constants

```
#include <limits.h>
```

Derivation First released in Issue 1.

Issue 6 The Open Group Corrigendum U033/4 is applied. The wording is made clear for `{CHAR_MIN}`, `{INT_MIN}`, `{LONG_MIN}`, `{SCHAR_MIN}`, and `{SHRT_MIN}` that these are maximum acceptable values.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The minimum value for `{CHILD_MAX}` is 25. This is a FIPS requirement.
- The minimum value for `{OPEN_MAX}` is 20. This is a FIPS requirement.
- The minimum value for `{NGROUPS_MAX}` is 8. This is also a FIPS requirement.

Symbolic constants are added for `{_POSIX_SYMLINK_MAX}`, `{_POSIX_SYMLOOP_MAX}`, `{_POSIX_RE_DUP_MAX}`, `{RE_DUP_MAX}`, `{SYMLOOP_MAX}`, and `{SYMLINK_MAX}`.

`{_POSIX_TZNAME_MAX}` is changed from 3 to 6.

The following values are added for alignment with IEEE Std 1003.1d-1999:


```
{_POSIX_SS_REPL_MAX}  
{SS_REPL_MAX}  
{POSIX_ALLOC_SIZE_MIN}  
{POSIX_REC_INCR_XFER_SIZE}  
{POSIX_REC_MAX_XFER_SIZE}  
{POSIX_REC_MIN_XFER_SIZE}  
{POSIX_REC_XFER_ALIGN}
```

Reference to `CLOCK_MONOTONIC` is added in the description of `{_POSIX_CLOCKRES_MIN}` for alignment with IEEE Std 1003.1j-2000.

The constants `{LLONG_MIN}`, `{LLONG_MAX}`, and `{ULLONG_MAX}` are added for alignment with the ISO/IEC 9899:1999 standard.

The following values are added for alignment with IEEE Std 1003.1q-2000:

```
{_POSIX_TRACE_EVENT_NAME_MAX}  
{_POSIX_TRACE_NAME_MAX}  
{_POSIX_TRACE_SYS_MAX}  
{_POSIX_TRACE_USER_EVENT_MAX}  
{TRACE_EVENT_NAME_MAX}  
{TRACE_NAME_MAX}  
{TRACE_SYS_MAX}  
{TRACE_USER_EVENT_MAX}
```

The new limits `{_XOPEN_NAME_MAX}` and `{_XOPEN_PATH_MAX}` are added as minimum values for `{PATH_MAX}` and `{NAME_MAX}` limits on XSI-conformant systems.

The legacy symbols `{PASS_MAX}` and `{TMP_MAX}` are removed.

The values for the limits `{CHAR_BIT}`, `{CHAR_MAX}`, `{SCHAR_MAX}`, and `{UCHAR_MAX}` are now required to be 8, +127, 255, and -128, respectively.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/10 is applied, correcting the value of `{_POSIX_CHILD_MAX}` from 6 to 25. This is for FIPS 151-2 alignment.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/19 is applied, updating the values for `{INT_MAX}`, `{UINT_MAX}`, and `{INT_MIN}` to be CX extensions over the ISO C standard, and correcting `{WORD_BIT}` from 16 to 32.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/20 is applied, removing `{CHARCLASS_NAME_MAX}` from the “Other Invariant Values” section (it also occurs under “Runtime Inceasable Values”).

locale.h

Category macros

```
#include <locale.h>
```

Derivation First released in Issue 3.

Issue 6 The **lconv** structure is expanded with the following new members for alignment with the ISO/IEC 9899:1999 standard: **int_n_cs_precedes**, **int_n_sep_by_space**, **int_n_sign_posn**, **int_p_cs_precedes**, **int_p_sep_by_space**, and **int_p_sign_posn**.

Extensions beyond the ISO C standard are marked.

math.h

Mathematical declarations

```
#include <math.h>
```

Derivation First released in Issue 1.

Issue 6 This reference page is updated to align with the ISO/IEC 9899: 1999 standard.

The following types are added: **float_t** and **double_t**.

The following constants and macros are added:

```
HUGE_VALF
HUGE_VALD
INFINITY
NAN
FP_INFINITE
FP_NAN
FP_NORMAL
FP_SUBNORMAL
FP_ZERO
FP_FAST_FMA
FP_FAST_FMAF
FP_FAST_FMAL
FP_ILOGBO
FP_ILOGBNAN
MATH_ERRNO
MATH_ERREXCEPT
math_errhandling
```

The following macros are added:

```
fpclassify(), isfinite(), isinf(), isnan(), isnormal(), signbit(), isgreater(),
isgreaterequal(), isless(), islessequal(), islessgreater(), isunordered()
```

The following function declarations are added:

```
acosf(), acoshf(), acoshl(), acosl(), asinf(), asinhf(), asinhl(), asinl(),
atan2f(), atan2l(), atanf(), atanhf(), atanhl(), atanl(), cbrt(), cbrtf(), cbrtl(),
ceilf(), ceill(), copysign(), copysignf(), copysignl(), cosf(), coshf(), coshl(),
cosl(), erfcf(), erfc_l(), erff(), erfl(), exp2(), exp2f(), exp2l(), expf(), expl(),
expm1f(), expm1l(), fabsf(), fabs_l(), fdim(), fdimf(), fdiml(), floorf(), floorl(),
fmaf(), fmaf(), fmal(), fmax(), fmaxf(), fmaxl(), fmin(), fminf(), fminl(),
fmodf(), fmodl(), frexpf(), frexpl(), hypotf(), hypotl(), ilogbf(), ilogbl(), lrint(),
lrintf(), lrintl(), lround(), lroundf(), lroundl(), ldexpf(), ldexpl(), lgammaf(),
lgammal(), log10f(), log10l(), log1pf(), log1pl(), log2(), log2f(), log2l(),
logbf(), logbl(), logf(), logl(), llrint(), llrintf(), llrintl(), llround(), llroundf(),
llroundl(), modff(), modfl(), nan(), nanf(), nanl(), nearbyint(), nearbyintf(),
nearbyintl(), nextafterf(), nextafterl(), nexttoward(), nexttowardf(),
nexttowardl(), powf(), powl(), remainderf(), remainderl(), remquo(),
remquof(), remquol(), rintf(), rintl(), round(), roundf(), roundl(), scalbln(),
scalblnf(), scalblnl(), scalbn(), scalbnf(), scalbnl(), sinf(), sinhf(), sinhl(),
sinl(), sqrtf(), sqrtl(), tanf(), tanhf(), tanhl(), tanl(), tgamma(), tgammaf(),
tgamma_l(), trunc(), truncf(), trunc_l()
```

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/21 is applied, making it clear that the meaning of the `FP_FAST_FMA` macro is unspecified if the macro is undefined.

monetary.h

Monetary types

XSI `#include <monetary.h>`

Derivation First released in Issue 4.

Issue 6 The **restrict** keyword is added to the prototype for `strfmon()`:

```
ssize_t strfmon(char *restrict, size_t,
                const char *restrict, ...);
```

mqueue.h

Message queues (**REALTIME**)

MSG `#include <mqueue.h>`

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6 The `<mqueue.h>` header is marked as part of the Message Passing option and need not be available in all implementations.

The `mq_timedreceive()` and `mq_timedsend()` functions are added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the prototypes for `mq_setattr()` and `mq_timedreceive()`:

```
int      mq_setattr(mqd_t, const struct mq_attr *restrict,
                   struct mq_attr *restrict);
ssize_t  mq_timedreceive(mqd_t, char *restrict, size_t,
                       unsigned *restrict, const struct timespec *restrict);
```

ndbm.h

Definitions for ndbm database operations

XSI `#include <ndbm.h>`

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes in this issue.

net/if.h

Sockets local interfaces

```
#include <net/if.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification. It contains information for sockets local interfaces. There have been no changes over the XNS, Issue 5.2 specification.

This header declares the **if_nameindex** structure and the `IF_NAMESIZE` macro for manipulating interface names.

The following functions are declared: `if_nametoindex()`, `if_indextoname()`, `if_nameindex()`, and `if_freenameindex()`.

netdb.h

Definitions for network database operations

```
#include <netdb.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification.

The Open Group Base Resolution bwg2001-009 is applied, which changes the return type for `gai_strerror()` from **char *** to **const char ***. This is for coordination with the IPnG Working Group.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/11 is applied, adding a description of the `NI_NUMERICSCOPE` macro and correcting the `getnameinfo()` function prototype. These changes are for alignment with IPv6.

netinet/in.h

Internet address family

```
#include <netinet/in.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification.

The `sin_zero` member was removed from the **sockaddr_in** structure as per The Open Group Base Resolution bwg2001-004.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/12 is applied, adding **const** qualifiers to the `in6addr_any` and `in6addr_loopback` external variables.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/22 is applied, making it clear which structure members are in network byte order.

netinet/tcp.h

Definitions for the Internet Transmission Control Protocol (TCP)

```
#include <netinet/tcp.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification.

nl_types.h

Data types

```
xSI #include <nl_types.h>
```

Derivation First released in Issue 2.

Issue 6 No functional changes in this issue.

poll.h

Definitions for the *poll()* function

```
xSI #include <poll.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 The description of the symbolic constants is updated to match the *poll()* function.
Text related to STREAMS has been moved to the *poll()* reference page.

A note is added to the DESCRIPTION regarding the significance and semantics of normal, priority, and high-priority data.

pthread.h

Threads

```
THR #include <pthread.h>
```

Derivation First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Issue 6 The RTT margin markers are now broken out into their POSIX options.

The Open Group Corrigendum U021/9 is applied, correcting the prototype for the *pthread_cond_wait()* function.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the *pthread_setschedparam()* function so that its second argument is of type **int**.

The *pthread_getcpuclockid()* and *pthread_mutex_timedlock()* functions are added for alignment with IEEE Std 1003.1d-1999.

The following functions are added for alignment with IEEE Std 1003.1j-2000:

pthread_barrier_destroy(), *pthread_barrier_init()*, *pthread_barrier_wait()*,
pthread_barrierattr_destroy(), *pthread_barrierattr_getpshared()*,
pthread_barrierattr_init(), *pthread_barrierattr_setpshared()*,
pthread_condattr_getclock(), *pthread_condattr_setclock()*,
pthread_rwlock_timedrdlock(), *pthread_rwlock_timedwrlock()*,
pthread_spin_destroy(), *pthread_spin_init()*, *pthread_spin_lock()*,
pthread_spin_trylock(), *pthread_spin_unlock()*

PTHREAD_RWLOCK_INITIALIZER is deleted for alignment with IEEE Std 1003.1j-2000.

Functions previously marked as part of the Read-Write Locks option are now moved to the Threads option and need not be available in all implementations.

The **restrict** keyword is added to the following prototypes:

pthread_attr_getguardsize(), *pthread_attr_getinheritsched()*,
pthread_attr_getschedparam(), *pthread_attr_getschedpolicy()*,
pthread_attr_getscope(), *pthread_attr_getstackaddr()*,
pthread_attr_getstacksize(), *pthread_attr_setschedparam()*,
pthread_barrier_init(), *pthread_barrierattr_getpshared()*, *pthread_cond_init()*,
pthread_cond_signal(), *pthread_cond_timedwait()*, *pthread_cond_wait()*,
pthread_condattr_getclock(), *pthread_condattr_getpshared()*,
pthread_create(), *pthread_getschedparam()*, *pthread_mutex_getprioceiling()*,
pthread_mutex_init(), *pthread_mutex_setprioceiling()*,
pthread_mutexattr_getprioceiling(), *pthread_mutexattr_getprotocol()*,
pthread_mutexattr_getpshared(), *pthread_mutexattr_gettype()*,
pthread_rwlock_init(), *pthread_rwlock_timedrdlock()*,
pthread_rwlock_timedwrlock(), *pthread_rwlockattr_getpshared()*,
pthread_sigmask()

as follows:

```
int pthread_attr_getguardsize(const pthread_attr_t *restrict,
    size_t *restrict);
int pthread_attr_getinheritsched(const pthread_attr_t *restrict,
    int *restrict);
int pthread_attr_getschedparam(const pthread_attr_t *restrict,
    struct sched_param *restrict);
int pthread_attr_getschedpolicy(const pthread_attr_t *restrict,
    int *restrict);
int pthread_attr_getscope(const pthread_attr_t *restrict,
    int *restrict);
int pthread_attr_getstack(const pthread_attr_t *restrict,
    void **restrict, size_t *restrict);
int pthread_attr_getstackaddr(const pthread_attr_t *restrict,
    void **restrict);
int pthread_attr_getstacksize(const pthread_attr_t *restrict,
    size_t *restrict);
int pthread_attr_setschedparam(pthread_attr_t *restrict,
    const struct sched_param *restrict);
int pthread_barrier_init(pthread_barrier_t *restrict,
    const pthread_barrierattr_t *restrict, unsigned);
int pthread_barrierattr_getpshared( \
    const pthread_barrierattr_t *restrict, int *restrict);
```

```
int pthread_cond_init(pthread_cond_t *restrict,  
    const pthread_condattr_t *restrict);  
int pthread_cond_timedwait(pthread_cond_t *restrict,  
    pthread_mutex_t *restrict,  
    const struct timespec *restrict);  
int pthread_cond_wait(pthread_cond_t *restrict,  
    pthread_mutex_t *restrict);  
int pthread_condattr_getclock(  
    const pthread_condattr_t *restrict, clockid_t *restrict);  
int pthread_condattr_getpshared(  
    const pthread_condattr_t *restrict, int *restrict);  
int pthread_create(pthread_t *restrict,  
    const pthread_attr_t *restrict, void (*)(void *),  
    void *restrict);  
int pthread_getschedparam(pthread_t, int *restrict,  
    struct sched_param *restrict);  
int pthread_mutex_getprioceiling(  
    const pthread_mutex_t *restrict, int *restrict);  
int pthread_mutex_init(pthread_mutex_t *restrict,  
    const pthread_mutexattr_t *restrict);  
int pthread_mutex_setprioceiling(pthread_mutex_t *restrict,  
    int, int *restrict);  
int pthread_mutexattr_getprioceiling(  
    const pthread_mutexattr_t *restrict, int *restrict);  
int pthread_mutexattr_getprotocol(  
    const pthread_mutexattr_t *restrict, int *restrict);  
int pthread_mutexattr_getpshared(  
    const pthread_mutexattr_t *restrict, int *restrict);  
int pthread_mutexattr_gettype(  
    const pthread_mutexattr_t *restrict, int *restrict);  
int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *, int);  
int pthread_mutexattr_setprotocol(pthread_mutexattr_t *, int);  
int pthread_rwlock_init(pthread_rwlock_t *restrict,  
    const pthread_rwlockattr_t *restrict);  
int pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict,  
    const struct timespec *restrict);  
int pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict,  
    const struct timespec *restrict);  
int pthread_rwlockattr_getpshared(  
    const pthread_rwlockattr_t *restrict, int *restrict);
```

IEEE PASC Interpretation 1003.1 #86 is applied, allowing the symbols from **<sched.h>** and **<time.h>** to be made visible when **<pthread.h>** is included. Previously this was an XSI extension.

IEEE PASC Interpretation 1003.1c #42 is applied, removing the requirement for prototypes for the *pthread_kill()* and *pthread_sigmask()* functions. These are required to be in the **<signal.h>** header. They are allowed here through the name space rules.

IEEE PASC Interpretation 1003.1 #96 is applied, adding the *pthread_setschedprio()* function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/13 is applied, correcting shading errors that were in contradiction with the System Interfaces volume of IEEE Std 1003.1-2001.

pwd.h

Password structure

```
#include <pwd.h>
```

Derivation First released in Issue 1.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **gid_t** and **uid_t** types are mandated.
- The *getpwnam_r()* and *getpwuid_r()* functions are marked as part of the `_POSIX_THREAD_SAFE_FUNCTIONS` option and need not be available in all implementations.

regex.h

Regular expression matching types

```
#include <regex.h>
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 The `REG_ENOSYS` constant is marked obsolescent.

The **restrict** keyword is added to the prototypes for *regcomp()*, *regerror()*, and *regexexec()*:

```
int    regcomp(regex_t *restrict, const char *restrict, int);
size_t regerror(int, const regex_t *restrict, char *restrict,
               size_t);
int    regexexec(const regex_t *restrict, const char *restrict,
                size_t, regmatch_t[restrict], int);
```

A statement is added that the **size_t** type is defined as described in `<sys/types.h>`.

sched.h

Execution scheduling (**REALTIME**)

PS

```
#include <sched.h>
```

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6 The `<sched.h>` header is marked as part of the Process Scheduling option and need not be available in all implementations.

Sporadic server members are added to the **sched_param** structure, and the `SCHED_SPORADIC` scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #108 is applied, correcting the **sched_param** structure whose members *sched_ss_repl_period* and *sched_ss_init_budget*

members should be type **struct timespec** and not **timespec**.

IEEE Std 1003.1-2001/Cor 1-2002, items XSH/TC1/D6/52 and XSH/TC1/D6/53 are applied, aligning the function prototype shading and margin codes with the System Interfaces volume of IEEE Std 1003.1-2001.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/23 is applied, updating the DESCRIPTION to differentiate between thread and process execution.

search.h

Search tables

XSI `#include <search.h>`

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The Open Group Corrigendum U021/6 is applied, updating the prototypes for *tdelete()* and *tsearch()*.

The **restrict** keyword is added to the prototype for *tdelete()*.

semaphore.h

Semaphores (**REALTIME**)

SEM `#include <semaphore.h>`

Derivation First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

Issue 6 The **<semaphore.h>** header is marked as part of the Semaphores option and need not be available in all implementations.

The Open Group Corrigendum U021/3 is applied, adding a description of SEM_FAILED.

The *sem_timedwait()* function is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the prototypes for *sem_getvalue()* and *sem_timedwait()*.

setjmp.h

Stack environment declarations

`#include <setjmp.h>`

Derivation First released in Issue 1.

Issue 6 No functional changes in this issue.

Extensions beyond the ISO C standard are marked.

signal.h

Signals

```
#include <signal.h>
```

Derivation First released in Issue 1.

Issue 6 The Open Group Corrigendum U035/2 is applied. In the DESCRIPTION, the wording for abnormal termination is clarified.

The Open Group Corrigendum U028/8 is applied, correcting the prototype for the *sigset()* function.

The Open Group Corrigendum U026/3 is applied, correcting the type of the *sigev_notify_function* function member of the **sigevent** structure.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The SIGCHLD, SIGCONT, SIGSTOP, SIGTSTP, SIGTTIN, and SIGTTOU signals are now mandated. This is also a FIPS requirement.
- The **pid_t** definition is mandated.

The RT markings are changed to RTS to denote that the semantics are part of the Realtime Signals Extension option and need not be available in all implementations.

The **restrict** keyword is added to the prototypes for *sigaction()*, *sigaltstack()*, *sigprocmask()*, *sigtimedwait()*, *sigwait()*, and *sigwaitinfo()*.

IEEE PASC Interpretation 1003.1 #85 is applied, adding the statement that symbols from **<time.h>** may be made visible when **<signal.h>** is included.

Extensions beyond the ISO C standard are marked.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/14 is applied, changing the descriptive text for members of **struct sigaction**.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/15 is applied, correcting the definition of the *sa_sigaction* member of **struct sigaction**.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/24 is applied, reworking the ordering of the **siginfo_t** type structure in the DESCRIPTION. This is an editorial change and no normative change is intended.

spawn.h

Spawn (ADVANCED REALTIME)

```
SPN #include <spawn.h>
```

Derivation First released in Issue 6. Included for alignment with IEEE Std 1003.1d-1999.

Issue 6 This is a new header included for alignment with IEEE Std 1003.1d-1999.

The following change is made over IEEE Std 1003.1d-1999:

- The **restrict** keyword is added to the following prototypes:

```
posix_spawn(), posix_spawn_file_actions_addopen(),  
posix_spawnattr_getsigdefault(), posix_spawnattr_getflags(),  
posix_spawnattr_getpgroup(), posix_spawnattr_getschedparam(),  
posix_spawnattr_getschedpolicy(), posix_spawnattr_getsigmask(),  
posix_spawnattr_setsigdefault(), posix_spawnattr_setschedparam(),  
posix_spawnattr_setsigmask(), posix_spawnnp()
```

as follows:

```
int posix_spawn(pid_t *restrict, const char *restrict,  
               const posix_spawn_file_actions_t *,  
               const posix_spawnattr_t *restrict, char *const [restrict],  
               char *const [restrict]);  
int posix_spawn_file_actions_addopen(  
    posix_spawn_file_actions_t *restrict, int,  
    const char *restrict, int, mode_t);  
int posix_spawnattr_getsigdefault(  
    const posix_spawnattr_t *restrict, sigset_t *restrict);  
int posix_spawnattr_getflags(  
    const posix_spawnattr_t *restrict, short *restrict);  
int posix_spawnattr_getpgroup(  
    const posix_spawnattr_t *restrict, pid_t *restrict);  
int posix_spawnattr_getschedparam(  
    const posix_spawnattr_t *restrict,  
    struct sched_param *restrict);  
int posix_spawnattr_getschedpolicy(  
    const posix_spawnattr_t *restrict, int *restrict);  
int posix_spawnattr_getsigmask(  
    const posix_spawnattr_t *restrict, sigset_t *restrict);  
int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict,  
    const sigset_t *restrict);  
int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict,  
    const struct sched_param *restrict);  
int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict,  
    const sigset_t *restrict);  
int posix_spawnnp(pid_t *restrict, const char *restrict,  
                  const posix_spawn_file_actions_t *,  
                  const posix_spawnattr_t *restrict,  
                  char *const [restrict], char *const [restrict]);
```

stdarg.h

Handle variable argument list

```
#include <stdarg.h>  
  
void va_start(va_list ap, argN);  
void va_copy(va_list dest, va_list src);  
type va_arg(va_list ap, type);  
void va_end(va_list ap);
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 The new function `va_copy()` is added to copy a variable argument list. This is added for alignment with the ISO/IEC 9899:1999 standard.

stdbool.h

Boolean type and values

```
#include <stdbool.h>
```

Derivation First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

Issue 6 This is a new header included for alignment with the ISO/IEC 9899:1999 standard. This header contains a **typedef** for `bool`, and macros for `true`, `false`, and `__bool_true_false_are_defined`.

stddef.h

Standard type definitions

```
#include <stddef.h>
```

Derivation First released in Issue 4. Derived from ANSI standard X3.159-1989, Programming Language C (ANSI C).

Issue 6 No functional changes in this issue.

stdint.h

Integer types

```
#include <stdint.h>
```

Derivation First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

Issue 6 This is a new header included for alignment with the ISO/IEC 9899:1999 standard. This header is a subset of **<inttypes.h>**, the rationale being that this subset is more suitable for use in environments that might not support the formatted I/O functions, thereby permitting those environments to avoid defining a large number of macros.

ISO/IEC 9899:1999 standard, Technical Corrigendum No. 1 is incorporated.

stdio.h

Standard buffered input/output

```
#include <stdio.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The constant `L_cuserid` and the external variables `optarg`, `opterr`, `optind`, and `optopt` are removed as they were previously marked LEGACY.

The `cuserid()`, `getopt()`, and `getw()` functions are removed as they were previously marked LEGACY.

Several functions are marked as part of the Thread-Safe Functions option and need not be available in all implementations.

Headers Migration

This reference page is updated to align with the ISO/IEC 9899:1999 standard. Note that the description of the **fpos_t** type is now explicitly updated to exclude array types.

Extensions beyond the ISO C standard are marked.

stdlib.h

Standard library definitions

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

Derivation First released in Issue 3.

Issue 6 The Open Group Corrigendum U021/1 is applied, correcting the prototype for *realpath()* to be consistent with the reference page. The **restrict** keyword is also added to the prototype for *realpath()*:

```
char *realpath(const char *restrict, char *restrict);
```

The Open Group Corrigendum U028/13 is applied, correcting the prototype for *putenv()* to be consistent with the reference page:

```
int putenv(char *);
```

The *rand_r()* function is marked as part of the Thread-Safe Functions option and need not be available in all implementations.

Function prototypes for *setenv()* and *unsetenv()* are added:

```
int setenv(const char *, const char *, int);
int unsetenv(const char *);
```

The *posix_memalign()* function is added for alignment with IEEE Std 1003.1d-1999:

```
int posix_memalign(void **, size_t, size_t);
```

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

The *ecvt()*, *fcvt()*, *gcvt()*, and *mktemp()* functions are marked LEGACY. The individual reference pages should be consulted for information on alternative recommended interfaces.

The *ttyslot()* and *valloc()* functions are removed as they were previously marked LEGACY.

Extensions beyond the ISO C standard are marked.

string.h

String operations

```
#include <string.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The *strtok_r()* function is marked as part of the Thread-Safe Functions option and need not be available in all implementations.

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

strings.h

String operations

XSI `#include <strings.h>`

Derivation First released in Issue 4, Version 2.

Issue 6 The Open Group Corrigendum U021/2 is applied, correcting the prototype for *index()* to be consistent with the reference page.

The *bcmp()*, *bcopy()*, *bzero()*, *index()*, and *rindex()* functions are marked LEGACY. The individual reference pages should be consulted for information on alternative recommended interfaces.

stropts.h

STREAMS interface (**STREAMS**)

XSR `#include <stropts.h>`

Derivation First released in Issue 4, Version 2.

Issue 6 This header is marked as part of the XSI STREAMS Option Group and may not be available on all implementations.

The **restrict** keyword is added to the prototypes for *getmsg()* and *getpmsg()*:

```
int getmsg(int, struct strbuf *restrict,
           struct strbuf *restrict, int *restrict);
int getpmsg(int, struct strbuf *restrict,
            struct strbuf *restrict, int *restrict, int *restrict);
```

sys/ipc.h

XSI interprocess communication access structure

XSI `#include <sys/ipc.h>`

Derivation First released in Issue 2. Derived from System V Release 2.0.

Issue 6 No functional changes in this issue.

sys/mman.h

Memory management declarations

`#include <sys/mman.h>`

Derivation First released in Issue 4, Version 2.

Issue 6 The **<sys/mman.h>** header is marked as dependent on support for either the Memory Mapped Files, Process Memory Locking, or Shared Memory Objects options.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The TYM margin code is added to the list of margin codes for the **<sys/mman.h>** header line, as well as for other lines.

- The following flags are added:

```
POSIX_TYPED_MEM_ALLOCATE
POSIX_TYPED_MEM_ALLOCATE_CONTIG
POSIX_TYPED_MEM_MAP_ALLOCATABLE
```

- The **posix_tmi_length** structure is added.
- The `posix_mem_offset()`, `posix_typed_mem_get_info()`, and `posix_typed_mem_open()` functions are added, with the **restrict** keyword added to the prototype for `posix_mem_offset()`:

```
int posix_mem_offset(const void *restrict, size_t,
                    off_t *restrict, size_t *restrict, int *restrict);
int posix_typed_mem_get_info(int,
                             struct posix_typed_mem_info *);
int posix_typed_mem_open(const char *, int, int);
```

IEEE PASC Interpretation 1003.1 #102 is applied, adding the prototype for `posix_madvise()`:

```
int posix_madvise(void *, size_t, int);
```

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/34 is applied, changing the margin code for the `mmap()` function from MFISHM to MC3 (notation for MFISHMITYM).

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/36 is applied, changing the margin code for the `munmap()` function from MFISHM to MC3 (notation for MFISHMITYM).

sys/msg.h

XSI message queue structures

```
xsi #include <sys/msg.h>
```

Derivation First released in Issue 2. Derived from System V Release 2.0.

Issue 6 No functional changes in this issue.

sys/resource.h

Definitions for XSI resource operations

```
xsi #include <sys/resource.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes in this issue.

sys/select.h

Select types

```
#include <sys/select.h>
```

Derivation First released in Issue 6. Derived from IEEE Std 1003.1g-2000.

Issue 6 This is a new header included for alignment with IEEE Std 1003.1g-2000. It includes many of the definitions that were previously found in the **<sys/time.h>** header. That header is now allowed to include the **<sys/select.h>** header.

A change from the Single UNIX Specification is that the requirement for the **fd_set** structure to have a member *fds_bits* has been removed as per The Open Group Base Resolution bwg2001-005.

The *pselect()* function is new. The **restrict** keyword has been added to the prototypes for *select()* and *pselect()*:

```
int pselect(int, fd_set *restrict, fd_set *restrict,
           fd_set *restrict, const struct timespec *restrict,
           const sigset_t *restrict);
int select(int, fd_set *restrict, fd_set *restrict,
           fd_set *restrict, struct timeval *restrict);
```

sys/sem.h

XSI semaphore facility

```
xsi #include <sys/sem.h>
```

Derivation First released in Issue 2. Derived from System V Release 2.0.

Issue 6 No functional changes in this issue.

sys/shm.h

XSI shared memory facility

```
xsi #include <sys/shm.h>
```

Derivation First released in Issue 2. Derived from System V Release 2.0.

Issue 6 No functional changes in this issue.

sys/socket.h

Main sockets header

```
#include <sys/socket.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification.

Changes over the XNS, Issue 5.2 specification include that the **restrict** keyword is added to the prototypes for *accept()*, *getpeername()*, *getsockname()*, *getsockopt()*, and *recvfrom()*:


```
int    accept(int, struct sockaddr *restrict,
          socklen_t *restrict);
int    getpeername(int, struct sockaddr *restrict,
          socklen_t *restrict);
int    getsockname(int, struct sockaddr *restrict,
          socklen_t *restrict);
int    getsockopt(int, int, int, void *restrict,
          socklen_t *restrict);
ssize_t recvfrom(int, void *restrict, size_t, int,
          struct sockaddr *restrict, socklen_t *restrict);
```

sys/stat.h

Data returned by the *stat()* function

```
#include <sys/stat.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The `S_TYPEISMQ()`, `S_TYPEISSEM()`, and `S_TYPEISSHM()` macros are unconditionally mandated.

The Open Group Corrigendum U035/4 is applied. In the DESCRIPTION, the types **blksize_t** and **blkcnt_t** have been described.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **dev_t**, **ino_t**, **mode_t**, **nlink_t**, **uid_t**, **gid_t**, **off_t**, and **time_t** types are mandated.
- The *lstat()* function is now mandatory.

`S_IFSOCK` and `S_ISSOCK` are added for sockets.

The description of **stat** structure members is changed to reflect contents when file type is a symbolic link.

The test macro `S_TYPEISTMO` is added for alignment with IEEE Std 1003.1j-2000.

The **restrict** keyword is added to the prototypes for *lstat()* and *stat()*:

```
int lstat(const char *restrict, struct stat *restrict);
int stat(const char *restrict, struct stat *restrict);
```

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/25 is applied, adding to the DESCRIPTION that the **timespec** structure may be defined as described in the **<time.h>** header.

sys/statvfs.h

VFS File System information structure

```
xSI #include <sys/statvfs.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 The Open Group Corrigendum U035/5 is applied. In the DESCRIPTION, the types **fsblkcnt_t** and **fsfilcnt_t** have been described.

The **restrict** keyword is added to the prototype for *statvfs()*:

```
int statvfs(const char *restrict, struct statvfs *restrict);
```

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/18 is applied, changing the description of ST_NOSUID from “Does not support *setuid()*/*setgid()* semantics” to “Does not support the semantics of the ST_ISUID and ST_ISGID file mode bits”.

sys/time.h

Time types

```
xSI #include <sys/time.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 The **restrict** keyword is added to the prototypes for *gettimeofday()*, *select()*, and *setitimer()*.

The note is added that inclusion of this header may also make symbols visible from **<sys/socket.h>**.

The *utimes()* function is marked LEGACY. The individual reference page should be consulted for information on alternative recommended interfaces.

sys/timeb.h

Additional definitions for date and time

```
xSI #include <sys/timeb.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 The *ftime()* function is marked LEGACY. The individual reference page should be consulted for information on alternative recommended interfaces.

sys/times.h

File access and modification times structure

```
#include <sys/times.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes in this issue.

sys/types.h

Data types

```
#include <sys/types.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The **pthread_barrier_t**, **pthread_barrierattr_t**, and **pthread_spinlock_t** types are added for alignment with IEEE Std 1003.1j-2000.

The margin code is changed from XSI to THR for the **pthread_rwlock_t** and **pthread_rwlockattr_t** types as Read-Write Locks have been absorbed into the POSIX Threads option. The threads types are marked THR.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/26 is applied, adding **pthread_t** to the list of types that are not required to be arithmetic types, thus allowing **pthread_t** to be defined as a structure.

sys/uio.h

Definitions for vector I/O operations

```
xSI #include <sys/uio.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 Text referring to scatter/gather I/O is added to the DESCRIPTION.

sys/un.h

Definitions for UNIX domain sockets

```
#include <sys/un.h>
```

Derivation First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

Issue 6 This is a new header included for alignment with the XNS, Issue 5.2 specification.

sys/utsname.h

System name structure

```
#include <sys/utsname.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 No functional changes in this issue.

sys/wait.h

Declarations for waiting

```
#include <sys/wait.h>
```

Derivation First released in Issue 3. Entry included for alignment with IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 The *wait3()* function is removed.

syslog

Definitions for system error logging

```
xSI #include <syslog.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes in this issue.

tar.h

Extended tar definitions

```
#include <tar.h>
```

Derivation First released in Issue 3. Derived from IEEE Std 1003.1-1988 (POSIX.1).

Issue 6 No functional changes in this issue.

The SEE ALSO section now refers to *pax* since XCU, Issue 6 no longer contains the *tar* utility.

termios.h

Define values for termios

```
#include <termios.h>
```

Derivation First released in Issue 3. Entry included for alignment with ISO/IEC 9945-1:1996 (POSIX-1).

Issue 6 The LEGACY symbols IUCLC, ULCUC, and XCASE are removed.

FIPS 151-2 requirements for the symbols CS7, CS8, CSTOPB, PARODD, and PARENB are reaffirmed.

tgmath.h

Type-generic macros

```
#include <tgmath.h>
```

Derivation First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

Issue 6 This is a new header included for alignment with the ISO/IEC 9899:1999 standard.

This header includes the **<math.h>** and **<complex.h>** and defines several type-generic macros.

Type-generic macros allow calling a function whose type is determined by the argument type, as is the case for C operators such as '+' and '*'. For example, with a type-generic *cos()* macro, the expression *cos((float)x)* will have type **float**. This feature enables writing more portably efficient code and alleviates need for awkward casting and suffixing in the process of porting or adjusting precision. Generic math functions are a widely appreciated feature of Fortran.

Generic macros are designed for a useful level of consistency with C++ overloaded math functions.

The great majority of existing C programs are expected to be unaffected when the **<tgmath.h>** header is included instead of the **<math.h>** or **<complex.h>** headers.

time.h

Time types

```
#include <time.h>
```

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 The Open Group Corrigendum U035/6 is applied. In the DESCRIPTION, the types **clockid_t** and **timer_t** have been described.

The following changes are made for alignment with ISO/IEC 9945-1:1996 (POSIX-1):

- The POSIX timer-related functions are marked as part of the Timers option and need not be available in all implementations.

The symbolic name CLK_TCK is removed. Application usage is added describing how its equivalent functionality can be obtained using *sysconf()*.

The *clock_getcpuclockid()* function and manifest constants CLOCK_PROCESS_CPUTIME_ID and CLOCK_THREAD_CPUTIME_ID are added for alignment with IEEE Std 1003.1d-1999.

The manifest constant CLOCK_MONOTONIC and the *clock_nanosleep()* function are added for alignment with IEEE Std 1003.1j-2000.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The range for seconds is changed from [0,61] to [0,60].
- The **restrict** keyword is added to the prototypes for *asctime_r()*, *gmtime_r()*, *localtime_r()*, *strptime()*, *strptime()*, *timer_create()*, and *timer_settime()*.

IEEE PASC Interpretation 1003.1 #84 is applied, adding the statement that symbols from the **<signal.h>** header may be made visible when the **<time.h>** header is included.

Extensions beyond the ISO C standard are marked.

trace.h

Tracing

```
TRC #include <trace.h>
```

Derivation First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

Issue 6 This is a new header included for alignment with IEEE Std 1003.1q-2000.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/40 is applied, adding the TRL margin code to the *posix_trace_flush()* function, for alignment with the System Interfaces volume of IEEE Std 1003.1-2001.

ucontext.h

User context

XSI `#include <ucontext.h>`

Derivation First released in Issue 4, Version 2.

Issue 6 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/28 is applied, updating the *getcontext()*, *makecontext()*, *setcontext()*, and *swapcontext()* functions to be obsolescent.

ulimit.h

Ulimit commands

XSI `#include <ulimit.h>`

Derivation First released in Issue 3.

Issue 6 No functional changes in this issue.

unistd.h

Standard symbolic constants and types

`#include <unistd.h>`

Derivation First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 6 `_POSIX2_C_VERSION` is removed.

The Open Group Corrigendum U026/4 is applied, adding the prototype for *fdatasync()*.

The Open Group Corrigendum U026/1 is applied, adding the following symbols:

```
_SC_XOPEN_LEGACY
_SC_XOPEN_REALTIME
_SC_XOPEN_REALTIME_THREADS
```

The symbols `_XOPEN_STREAMS` and `_SC_XOPEN_STREAMS` are added to support the XSI STREAMS Option Group.

Text in the DESCRIPTION relating to conformance requirements is moved elsewhere in IEEE Std 1003.1-2001.

The legacy symbol `_SC_PASS_MAX` is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The `_CS_POSIX_*` and `_CS_XBS5_*` constants are added for the *confstr()* function.
- The `_SC_XBS5_*` constants are added for the *sysconf()* function.
- The symbolic constants `F_ULOCK`, `F_LOCK`, `F_TLOCK`, and `F_TEST` are added.
- The `uid_t`, `gid_t`, `off_t`, `pid_t`, and `useconds_t` types are mandated.

The `gethostname()` prototype is added for sockets:

```
int gethostname(char *, size_t);
```

A new section is added for System-Wide Options.

Function prototypes for `setegid()` and `seteuid()` are added:

```
int setegid(gid_t);
int seteuid(uid_t);
```

Option symbolic constants are added for `_POSIX_ADVISORY_INFO`, `_POSIX_CPUTIME`, `_POSIX_SPAWN`, `_POSIX_SPORADIC_SERVER`, `_POSIX_THREAD_CPUTIME`, `_POSIX_THREAD_SPORADIC_SERVER`, and `_POSIX_TIMEOUTS`, and `pathconf()` variables are added for `_PC_ALLOC_SIZE_MIN`, `_PC_REC_INCR_XFER_SIZE`, `_PC_REC_MAX_XFER_SIZE`, `_PC_REC_MIN_XFER_SIZE`, and `_PC_REC_XFER_ALIGN` for alignment with IEEE Std 1003.1d-1999.

The following are added for alignment with IEEE Std 1003.1j-2000:

- Option symbolic constants `_POSIX_BARRIERS`, `_POSIX_CLOCK_SELECTION`, `_POSIX_MONOTONIC_CLOCK`, `_POSIX_READER_WRITER_LOCKS`, `_POSIX_SPIN_LOCKS`, and `_POSIX_TYPED_MEMORY_OBJECTS`
- `sysconf()` variables `_SC_BARRIERS`, `_SC_CLOCK_SELECTION`, `_SC_MONOTONIC_CLOCK`, `_SC_READER_WRITER_LOCKS`, `_SC_SPIN_LOCKS`, and `_SC_TYPED_MEMORY_OBJECTS`

The `_SC_XBS5` macros associated with the ISO/IEC 9899:1990 standard are marked LEGACY, and new equivalent `_SC_V6` macros associated with the ISO/IEC 9899:1999 standard are introduced.

The `getwd()` function is marked LEGACY. The individual reference page should be consulted for information on alternative recommended interfaces.

The **restrict** keyword is added to the prototypes for `readlink()` and `swab()`.

Constants for options are now harmonized, so when supported they take the year of approval of IEEE Std 1003.1-2001 as the value.

The following are added for alignment with IEEE Std 1003.1q-2000:

- Optional symbolic constants `_POSIX_TRACE`, `_POSIX_TRACE_EVENT_FILTER`, `_POSIX_TRACE_LOG`, and `_POSIX_TRACE_INHERIT`
- The `sysconf()` symbolic constants `_SC_TRACE`, `_SC_TRACE_EVENT_FILTER`, `_SC_TRACE_LOG`, and `_SC_TRACE_INHERIT`

The `brk()` and `sbrk()` legacy functions are removed.

The Open Group Base Resolution bwg2001-006 is applied, which reworks the XSI versioning information.

The Open Group Base Resolution bwg2001-008 is applied, changing the `namelen` parameter for `gethostname()` from **socklen_t** to **size_t**.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/2 is applied, changing “Thread Stack Address Size” to “Thread Stack Size Attribute”.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/20 is applied, adding the `_POSIX_IPV6`, `_SC_V6`, and `_SC_RAW_SOCKETS` symbols.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/21 is applied, correcting the description in “Constants for Functions” for the `_CS_POSIX_V6_LP64_OFF64_CFLAGS`, `_CS_POSIX_V6_LP64_OFF64_LDFLAGS`, and `_CS_POSIX_V6_LP64_OFF64_LIBS` symbols.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/22 is applied, removing the shading for the `_PC*` and `_SC*` constants, since these are mandatory on all implementations.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/23 is applied, adding the `_PC_SYMLINK_MAX` and `_SC_SYMLINK_MAX` constants.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/24 is applied, correcting the shading and margin code for the `fsync()` function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/25 is applied, adding the following text to the APPLICATION USAGE section: “New applications should not use `_XOPEN_SHM` or `_XOPEN_ENH_I18N`”.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/29 is applied, clarifying the requirements for when constants for Options and Option Groups can be defined or undefined.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/30 is applied, changing the `_V6_ILP32_OFF32`, `_V6_ILP32_OFFBIG`, `_V6_LP64_OFF64`, and `_V6_LP64_OFFBIG` symbols to `_POSIX_V6_ILP32_OFF32`, `_POSIX_V6_ILP32_OFFBIG`, `_POSIX_V6_LP64_OFF64`, and `_POSIX_V6_LP64_OFFBIG`, respectively. This is for consistency with the `sysconf()` and `c99` reference pages.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/31 is applied, adding that the format of names of programming environments can be obtained using the `getconf -v` option.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/32 is applied, deleting the `_SC_FILE_LOCKING`, `_SC_2_C_VERSION`, and `_SC_XOPEN_XCU_VERSION` constants.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/33 is applied, adding `_SC_SS_REPL_MAX`, `_SC_TRACE_EVENT_NAME_MAX`, `_SC_TRACE_NAME_MAX`, `_SC_TRACE_SYS_MAX`, and `_SC_TRACE_USER_EVENT_MAX` to the list of symbolic constants for `sysconf()`.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/34 is applied, updating the prototype for the `symlink()` function to match that in the System Interfaces volume of IEEE Std 1003.1-2001.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/35 is applied, adding `_PC_2_SYMLINKS` to the symbolic constants list for `pathconf()`. This corresponds to the definition of `POSIX2_SYMLINKS` in the Shell and Utilities volume of IEEE Std 1003.1-2001.

utime.h

Access and modification times structure

```
#include <utime.h>
```

Derivation First released in Issue 3.

Issue 6 The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **time_t** type is defined.

utmpx.h

User accounting database definitions

```
xSI #include <utmpx.h>
```

Derivation First released in Issue 4, Version 2.

Issue 6 No functional changes in this issue.

wchar.h

Wide-character handling

```
#include <wchar.h>
```

Derivation First released in Issue 4.

Issue 6 The Open Group Corrigendum U021/10 is applied. The prototypes for *wcswidth()* and *wcwidth()* are marked as extensions.

The Open Group Corrigendum U028/5 is applied, correcting the prototype for the *mbsinit()* function.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- Various function prototypes are updated to add the **restrict** keyword.
- The functions *vwscanf()*, *vswscanf()*, *wcstof()*, *wcstold()*, *wcstoll()*, and *wcstoull()* are added.

The type **wctype_t**, the *isw*()*, *to*()*, and *wctype()* functions are marked as XSI extensions.

wctype.h

Wide-character classification and mapping utilities

```
#include <wctype.h>
```

Derivation First released in Issue 5. Derived from ISO C Amendment 1 (MSE).

Issue 6 The *iswblank()* function is added for alignment with the ISO/IEC 9899:1999 standard:

```
int iswblank(wint_t);
```

wordexp.h

Word-expansion types

```
#include <wordexp.h>
```

Derivation First released in Issue 4. Derived from ISO/IEC 9945-2:1993 (POSIX-2).

Issue 6 The **restrict** keyword is added to the prototype for *wordexp()*:

```
int wordexp(const char *restrict, wordexp_t *restrict, int);
```

The WRDE_NOSYS constant is marked obsolescent.

ISO C Migration

By Finnbar P. Murphy, Compaq Computer Corporation

Finnbar P. Murphy is a software engineer in the Business Critical Systems Group (BCSG) at Compaq Computer Corporation in Nashua, New Hampshire. He can be contacted via email at Finnbar.Murphy@Compaq.com.

16.1 Introduction

The original ISO/IEC C language programming standard (the ISO/IEC 9899: 1990 standard) was adopted by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) in 1990. Subsequently, two technical corrigenda (TC1 and TC2) were approved together with the normative ISO C Amendment 1 (MSE), Multibyte Support Extension.

At the end of 1993, there was general agreement that work should start on the next revision of the standard. The revised standard (C99) was sent for FCD ballot in August 1998, and adopted by ISO/IEC in 1999 as the ISO/IEC 9899: 1999 standard.

This chapter is intended to provide the reader with a good, but not exhaustive, overview of the differences between the two revisions of the standard. Thus the reader is strongly advised to reference the ISO/IEC 9899: 1999 standard for specific details.

16.2 Language Changes

A significant number of changes occurred in the standard, including new keywords and types, type qualifiers, better floating-point support, and support for complex numbers.

16.2.1 New Keywords

The following new keywords were defined:

- **inline**
- **restrict**
- **_Bool**
- **_Complex**
- **_Imaginary**
- **long long**

16.2.2 New Types

Two new types were added:

- **_Bool**
- **long long**

The **long long** type is an integer type with at least 64 bits of precision.

Note: In some programming models such as LP64 and ILP64, **long long** and **long** are equivalent. In the others—for example, LLP64—**long long** is larger than **long**.

16.2.3 Type Qualifiers

Type qualifiers are now idempotent. If a type qualifier appears more than once (either directly or indirectly) in a type specification, it is as if it appeared only once. Thus **const const int *fpm***; and **const int *fpm***; are equivalent.

restrict is a new type qualifier which enables programs to be written so that compilers can produce significantly faster executables. It is intended to be used only with pointers. Objects referenced through a **restrict**-qualified pointer are special in that all references to the object must directly or indirectly use the value of the **restrict**-qualified pointer. It is intended to facilitate better alias analysis by compilers. In the absence of this qualifier, other pointers can alias the object and prevent compiler optimizations since a compiler may not be able to determine that different pointers are being used to reference different objects. Note that a restricted pointer and a non-restricted pointer can be aliases.

A number of function definitions were modified to take advantage of the **restrict** qualifier. A typical example is the *fopen()* function which was changed from:

```
FILE *fopen(const char *filename, const char *mode);
```

to:

```
FILE *fopen(const char *restrict filename,
            const char *restrict mode);
```

Changed functions include:

<i>fgetpos()</i>	<i>freopen()</i>	<i>memcpy()</i>	<i>strncpy()</i>	<i>vwprintf()</i>
<i>fgets()</i>	<i>fwprintf()</i>	<i>setbuf()</i>	<i>strxfrm()</i>	<i>wcstod()</i>
<i>fgetws()</i>	<i>fwrite()</i>	<i>setvbuf()</i>	<i>swprintf()</i>	<i>wcstol()</i>
<i>fopen()</i>	<i>fwscanf()</i>	<i>strcat()</i>	<i>swscanf()</i>	<i>wcstombs()</i>
<i>fputs()</i>	<i>mbstowcs()</i>	<i>strcpy()</i>	<i>vfwprintf()</i>	<i>wcstoul()</i>
<i>fread()</i>	<i>mbtowc()</i>	<i>strncat()</i>	<i>vswprintf()</i>	<i>wprintf()</i>

16.2.4 Boolean

The standard now supports a boolean type **_Bool** which is an integer type which can hold either 0 or 1.

The header `<stdbool.h>` also defines the macro `bool` which expands to **_Bool**, `true` which expands to the integer constant 1, and `false` which expands to the integer constant 0.

16.2.5 Universal Character Names

Prior to this revision of the standard, “native” characters, in the form of multibyte and wide characters, could be used in string literals and character constants, but not as part of an identifier.

This standard introduced the concept of a universal character name (UCN) that may be used in identifiers, character constants, and string literals to designate characters that are not in the basic character set.

The two forms of a UCN are:

```
\unnnn      where nnnn is hex-quad
\Uxxxxxxxx  where xxxxxxxx is hex-quad hex-quad
```

A *hex-quad* consists of 4 hexadecimal digits.

The UNC `\Uxxxxxxxx` designates the character whose eight-digit short identifier as specified by the ISO/IEC 10646-1:2000 standard is *xxxxxxxx*.

Similarly, the UCN `\unnnn` can be used to designate a given character whose four-digit short identifier as specified by the ISO/IEC 10646-1:2000 standard is *nnnn* (and whose eight-digit short identifier is `0000nnnn`).

There are a number of disallowed characters; that is, those in the basic character set, and code positions reserved in the ISO/IEC 10646-1:2000 standard for control and DELETE characters and UTF-16.

Note: A strictly conforming program may use only the extended characters listed in Annex I (Universal Character Names for Identifiers) and may not begin an identifier with an extended digit. Also, use of native characters in comments has always been strictly conforming, though what happens when such a program is printed in a different locale is unspecified.

16.2.6 inline

The **inline** keyword is intended to provide users with a portable way to suggest to implementations that inlining a function might result in program optimizations.

It is a function-specifier that can be used only in function declarations. It was adopted from C++ but extended in such a way that it can be implemented with existing linker technology. The translation unit that contains the definition of an inline function is the unit that provides the external definition for the function. If a function is declared inline in one translation unit, it need not be declared inline in every other translation unit.

16.2.7 Predefined Identifiers

Predefined identifiers are variables that have block scope.

The standard defined one predefined identifier `__func__` which is declared implicitly by the compiler as if, immediately following the opening brace of each function definition, the following declaration was included in the source code:

```
static const char __func__[ ] = "function-name";
```

where *function-name* is the name of the lexically-enclosing function. This enables a function name to be obtained at runtime.

The `assert()` macro now includes the identifier `__func__` in the output to `stderr`:

```
void assert(scalar expression);
```

Note that the parameter type of the `assert()` macro was also changed from **int** to **scalar**.

16.2.8 Compound Literals

Compound literals (also known as anonymous aggregates) provide a mechanism for specifying constants of aggregate or union type. This eliminates the requirement for temporary variables when an aggregate or union value may only be needed once. Compound literals are primary expressions which can also be combined with designated initializers to form an even more convenient aggregate or union constant notation.

Compound literals are created using the notation:

```
( type-name ) { initializer-list }
```

For example:

```
int *ap = (int a[]) {1, 2, 3};
```

Note that a trailing comma before the closing brace is permitted.

16.2.9 Designated Initializers

Designated initializers provide a mechanism for initializing aggregates such as sparse arrays, a common requirement in numerical programming. This mechanism also allows initialization of sparse structures and initialization of unions via any member, regardless of whether or not it is the first member.

Initializers have a named notation for initializing members. For array elements, the element is designated by [*const-expression*], for **struct** and **union** members by a dot member-name notation.

For example:

```
struct s { int a; int b; };  
struct s mystruct = {.b = 2}; // initialize member b  
struct {int a[3], b[3]} w[] = { [0].a = {1}, [1].b = 2 };
```

If an initializer is present, any members not explicitly set are zeroed out. Initializers for **auto** aggregates can be non-constant expressions.

16.3 Decimal Integer Constants

The default type of a decimal integer constant is either **int**, **long**, or **long long** (previously **int**, **long**, and **unsigned long**), depending on which type is large enough to hold the value without overflow.

The standard added LL to specify **long long**, and ULL to specify **unsigned long long**.

16.3.1 String Literals

The standard defines a number of macros as expanding into character string literals that are frequently needed as wide strings.

One example is the format specifier macros in `<inttypes.h>`. Rather than specifying two forms of each macro, one character string literal and one wide string literal, the decision was made to define the result of concatenating a character string literal and a wide string literal as a wide string literal.

16.4 Implicit Declarations

Implicit declaration of functions is no longer permitted by the standard. There must be a least one type specifier otherwise a diagnostic is issued. However, after issuing the diagnostic, an implementation may choose to assume an implicit declaration and continue translation in order to support existing source code.

For example, the declaration `fpm();` was valid in previous revisions of the standard (equivalent to `int fpm();`) but is now invalid.

16.4.1 `sizeof`

With the addition of variable length arrays, the `sizeof` operator is a constant expression only if the type of the operand is not a variable length array type.

Note: It is still possible to determine the number of elements in a variable length array `vla` with `sizeof(vla)/sizeof(vla[0])`.

16.4.2 Multiplicative Operators

In previous revisions of the standard, division of integers involving negative operands could round upward or downward in an implementation-defined manner. The standard now mandates that, as in Fortran, the result always truncates toward zero.

For example, both of the following truncate towards zero:

```
-22 / 7 = -3  
-22 % 7 = -1
```

This was done to facilitate porting of code from Fortran to C.

16.4.3 Enumeration Specifiers

A common extension to many C implementations is to allow a trailing comma after the list of enumeration constants. The standard now permits this.

16.5 Variable Length Array

A new array type, called a variable length array type, was added to the standard. The number of elements specified in the declaration of a variable length array type is not specified by the source code; rather it is a computed value determined at runtime.

Multi-dimensional variable-length arrays are permitted.

Some things cannot be declared as a variable length array type, including:

- File scope identifiers
- Arrays declared using either **static** or **extern** storage class specifiers
- Structure and union members

The rationale behind this new array type was that some standard method to support runtime array sizing was considered crucial for C's acceptance in the numerical computing world. Before this revision of the standard, the size expression was required to be an integer constant expression.

16.5.1 Array Declarations

The **static** storage class specifier and the type-qualifiers **restrict**, **const**, or **volatile** can now be used inside the square brackets of an array type declaration, but only in the outermost array type derivation of a function parameter.

```
int foo(const int a[static 10]);
```

In the above example, the **static** keyword will guarantee that the pointer to the array *a* is not NULL, and points to an object of the appropriate type.

16.5.2 Array Type Compatibility

Array type compatibility was extended so that variable length arrays are compatible with both an array of known constant size and an array with an incomplete type.

16.5.3 Incomplete Array Structure Members

The last member of a structure with more than one member can now be an incomplete array type. This incomplete member is called a flexible array member.

Consider the following example

```
struct s { int n;
          double d[];
};

size_t sz = sizeof( struct s );
struct s *sp = malloc(sz + 10);
```

The structure pointer **sp** behaves as if the structure **s** had been declared as

```
struct s { int n;
          double d[10];
};
```

The size of the structure is equal to the offset of the last element of an otherwise identical structure that replaces the flexible array member with an array of unspecified length. When a `'.'` or a `'->'` operator point to a structure with a flexible array member and the right operand names that member, it behaves as if that member were replaced with the longest array with the

same element type that would not make the structure larger than the object being accessed.

The offset of the array remains that of the flexible array member, even if this would differ from that of the replacement array. If this array would have no elements, it behaves as if it had one element. However, behavior is undefined if any attempt is made to access that element or to generate a pointer one past it.

16.5.4 Blocks

A common coding practice is to always use compound statements for every selection and iteration statement to guard against inadvertent problems when changes are made to the source code.

Because this can lead to surprising behavior in connection with certain uses of compound literals, the concept of a block was expanded in this revision of the standard.

As in C++, all selection and iteration statements, and their associated substatements, are now defined to be blocks, even if they are not also compound statements. If compound literals are defined in selection or iteration statements, their lifetimes are limited to the implied enclosing block.

16.5.5 The for Statement

The standard now permits loop counter variables as part of a **for** statement. Such a variable is in a new scope (so it does not affect any other variable of the same name), is destroyed at the end of the loop, and must have **auto** or **register** storage class.

```
for (int i = 0; i < 10; i++)
    printf("Loop number: %d\n", i);
```

16.5.6 errno

For underflow, *errno* is no longer required to be set to [EDOM] or [ERANGE].

16.6 Comments

Support for *//*-style comments was added due to their utility and widespread existing practice, especially in dual C/C++ translators. This is a quiet change which could cause different semantics between this standard and C89. Consider the following example:

```
a = b /*divisor:*/ f
    + e;
```

According to this standard this is the same as:

```
a = b + e;
```

but in previous revisions of the standard it was the same as:

```
a = b / f + e;
```

16.6.1 Hexadecimal Floating-Point Constants

Because hexadecimal notation more clearly expresses the significance of floating constants, the standard now supports hexadecimal floating-point constants.

The binary-exponent part is required, instead of being optional as it is for decimal notation, to avoid ambiguity resulting from an 'f' suffix being mistaken as a hexadecimal digit. The exponent indicates the power of 2 by which the significant part is to be scaled.

16.6.2 Predefined Macros

New predefined macros include:

`__STDC_VERSION__` Defined to be 199901L to indicate the current revision of the standard.

`__STDC_HOSTED__` Defined as 1 if the implementation is hosted; otherwise, 0.

16.6.3 Source File Inclusion

The number of significant characters in header and source file names was raised from six to eight, and digits are now allowed.

16.6.4 Translation-Time Arithmetic

The standard now mandates that translation-time arithmetic be done using `intmax_t` or `uintmax_t`, which must comprise at least 64 bits and must match the execution environment.

Previously, a translator was permitted to evaluate expressions using the `long` integer or `unsigned long` integer arithmetic native to the translation environment.

16.6.5 Minimum Maximum Line Length

The minimum maximum line length was increased from 254 to 4095.

16.6.6 Case-Sensitive Identifiers

All identifiers are now case-sensitive. In previous revisions of the standard, it was implementation-defined whether an implementation ignored the case of external identifiers.

16.6.7 #line Directive

This directive now allows the specification of a line number up to $2^{31}-1$. Previously the limit was 32767.

16.6.8 Empty Argument Macros

Empty arguments are now explicitly allowed. In previous revisions of the standard, this resulted in undefined behavior. Stringification (`#` operator) of an empty argument yields the empty string, concatenation (`##` operator) of an empty argument with a non-empty argument produces the non-empty argument, and concatenation of two empty arguments produces nothing.

16.6.9 Pragmas

Some **pragma** directives have been standardized. Directives whose first preprocessing token is **STDC** are reserved for standardized directives.

As an alternative syntax for a **pragma** directive, the preprocessing operator **_Pragma** is specified. This has the advantage that it can be used in a macro replacement list.

16.6.10 Translation Limits

A number of the program translation limits were significantly increased.

The number of significant initial characters in an internal identifier or a macro name was increased from 31 to 63.

The number of significant characters in an external identifier has increased from 6 to 31 case-sensitive characters.

Note that each universal character name (UCN) specifying a short identifier of 0000FFFF or less is considered to be 6 characters, while a long UCN counts as 10 characters.

While an implementation is not obliged to remember more than the first 63 characters of an identifier with internal linkage, or the first 31 characters of an identifier with external linkage, the programmer is effectively prohibited from intentionally creating two different identifiers that are the same within the appropriate length.

The minimum maximum limit of cases in a switch statement was increased to 1 023.

16.6.11 Token Pasting

The standard replaced non-digit with identifier-non-digit in the grammar to allow the token pasting operator, **##**, to work as expected with characters which are not part of the basic character set.

16.6.12 Variadic Macros

The standard extended the functionality of the punctuator **"..."** (ellipsis; denoting a variable number of trailing arguments) to function-like macros. For replacement, the variable arguments (including the separating commas) are “collected” into one single extra argument that can be referenced as **__VA_ARGS__** within the macro’s replacement list.

For example:

```
#define MyLog(...) fprintf(stderr, __VA_ARGS__)

main()
{
    int array_bound = 10;
    int array_index = 11;
    .....
    MyLog("ERROR: Index out of bound: %d %d\n", array_index,
        array_bound);
    .....
}
```

There must be at least one argument to match the ellipsis. This requirement avoids problems that might occur when the trailing arguments are included in a list of arguments to another macro or function.

16.6.13 `va_copy()`

In previous revisions of the standard, it was not possible to backtrack and examine one or more arguments a second time when processing a variable argument list. The only way to do this was to reprocess the variable argument list.

The `va_copy()` macro provides a mechanism for copying the `va_list` object used to represent processing of the arguments. Calling the `va_copy()` macro exactly duplicates the `va_list` object.

Note: A separate call to the `va_end()` macro is required to remove the new `va_list` object.

16.7 Headers

The following new headers were added to the standard:

<complex.h>	Defines a number of macros and functions for use with the three complex arithmetic types defined in the standard.
<fenv.h>	Defines a number of types, macros, and functions that can be used to test, control, and access an implementation's floating-point environment.
<inttypes.h>	Defines a type and a number of macros and functions for manipulating integers; <stdint.h> is a subset of this header.
<stdbool.h>	Defines a number of macros for accessing the new Boolean type _Bool and writing Boolean tests.
<tgmath.h>	Defines a large number of type-generic macros that invoke the correct math function from <math.h> or from <complex.h> depending upon their argument types.

16.8 Integer Types

The purpose of the **<inttypes.h>** header is to provide a set of integer types whose definitions are consistent across platforms. Consistent use of these integer types should greatly increase the portability of source code across platforms.

The header **<stdint.h>** is a subset of **<inttypes.h>** and may be more suitable for use in freestanding environments, which might not support the formatted I/O functions. It declares sets of integer types having specified widths and corresponding macros that specify limits of the declared types and construct suitable constants.

The following categories of integer types are defined:

- Types having exact widths
- Types having at least certain specified widths
- Fastest types having at least certain specified widths
- Types wide enough to hold pointers to objects
- Types having greatest width

16.8.1 Exact-Width Integer Types

The **typedef** name `intN_t` designates a signed integer type with width N bits, no padding bits, and a two's complement representation. The **typedef** name `uintN_t` designates an unsigned integer type with width N .

For example, `int16_t` is an unsigned integer type with a width of exactly 16 bits.

Exact-width types are optional. However, if an implementation provides integer types with widths of 8, 16, 32, or 64 bits, it must define the corresponding **typedef** names.

16.8.2 Minimum-Width Integer Types

The **typedef** names `int_leastN_t` and `uint_leastN_t`, respectively, designate signed and unsigned integer types with a width of at least N bits, such that no signed integer type with lesser size has at least the specified width.

For example, `uint_least16_t` denotes an unsigned integer type with a width of at least 16 bits.

The following types are mandatory:

```
int_least8_t   int_least32_t   uint_least8_t   uint_least32_t
int_least16_t  int_least64_t   uint_least16_t  uint_least64_t
```

16.8.3 Fastest Minimum-Width Integer Types

The **typedef** names `int_fastN_t` and `uint_fastN_t`, respectively, designate the (usually) fastest signed and unsigned integer types with a width of at least N bits.

The following types are mandatory:

```
int_fast8_t   int_fast32_t   uint_fast8_t   uint_fast32_t
int_fast16_t  int_fast64_t   uint_fast16_t  uint_fast64_t
```

16.8.4 Integer Types Capable of Holding Object Pointers

The optional `intptr_t` and `uintptr_t` types, respectively, designate a signed and unsigned integer type with the property that any valid pointer to `void` can be converted to this type, then converted back to a pointer to `void` and the result will compare equal to the original pointer.

16.8.5 Greatest-Width Integer Types

The `intmax_t` and `uintmax_t` types, respectively, designate a signed integer type capable of representing any value of any signed integer type, and an unsigned integer type capable of representing any value of any unsigned integer type.

For each type declared in `<stdint.h>` conversion macros, which expand to the correct format specifiers, are defined for use with the formatted input/output functions such as `fprintf()` and `fscanf()`.

The `fprintf()` macros for signed integers are:

```
PRIdFASTN   PRIdMAX   PRIdPRT   PRIiLEASTN   PRIiN
PRIdLEASTN  PRIdN     PRIiFASTN  PRIiMAX     PRIiPRT
```

The `PRId` macros each expand to a string literal suitable for use as a `d` print conversion specifier, plus any needed qualifiers, to convert values of the types `int8_t`, `int16_t`, `int32_t`, or `int64_t`, respectively.

The PRIdLEAST macros each expand to a string literal suitable for use as a d print conversion specifier, plus any needed qualifiers, to convert values of the types `int_least8_t`, `int_least16_t`, `int_least32_t`, or `int_least64_t`, respectively.

The PRIdFAST macros each expand to a string literal suitable for use as a d print conversion specifier, plus any needed qualifiers, to convert values of the types `int_fast8_t`, `int_fast16_t`, `int_fast32_t`, or `int_fast64_t`, respectively.

The PRIdMAX macro expands to a string literal suitable for use as a d print conversion specifier, plus any needed qualifiers, to convert values of the type `intmax_t`.

The PRIdPTR macro expands to a string literal suitable for use as a d print conversion specifier, plus any needed qualifiers, to convert values of the type `intptr_t`.

The following example shows part of one possible implementation of these macros in an LP64 programming model:

```
#define PRId8      "hhd"
#define PRId16    "hd"
#define PRId32    "d"
#define PRId64    "ld"

#define PRIdFAST8  "hhd"
#define PRIdFAST16 "hd"
#define PRIdFAST32 "d"
#define PRIdFAST64 "ld"

#define PRIdLEAST8  "hhd"
#define PRIdLEAST16 "hd"
#define PRIdLEAST32 "d"
#define PRIdLEAST64 "ld"
```

Corresponding `fprintf()` macros are defined for unsigned integers (PRIo, PRlu, PRlx, and PRlX).

For `fscanf()`, the macro names start with SCN instead of PRN. (SCNd and SCNi for signed integers; SCNo, SCNu, SCNx for unsigned integers.)

A new structure type `imaxdiv_t` is defined. It is the type of structure returned by `imaxdiv()`.

The following function computes the absolute value of an integer *j*:

```
intmax_t imaxabs(intmax_t j);
```

The following function computes both the quotient and remainder in a single operation:

```
imaxdiv_t imaxdiv(intmax_t numer, intmax_t denom);
```

The following functions are equivalent to the `strtol` family of functions, except that the initial portion of the string is converted to `intmax_t` and `uintmax_t` representation, respectively:

```
intmax_t strtoumax(const char *restrict nptr,
                  char **restrict endptr, int base);
uintmax_t strtoumax(const char *restrict nptr,
                   char **restrict endptr, int base);
```

The following functions are equivalent to the `wcstol` family of functions, except that the initial portion of the wide string is converted to `intmax_t` and `uintmax_t` representation, respectively:

```
intmax_t wcstoumax(const wchar_t *restrict nptr,
                  wchar_t **restrict endptr, int base);
uintmax_t wcstoumax(const wchar_t *restrict nptr,
```

```
wchar_t **restrict endptr, int base);
```

16.8.6 Limits of Specified-Width Integer Types

The standard specifies the minimum and maximum limits for all of the types declared in the `<stdint.h>` header.

For example, the minimum value of an exact-width unsigned integer type is `{UINTn_MAX}`, where n is an unsigned decimal integer with no leading zeros, whose value is exactly $2^n - 1$.

16.8.7 Macros

The macros `INTN_C()` and `UINTN_C()` expand to a signed integer constant whose type and value is `int_leastN_t`, and an unsigned integer constant whose type and value is `uint_leastN_t`, respectively.

The macro `INTMAX_C()` expands to a integer constant whose type is `intmax_t`, and `UINTMAX_C()` expands to an unsigned integer constant whose type `uintmax_t`.

16.9 Complex Numbers

Support for complex numbers and complex number arithmetic is new, and was added as part of the effort to make the C language more attractive for general numerical programming. The underlying implementation of the complex types is explicitly stated to be Cartesian, rather than polar, for consistency with other programming languages. Thus values are interpreted as radians, not degrees.

The header `<complex.h>` contains the macro definitions and function declarations that support complex arithmetic.

Two new type specifiers were defined:

`_Complex`

`_Imaginary` (Only if an implementation supports a pure imaginary type.)

A new type qualifier `complex` (actually a macro which expands to `_Complex`) is used to denote a number as being a complex number.

Three complex types were defined:

- **float complex**
- **double complex**
- **long double complex**

The corresponding real type is the type obtained by deleting the type qualifier `complex` from the complex type name.

A complex type has the same representation and alignment requirements as an array type containing exactly two elements of the corresponding real type; the first element is equal to the real part, and the second element to the imaginary part, of the complex number.

There is no special syntax for constants; instead there is a new macro `_Complex_I`, which has a complex value whose real part is zero and whose imaginary part is x . Note that `_Complex_I*_Complex_I` has a value of -1 , but the type of that value is `complex`.

The standard reserves the keyword **_Imaginary** for use as a type-specifier in conjunction with the pure imaginary type. The macros **imaginary** and **Imaginary_I** are defined only if an implementation supports a pure **imaginary** type. Such support is optional. See Annex G of the standard for further details. If defined, they expand to **_Imaginary** and a constant expression of type **const float _Imaginary** with the value of the imaginary unit.

The macro **I** expands to **_Imaginary_I**, if defined, else to **_Complex_I**. Thus a complex number constant (3.0+4.0i) could be written as either 3.0+4.0*I or 3.0+4.0*_Complex_I.

The choice of 'I' instead of 'i' for the imaginary unit was because of the widespread use of the identifier 'i' for other purposes. A program can use a different identifier (for example, 'z') for the imaginary unit by undefining 'I' and defining 'z' as follows:

```
#include <complex.h>
#undef I
#define z _Imaginary_z
```

Annex G, which is marked informative, specifies complex arithmetic intended to be compatible with the IEC 60559:1989 standard real floating-point arithmetic. This annex was designated as informative because of insufficient prior art for normative status. An implementation claiming such conformance should define **__STDC_IEC_559_COMPLEX** to be 1.

The **pragma STDC CX_LIMITED_RANGE** can be used to indicate (ON) that the usual mathematical formulas for complex arithmetic may be used. Such formulas are problematic because of overflow, underflow, and handling of infinities.

The following new functions relate to complex arithmetic.

16.9.1 Trigonometric Functions

The complex arc cosine functions compute the complex arc cosine of z , with branch cuts outside the interval $[-1,+1]$ along the real axis.

```
double complex cacos(double complex z);
float complex cacosf(float complex z);
long double complex cacosl(long double complex z);
```

The complex arc sine functions compute the complex arc sine of z , with branch cuts outside the interval $[-1,+1]$ along the real axis.

```
double complex casin(double complex z);
float complex casinf(float complex z);
long double complex casinl(long double complex z);
```

The complex arc tangent functions compute the complex arc tangent of z , with branch cuts outside the interval $[-i,+i]$ along the imaginary axis.

```
double complex catan(double complex z);
float complex catanf(float complex z);
long double complex catanl(long double complex z);
```

The complex cosine functions compute the complex cosine of z .

```
double complex ccos(double complex z);
float complex ccosf(float complex z);
long double complex ccosl(long double complex z);
```

The complex sine functions compute the complex sine of z .


```
double complex csin(double complex z);
float complex csinf(float complex z);
long double complex csinl(long double complex z);
```

The complex tangent functions compute the complex tangent of z .

```
double complex ctan(double complex z);
float complex ctanf(float complex z);
long double complex ctanl(long double complex z);
```

16.9.2 Hyperbolic Functions

The complex arc hyperbolic cosine functions compute the complex arc hyperbolic cosine of z , with a branch cut at values less than 1 along the real axis.

```
double complex cacosh(double complex z);
float complex cacoshf(float complex z);
long double complex cacoshl(long double complex z);
```

The complex arc hyperbolic sine functions compute the complex arc hyperbolic sine of z , with branch cuts outside the interval $[-i,+i]$ along the imaginary axis.

```
double complex casinh(double complex z);
float complex casinhf(float complex z);
long double complex casinhl(long double complex z);
```

The complex arc hyperbolic tangent functions compute the complex arc hyperbolic tangent of z , with branch cuts outside the interval $[-1,+1]$ along the real axis.

```
double complex catanh(double complex z);
float complex catanhf(float complex z);
long double complex catanhl(long double complex z);
```

The complex hyperbolic cosine functions compute the complex hyperbolic cosine of z .

```
double complex ccosh(double complex z);
float complex ccoshf(float complex z);
long double complex ccoshl(long double complex z);
```

The complex hyperbolic sine functions compute the complex hyperbolic sine of z .

```
double complex csinh(double complex z);
float complex csinhf(float complex z);
long double complex csinhl(long double complex z);
```

The complex hyperbolic tangent functions compute the complex hyperbolic tangent of z .

```
double complex ctanh(double complex z);
float complex ctanhf(float complex z);
long double complex ctanhl(long double complex z);
```

16.9.3 Exponential and Logarithmic Functions

The complex exponential functions compute the complex base-e exponential of z .

```
double complex cexp(double complex z);
float complex cexpf(float complex z);
long double complex cexpl(long double complex z);
```

The complex natural logarithm functions compute the complex natural logarithm of z , with a branch cut along the negative real axis.

```
double complex clog(double complex z);
float complex clogf(float complex z);
long double complex clogl(long double complex z);
```

16.9.4 Power and Absolute-Value Functions

The complex absolute value functions compute the modulus of x .

```
double complex cabs(double complex x);
float complex cabsf(float complex x);
long double complex cabsl(long double complex x);
```

The complex power functions compute the complex power function x^y , with a branch cut for the first parameter along the negative real axis.

```
double complex cpow(double complex x, double complex y);
float complex cpowf(float complex x, float complex y);
long double complex cpowl(long double complex x,
    long double complex y);
```

The complex square root functions compute the complex square root of z , with a branch cut along the negative real axis.

```
double complex csqrt(double complex z);
float complex csqrtf(float complex z);
long double complex csqrtl(long double complex z);
```

16.9.5 Manipulation Functions

The complex argument functions compute the argument of z , with a branch cut along the negative real axis.

```
double carg(double complex z);
float cargf(float complex z);
long double cargl(long double complex z);
```

The complex imaginary functions compute the imaginary part of z .

```
double cimag(double complex z);
float cimagf(float complex z);
long double cimagl(long double complex z);
```

The complex conjugate functions compute the complex conjugate of z , by reversing the sign of its imaginary part.

```
double complex conj(double complex z);
float complex conjf(float complex z);
long double complex conjl(long double complex z);
```

The complex projection functions compute a projection of z onto the Riemann sphere.

```
double complex cproj(double complex z);
float complex cprojf(float complex z);
long double complex cprojl(long double complex z);
```

The complex real functions compute the real part of z .

```
double creal(double complex z);
float crealf(float complex z);
long double creall(long double complex z);
```

Note that no errors are defined for any of the above functions.

16.10 Other Mathematical Changes

The standard extended the mathematical support via `<math.h>` by providing versions of functions to support **float** and **long double** as well as the existing double floating type functions.

The functions `ecvt()`, `fcvt()`, and `gcvt()` were dropped from the standard since their capability is available using the `sprintf()` function.

The **pragma STDC FP_CONTACT** indicates to an implementation whether it is allowed (ON) or disallowed (OFF) to contract expressions; that is, evaluated as though an expression is an atomic operation, thereby omitting certain rounding errors.

The macro `NAN` is defined only if an implementation supports quiet NaNs.

16.10.1 Classification Macros

The following are defined for use with classification macros:

<code>FP_NAN</code>	The floating-point number x is “Not a Number”.
<code>FP_INFINITE</code>	The value of the number is either plus or minus infinity.
<code>FP_ZERO</code>	The value of the number is either plus or minus zero.
<code>FP_SUBNORMAL</code>	The number is in denormalized format.
<code>FP_NORMAL</code>	There is nothing special about the number.

The macro `fpclassify()` classifies its argument as either NaN, infinite, normal, subnormal, zero, or into another implementation-defined category.

```
int fpclassify(real-floating x);
```

The macro `isfinite()` determines whether its argument has a finite value (zero, subnormal, or normal, and not infinite or NaN).

```
int isfinite(real-floating x);
```

The macro `isinf()` determines whether its argument value is an infinity (positive or negative).

```
int isinf(real-floating x);
```

The macro `isnan()` determines whether its argument value is a NaN.

```
int isnan(real-floating x);
```

The macro `isnormal()` determines whether its argument value is normal (neither zero, subnormal, infinite, nor NaN).

```
int isnormal(real-floating x);
```

The macro *signbit()* determines whether the sign of its argument value is negative.

```
int signbit(real-floating x);
```

16.10.2 Trigonometric Functions

The following functions compute the arc cosine, arc sin, and arctan of x , respectively:

```
float acosf(float x);
long double acosl(long double x);
float asinf(float x);
long double asinl(long double x);
long double tanl(long double x);
```

The following functions compute the arc tangent of y/x :

```
float atan2f(float y, float x);
long double atan2l(long double y, long double x);
```

The following functions compute the cosine, sin, and tangent of x , respectively:

```
float cosf(float x);
long double cosl(long double x);
float sinf(float x);
long double sinl(long double x);
float tanf(float x);
long double tanl(long double x);
```

16.10.3 Hyperbolic Functions

The following functions compute the arc hyperbolic cosine of x :

```
float acoshf(float x);
long double acoshl(long double x);
```

The following functions compute the arc hyperbolic sine of x :

```
float asinhf(float x);
long double asinhl(long double x);
```

The following functions compute the arc hyperbolic tangent of x :

```
float atanhf(float x);
long double atanh1(long double x);
```

The following functions compute the hyperbolic cosine of x :

```
float coshf(float x);
long double coshl(long double x);
```

The following functions compute the hyperbolic sine of x :

```
float sinh1(float x);
long double sinh1(long double x);
```

The following functions compute the hyperbolic tangent of x :

```
float tanhf(float x);
long double tanhl(long double x);
```

16.10.4 Exponential and Logarithmic Functions

The following functions compute the base-e exponential of x :

```
float expf(float x);
long double expl(long double x);
```

The following functions compute the base-2 exponential of x :

```
double exp2(double x);
float exp2f(float x);
long double exp2l(long double x);
```

The following functions compute the base-e exponential of $(x-1)$:

```
float expm1f(float x);
long double expm1l(long double x);
```

The following functions break a floating-point number into a normalized fraction and an integral power of 2:

```
float frexpf(float value, int *exp);
long double frexpl(long double value, int *exp);
```

The following functions extract the exponent of x :

```
int ilogbf(float x);
int ilogbl(long double x);
```

The following functions multiply a floating-point number by an integral power of 2:

```
float ldexpf(float x, int exp);
long double ldexpl(long double x, int exp);
```

The following functions compute the natural logarithm of x :

```
float logf(float x);
long double logl(long double x);
```

The following functions compute the base-10 logarithm of x :

```
float log10f(float x);
long double log10l(long double x);
```

The following functions compute the natural logarithm of $(x+1)$:

```
float log1pf(float x);
long double log1pl(long double x);
```

The following functions compute the base-2 logarithm of x :

```
double log2(double x);
float log2f(float x);
long double log2l(long double x);
```

The following functions extract the exponent of x :

```
float logbf(float x);
long double logbl(long double x);
```

The following functions break x into integral and fractional parts:

```
float modff(float x, float *iptr);
long double modfl(long double x, long double *iptr);
```

The following functions compute $x^{\text{FLT_RADIX}^n}$ efficiently:

```
double scalbn(double x, int n);
float scalbnf(float x, int n);
long double scalbnl(long double x, int n);
double scalbln(double x, long int n);
float scalblnf(float x, long int n);
long double scalblnl(long double x, long int n);
```

The following functions compute the real cube root of x :

```
double cbrt(double x);
float cbrtf(float x);
long double cbrtl(long double x);
```

The following functions compute the absolute value of x :

```
float fabsf(float x);
long double fabsl(long double x);
```

The following functions compute the square root of the sum of the squares of x and y :

```
float hypotf(float x, float y);
long double hypotl(long double x, long double y);
```

The following functions compute x raised to the power of y :

```
float powf(float x, float y);
long double powl(long double x, long double y);
```

The following functions compute the non-negative square root of x :

```
float sqrtf(float x);
long double sqrtl(long double x);
```

The following functions compute the error function of x :

```
float erff(float x);
long double erfl(long double x);
```

The following functions compute the natural logarithm of the absolute value of the gamma function of x :

```
float lgammaf(float x);
long double lgammal(long double x);
```

The following functions compute the (true) gamma function of x :

```
double tgamma(double x);
float tgammaf(float x);
long double tgammal(long double x);
```

16.10.5 Nearest Integer Functions

The following functions compute the smallest integer value not less than x :

```
float ceilf(float x);
long double ceill(long double x);
```

The following functions compute the largest integer value not greater than x :

```
float floorf(float x);
long double floorl(long double x);
```

The following functions round x to an integer value in floating-point format, using the current rounding direction and without raising the inexact floating-point exception:

```
double nearbyint(double x);
float nearbyintf(float x);
long double nearbyintl(long double x);
```

The following functions round x to an integer value in floating-point format, using the current rounding direction and may raise the inexact floating-point exception if the result differs in value from the argument:

```
float rintf(float x);
long double rintl(long double x);
```

The following functions round x to the nearest integer value, rounding according to the current rounding direction:

```
long int lrint(double x);
long int lrintf(float x);
long int lrintl(long double x);
long long int llrint(double x);
long long int llrintf(float x);
long long int llrintl(long double x);
```

The following functions round x to the nearest integer value in floating-point format, rounding halfway cases away from zero, regardless of the current rounding direction:

```
double round(double x);
float roundf(float x);
long double roundl(long double x);
```

The following functions round x to the nearest integer value, rounding halfway cases away from zero, regardless of the current rounding direction:

```
long int lround(double x);
long int lroundf(float x);
long int lroundl(long double x);
long long int llround(double x);
long long int llroundf(float x);
long long int llroundl(long double x);
```

The following functions round x to the integer value, in floating format, nearest to but no larger in magnitude than x :

```
double trunc(double x);
float truncf(float x);
long double trunc1(long double x);
```

16.10.6 Remainder Functions

The following functions compute the floating-point remainder of x/y :

```
float fmodf(float x, float y);
long double fmodl(long double x, long double y);
```

The following functions compute the IEC 60559: 1989 standard remainder $x \text{ REM } y$:

```
float remainderf(float x, float y);
long double remainderl(long double x, long double y);
```

The following functions shall compute the same remainder as the remainder family of functions, but in a different manner:

```
double remquo(double x, double y, int *quo);
float remquof(float x, float y, int *quo);
long double remquol(long double x, long double y, int *quo);
```

16.10.7 Manipulation Functions

The following functions produce a value with the magnitude of *and the sign of* y :

```
double copysign(double x, double y);
float copysignf(float x, float y);
long double copysignl(long double x, long double y);
```

The following functions return a quiet NaN, if available, with content indicated by *tagp*:

```
double nan(const char *tagp);
float nanf(const char *tagp);
long double nanl(const char *tagp);
```

The following functions determine the next representable value:

```
float nextafterf(float x, float y);
long double nextafterl(long double x, long double y);
```

The following functions are equivalent to the *nextafter* functions, except that the second parameter has type **long double** and the functions return y converted to the type of the function if x equals y :

```
double nexttoward(double x, long double y);
float nexttowardf(float x, long double y);
long double nexttowardl(long double x, long double y);
```

The following functions determine the positive difference between their arguments:

```
double fdim(double x, double y);
float fdimf(float x, float y);
long double fdiml(long double x, long double y);
```

The following functions determine the maximum numeric value of their arguments:

```
double fmax(double x, double y);
float fmaxf(float x, float y);
long double fmaxl(long double x, long double y);
```

If the optional macros `FP_FAST_FMA`, `FP_FAST_FMAF`, and `FP_FAST_FMAL` are defined, it indicates that the corresponding *fma()* function executes at least as fast as a multiply and an add of double operands.

The following functions determine the minimum numeric value of their arguments:

```
double fmin(double x, double y);
float fminf(float x, float y);
long double fminl(long double x, long double y);
```

The following functions compute $(x*y)+z$, rounded as one ternary operation:

```
double fma(double x, double y, double z);
float fmaf(float x, float y, float z);
long double fmal(long double x, long double y,
                 long double z);
```

16.10.8 Comparison Macros

The *isgreater()* macro tests whether *x* is greater than *y*.

```
int isgreater(real-floating x, real-floating y);
```

The *isgreaterequal()* macro tests whether *x* is greater than or equal to *y*.

```
int isgreaterequal(real-floating x, real-floating y);
```

The *isless()* macro tests whether *x* is less than *y*.

```
int isless(real-floating x, real-floating y);
```

The *islessequal()* macro tests whether *x* is less than or equal to *y*.

```
int islessequal(real-floating x, real-floating y);
```

The *islessgreater()* macro tests whether *x* is less than or greater than *y*.

```
int islessgreater(real-floating x, real-floating y);
```

The *isunordered()* macro tests whether *x* and *y* are unordered.

```
int isunordered(real-floating x, real-floating y);
```

Note: Annex F (normative) was added to specify the IEC 60559:1989 standard floating-point arithmetic. An implementation that defines `__STDC_IEC_559__` must conform to the specification detailed in this annex.

16.11 Floating-Point Environment Support

The header `<fenv.h>` declares the types, and defines the macros and functions that support access to an implementation's floating-point environment.

Two types are declared:

fenv_t Represents the entire floating-point environment.

fexcept_t Represents the collective floating-point status flags.

The **pragma** directive has three reserved forms, all starting with the preprocessor token **STDC**. These are used to specify certain characteristics of the floating-point support to comply with the IEC 60559: 1989 standard.

The **pragma STDC FENV_ACCESS** provides the means of informing an implementation when a program might access the floating-point environment.

For example:

```
double a;
#pragma STDC FENV_ACCESS ON
a = 1.0 + 2.0;
#pragma STDC FENV_ACCESS OFF
```

16.11.1 Exceptions

The following function clears the supported floating-point exceptions:

```
void feclearexcept(int excepts);
```

The following function stores an implementation-dependent representation of the states of the floating-point status flags:

```
void fegetexceptflag(fexcept_t *flagp, int excepts);
```

The following function raises the supported floating-point exceptions represented by its argument:

```
void feraiseexcept(int excepts);
```

The following function sets the floating-point status flags:

```
void fesetexceptflag(const fexcept_t *flagp, int excepts);
```

The following function tests which of a specified subset of the floating-point exception flags are currently set:

```
int fetestexcept(int excepts);
```

Each of the following floating-point exception macros is defined if an implementation supports these functions:

```
FE_DIVBYZERO
FE_INEXACT
FE_INVALID
FE_OVERFLOW
FE_UNDERFLOW
FE_ALL_EXCEPT (Bitwise OR of all the other macros.)
```

Additional implementation-defined floating-point exceptions, with macro definitions beginning with FE_ and an uppercase letter, may also be defined by an implementation.

16.11.2 Rounding

The following functions respectively set and return the current rounding direction:

```
int fesetround(int round);
int fegetround(void);
```

Each of the following floating-point macros is defined if an implementation supports these functions:

```
FE_DOWNWARD
FE_TONEAREST
FE_TOWARDZERO
FE_UPWARD
```

Additional implementation-defined rounding directions, with macro definitions beginning with `FE_` and an uppercase letter, may also be defined by an implementation.

16.11.3 Environment

The following functions respectively set and return the floating-point environment function:

```
void fesetenv(const fenv_t *envp);
void fegetenv(fenv_t *envp);
```

The following function saves the currently raised floating-point exception(s), installs the floating-point environment represented by the object pointed to by `envp`, and then raises the saved floating-point exception(s):

```
void feupdateenv(const fenv_t *envp);
```

The following function saves the current floating-point environment in the object pointed to by `envp`, clears the floating-point status flags, and then installs a non-stop (continue on floating-point exceptions) mode, if available, for all floating-point exceptions:

```
int feholdexcept(fenv_t *envp);
```

The macro `FE_DFL_ENV` represents the default floating-point environment; that is, the one installed at program startup. It can be used as an argument with the above functions and is of type `*const fenv_t`.

16.12 Type-Generic Math

Type-generic macros may enable the writing of more portable code, and reduce need for casting and suffixing when porting applications to new platforms.

The header `<tgmath.h>` includes the headers `<math.h>` and `<complex.h>` and defines numerous type-generic macros. Except for `modf`, there is a type-generic macro for each of the functions in `<math.h>` and `<complex.h>` that do not have an `'f'` (**float**) or `'l'` (**long double**) suffix and have one or more parameters whose corresponding real type is **double**.

Such parameters are called generic parameters.

Use of a type-generic macro invokes a function whose corresponding real type and type domain are determined by the arguments for the generic parameters. The real type is determined as follows:

1. First, if any argument for generic parameters is a **long double**, the real type is **long double**.
2. Otherwise, if any argument for generic parameters is a double or an integer type, the real type is **double**.
3. Otherwise, the real type is **float**.

Type-generic macros that accept complex arguments also accept imaginary arguments. If an argument is imaginary, the macro expands to an expression whose type is **real**, **imaginary**, or **complex**, as appropriate for the particular function.

16.12.1 Unsuffixed Functions With a C-Prefixed Counterpart

For each unsuffixed function in `<math.h>` for which there is a function in `<complex.h>` with the same name except for a 'c' prefix, the corresponding type-generic macro for both functions has the same name as the function in `<math.h>`.

For example, the type-generic macro for `tan()` and `ctan()` is `tan`.

If at least one argument for a generic parameter is complex, then use of the macro invokes a complex function; otherwise, a real function is invoked.

16.12.2 Unsuffixed Functions Without a C-Prefixed Counterpart

For each unsuffixed function in `<math.h>` for which there is not a function in `<complex.h>` with the same name but having a 'c' prefix, the corresponding type-generic macro for both functions has the same name as the function in `<math.h>`. If all arguments for generic parameters are real, then use of the macro invokes a real function; otherwise, use of the macro results in undefined behavior. Examples of such functions include `fdim()` and `lround()`.

For each unsuffixed function in `<complex.h>` for which there is not a function in `<math.h>` with the same name but without a 'c' prefix, the corresponding type-generic macro for both functions has the same name as the function in `<complex.h>`. Use of the macro with any real or complex argument invokes a complex function.

16.13 Other Library Changes

A number of new functions were added to the standard, prototypes for many functions now contain the new keyword **restrict** as part of some parameter declarations, and a number of functions had their definition clarified or extended.

atoll()

A numeric conversion function for the conversion of a string to a **long long int** representation.

```
long long int atoll(const char *nptr);
```

_Exit()

This function causes normal program termination to occur and control to be returned to the host environment without triggering signals or `atexit()` registered functions.

This function name (rather than `_exit()`) was chosen to avoid potential conflict with existing practice.

fpos_t

The description of **fpos_t** was changed to exclude array type objects.

isblank()

This function tests whether *c* is a character of class **blank** in a program's current locale.

```
int isblank(int c);
```

iswblank()

This function tests whether *wc* is a wide-character which is a member of the class **blank** in the program's current locale.

```
int iswblank(wint_t wc);
```

llabs()

In a similar manner to its counterparts *abs()* and *labs()*, this function computes the absolute value of an integer.

```
long long int llabs(long long int j);
```

lldiv()

In a similar manner to its counterparts *div()* and *ldiv()*, this function returns a structure of type **lldiv_t** which contains both the quotient and the remainder, each of which is of type **long long int**.

```
lldiv_t lldiv(long long int numer, long long int denom);
```

localeconv()

The standard added the following members to the **lconv** structure (defined in **<locale.h>**) to assign with long-standing POSIX practice and to permit additional flexibility with internationally formatted monetary quantities:

char p_cs_precedes	Set to 1 or 0 if the currency_symbol respectively precedes or succeeds the value for a non-negative locally formatted monetary quantity.
char n_cs_precedes	Set to 1 or 0 if the currency_symbol respectively precedes or succeeds the value for a negative locally formatted monetary quantity.
char p_sep_by_space	Set to a value indicating the separation of the currency_symbol , the sign string, and the value for a non-negative locally formatted monetary quantity.
char n_sep_by_space	Set to a value indicating the separation of the currency_symbol , the sign string, and the value for a negative locally formatted monetary quantity.
char p_sign_posn	Set to a value indicating the positioning of the positive_sign for a non-negative locally formatted monetary quantity.
char n_sign_posn	Set to a value indicating the positioning of the negative_sign for a negative locally formatted monetary quantity.

printf(), fprintf(), sprintf()

New length modifiers were added to the standard:

- hh Specifies that a following *d*, *i*, *o*, *u*, *x*, or *X* conversion specifier applies to a **signed char** or **unsigned char** argument; or that a following *n* conversion specifier applies to a pointer to a **signed char** argument. This modifier enables character types to be treated the same as all other integer types.
- ll Added to support the new **long long int** type. Specifies that a following *d*, *I*, *o*, *u*, *x*, or *X* conversion specifier applies to a **long long int** or **unsigned long long int** argument; or that a following *n* conversion specifier applies to a pointer to a **long long int** argument.

The maximum number of characters that can be produced by any single conversion was increased from 509 characters (C89) to 4 095 characters.

realloc()

The description of this function was changed to make it clear that the pointed-to object is deallocated, a new object is allocated, and the content of the new object is the same as that of the old object up to the lesser of the two sizes.

scanf(), fscanf(), sscanf()

The *hh* and *ll* length modifiers (see *printf()* above) were added.

Also the conversion modifiers *a* and *A* were added with *A* being equivalent to *a*.

These conversion modifiers match an optionally signed floating-point number, infinity, or NaN, whose format is the same as expected for the subject sequence of the *strtod()* function. The corresponding argument shall be a pointer to floating.

The behavior of the *sscanf()* function on encountering the end of a string has been clarified.

setvbuf()

The function prototype was changed to include the **restrict** type qualifier:

```
int setvbuf(FILE *restrict stream, char *restrict buf,
            int type, size_t size);
```

In previous revisions of the standard it was not clear about what, if anything, *size* means when *buf* is a null pointer. The standard now warns that *size* might not be ignored, so portable programs should supply a reasonable value.

snprintf()

This function was added to the standard to address the problem of *sprintf()* potentially overrunning an output buffer. It is equivalent in functionality to *sprintf()* except that it performs bounds checking on the output array. Extra characters are discarded and a null character is written at the end of the characters actually written to the array.

strftime()

The definition of this function was changed to incorporate additional conversion specifiers defined in IEEE Std 1003.1-1996 (POSIX.1), including %C, %D, %e, %F, %g, %G, %h, %n, %r, %R, %t, %T, %u, and %V, as well as the E and O modifiers.

strtod(), strtodf(), strtold()

The following two functions were added to the standard:

```
float strtodf(const char *restrict nptr,
             char **restrict endptr);
long double strtold (const char *restrict nptr,
                    char **restrict endptr);
```

In a similar manner to their counterpart, the *strtod()* function, these functions convert the initial portion of the string pointed to by *nptr* to **float** and **long double** representation, respectively. Support for subject sequences relating to floating-point (NaN, INF, and so on) was also added.

strtoll(), strtoull()

The following two functions were added to the standard:

```
long long int strtoll(const char *restrict nptr,
                    char **restrict endptr, int base);
unsigned long int strtoull(const char *restrict nptr,
                         char **restrict endptr, int base5);
```

In a similar manner to their counterparts, the *strtoll()* and *strtoull()* functions, these functions convert the initial portion of the string pointed to by *nptr* to **long long int** and **unsigned long int** representation, respectively.

tmpnam()

The previous standard had a serious flaw regarding this function. If the function were called fewer than {TMP_MAX} times but was unable to generate a suitable string because every potential string named an existing file, there was no way to report failure and no undefined behavior; hence there was no option other than to never return.

This standard resolved this issue by allowing the function to return a null pointer when it cannot generate a suitable string and by specifying that {TMP_MAX} is the number of potential strings, any or all of which may name existing files and thus not be suitable return values.

Note: This is a quiet change in the standard. Programs that call this function without checking for a null return value may produce undefined behavior.

ungetc()

The standard deprecated the use of this function on a binary stream at the beginning of the file.

vfscanf()

The following functions are functionally the same as *scanf()*, *fscanf()*, and *sscanf()* respectively, except that instead of being called with a variable number of arguments, they are called with an argument list:

```
int vscanf(const char *restrict format, va_list arg);
int vfscanf(FILE *restrict stream,
            const char *restrict format, va_list arg);
int vsscanf(const char *restrict s,
            const char *restrict format, va_list arg);
```

vfwscanf()

The following functions are functionally the same as *fwscanf()*, *swscanf()*, and *wscanf()* respectively, except that instead of being called with a variable number of arguments, they are called with an argument list:

```
int vfwscanf(FILE *restrict stream,
             const wchar_t *restrict format, va_list arg);
int vswscanf(const wchar_t *restrict s,
             const wchar_t *restrict format, va_list arg);
int vwscanf(const wchar_t *restrict format, va_list arg);
```

16.13.1 Wide-String Numeric Conversion Functions

The following functions were added to the existing wide-string numeric conversion functions:

```
float wcstof(const wchar_t *restrict nptr,
             wchar_t **restrict endptr);
long double wcstold(const wchar_t *restrict nptr,
                   wchar_t **restrict endptr);
long long int wcstoll(const wchar_t *restrict nptr,
                    wchar_t **restrict endptr, int base);
long long int wcstoull(const wchar_t *restrict nptr,
                     wchar_t **restrict endptr, int base);
```

16.14 Annexes

A number of new normative and informative annexes were added to the standard and some exiting annexes were modified.

Annexes A, B, and E were modified to include the new keywords, universal character names, types, implementation limits, macros and functions, and other changes to the C language.

Annex F (normative) was added to specify the IEC 60559:1989 standard floating-point arithmetic. An implementation that defines `__STDC_IEC_559__` must conform to the specification detailed in this annex.

Annex G (informative) was added to specify recommended IEC 60559:1989 standard-compatible complex arithmetic. An implementation that defines `__STDC_IEC_559_COMPLEX__` should conform to the specification detailed in this annex. It is non-normative because there were few existing implementations at the time this standard was approved.

Annex H (informative) describes the extent of support in this standard for language-independent arithmetic as specified in the ISO/IEC 10967-1:1994 standard. This annex was added, however, because all programming languages covered by ISO/IEC JTC1 SC22 standards are expected to review the ISO/IEC 10967-1:1994 standard and incorporate and further define the binding between that standard and each programming language.

Information about the ISO C Working Group (JTC1/SC22/WG14) can be found at:
wwwold.dkuug.dk/JTC1/SC22/WG14/.

CD-ROM Instructions

This chapter covers the contents and installation of the CD-ROM.

A.1 About the CD-ROM

The CD-ROM has a simple HTML interface. The easiest way to read the CD-ROM is by means of a Web browser. The top-level directory contains an HTML document, **index.htm** or **index.html**, that describes the full contents of the disk.

A.2 Contents

The CD-ROM contains the following:

- The Single UNIX Specification, Version 1
- The Single UNIX Specification, Version 2
- The Single UNIX Specification, Version 3 (2004 Edition including Technical Corrigendum 1 and Technical Corrigendum 2), which includes IEEE Std 1003.1-2001, IEEE Std 1003.1-2001/Cor 1-2002, and IEEE Std 1003.1-2001/Cor 2-2004.

For the Single UNIX Specification, Version 2 and the Single UNIX Specification, Version 3, the versions supplied are:

- An HTML translation of the specification which can be viewed directly off the CD-ROM using a Web browser, or alternately installed on your local machine, and viewed using a Web browser.
- PDF of the publications which may be viewed directly off the CD-ROM using a PDF viewer. Two PDF viewers are supplied on the CD-ROM.

For the Single UNIX Specification, Version 1, only the PDF version is supplied.

Additional specifications are also included on the CD-ROM; see the top-level **index.htm** or **index.html** file.

A.3 Viewing the Files

For the **.htm** and **.html** files, a Web browser is required.

For the **.pdf** files, a PDF reader is required. For more information on where to locate PDF reader software, visit www.opengroup.org/public/acrobat.html.

CD-ROM Terms and Conditions

1. Definitions

In these Terms and Conditions the following words have the meanings given:

CD-ROM	The CD-ROM with the functionality set out in the “Help Text” or user documentation;
Derivative Work	Any work which incorporates a substantial portion of any published material from any Work, whether or not authorized by this or any other License;
Internet	The world-wide publically-accessible network of inter-linked computers;
Intranet	A company-wide network of computers, accessible by company employees but protected from enquiry from Internet users and public dial-in access;
Licensee	The company/organization/individual to whom the CD-ROM is supplied;
Re-publication	The act of re-publishing, whether or not the new work is made public;
Re-publish	To reproduce or make available publicly or otherwise the text, design, or graphic material from any Work, whether or not in a form or on a medium different from that in or on which it was supplied by The Open Group;
Work	Any data on the CD-ROM;
The Open Group	The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, United Kingdom.

2. Right of Use

The Licensee shall have the right to use the Works contained on the CD-ROM for any purpose, subject to the limitations of these Terms and Conditions on Re-publication. Furthermore, the Licensee shall have the right:

- a. To store and have the CD-ROM stored in electronic form on one computer at a time, subject to the waiver set out in Clause 4;
- b. To display the contents of the CD-ROM on a computer screen; and
- c. For the Licensee’s internal use only, to produce paper copies of limited parts of any Work, as provided within the functionality of the software.

For the avoidance of doubt, nothing in these Terms and Conditions shall entitle the Licensee to use the Work in any way other than for the Licensee’s own internal business purposes, nor to disclose the Work to any third party.

Where any Work is subject to the copyright of any person other than The Open Group (“a joint publication”) the rights set out here are subject to any further conditions or limitations prescribed in respect of that Work (as set out therein) in order to protect the rights of that

other copyright owner. Where more than one CD-ROM is supplied under this Agreement, these rights shall apply separately in respect of each CD-ROM.

3. Restrictions in Liability

THE OPEN GROUP ACCEPTS LIABILITY UP TO A LIMIT EQUAL TO THE PRICE PAID FOR THE CD-ROM. THERE SHALL BE NO LIMIT TO ANY LEGAL LIABILITY FOR ANY DEATH OR PERSONAL INJURY.

4. Use of HTML and PDF Files

Files ending in the extension **.htm**, **.html**, and **.pdf** are intended for viewing by intranet browser and reader software. Implicit in this method of browsing and reading is the concept that the files may be viewed simultaneously by users on remote computers or consoles. It is permitted that the **.htm**, **.html**, and **.pdf** files on the CD-ROM be simultaneously accessed by two or more computers or consoles at any given time, provided that the users of those computers or consoles are employees of the Licensee. The Licensee shall not be permitted to distribute the CD-ROM **.htm**, **.html**, or **.pdf** files (or any other files on the CD-ROM) onto the Internet without prior written permission of The Open Group.

5. Re-publication

Save as set out above or as separately agreed, the Licensee shall have no right to Re-publish or distribute the CD-ROM or its contents in any way or on any medium whatever.

6. Termination

Either party may by written notice forthwith terminate this Agreement if the other party commits any material breach of these Terms and Conditions and fails to remedy it within a reasonable period after receiving written notice from the other party; the other party suffers distress or execution, commits an act of bankruptcy, goes or is put into liquidation (otherwise than solely for the purpose of solvent amalgamation or reconstruction), or seeks any form of protection against its creditors from any competent Court or Tribunal; or a receiver or administrative receiver is appointed over any substantial part of the other party's business.

Upon termination under this Clause or otherwise, the Licensee shall within one month deliver up any materials or equipment, including any special computer programs or electronic media supplied by The Open Group and shall certify that all copies (in electronic or other form) derived from the CD-ROM have been destroyed. Termination of this Agreement (however occurring) shall not affect any accrued rights or liabilities of either party nor shall it affect the coming into force or the continuance in force of any provision which is expressly or by implication intended to come into or continue in force on or after such termination.

7. Disputes

These Terms and Conditions shall be construed in accordance with the law of England and Wales. Any dispute arising under or in connection with these Terms and Conditions shall be referred to arbitration before a sole arbitrator to be appointed jointly or, in default of agreement, to be appointed by the President of the British Computer Society.

Terms and Conditions of Supply of the CD-ROM, June 2004.

Index

#line	432	<sys/ipc.h>	65
<aio.h>	65	<sys/mman.h>	65
<assert.h>	64	<sys/msg.h>	65
<cpio.h>	65	<sys/resource.h>	65
<ctype.h>	64	<sys/sem.h>	65
<dirent.h>	65	<sys/shm.h>	65
<errno.h>	64-65	<sys/stat.h>	65
<fcntl.h>	65	<sys/statvfs.h>	65
<float.h>	64	<sys/time.h>	65
<fmtmsg.h>	65	<sys/timeb.h>	65
<fnmatch.h>	65	<sys/times.h>	65
<ftw.h>	65	<sys/types.h>	65
<grp.h>	65	<sys/uid.h>	65
<iconv.h>	65	<sys/un.h>	65
<inttypes.h>	65	<sys/utsname.h>	65
<langinfo.h>	65	<sys/wait.h>	65
<libgen.h>	65	<syslog.h>	65
<limits.h>	64-65	<tar.h>	65
<locale.h>	64	<termios.h>	65
<math.h>	64	<time.h>	64-65
<monetary.h>	65	<ucontext.h>	65
<queue.h>	65	<ulimit.h>	65
<ndbm.h>	65	<unistd.h>	65
<nl_types.h>	65	<utime.h>	65
<poll.h>	65	<utmpx.h>	65
<pthread.h>	65	<varargs.h>	65
<pwd.h>	65	_Exit	156
<regex.h>	65	_exit	156
<regexp.h>	65	_Exit	450
<re_comp.h>	65	_longjmp	120
<sched.h>	65	_POSIX_ADVISORY_INFO	51
<search.h>	65	_POSIX_ASYNCHRONOUS_IO	50
<semaphore.h>	65	_POSIX_BARRIERS	51
<setjmp.h>	64-65	_POSIX_CHOWN_RESTRICTED	57
<signal.h>	64-65	_POSIX_CLOCK_SELECTION	51
<stdarg.h>	64	_POSIX_CPUTIME	51
<stddef.h>	64	_POSIX_C_SOURCE	25, 64
<stdio.h>	64-65	_POSIX_FSYNC	50
<stdlib.h>	64	_POSIX_JOB_CONTROL	57
<string.h>	64	_POSIX_MAPPED_FILES	50
<strings.h>	65	_POSIX_MEMLOCK	50
<stropts.h>	65	_POSIX_MEMLOCK_RANGE	50

_POSIX_MEMORY_PROTECTION.....	50
_POSIX_MESSAGE_PASSING.....	50
_POSIX_MONOTONIC_CLOCK.....	51
_POSIX_PRIORITIZED_IO.....	50
_POSIX_PRIORITY_SCHEDULING.....	50
.....	50
_POSIX_REALTIME_SIGNALS.....	50
_POSIX_SAVED_IDS.....	57
_POSIX_SEMAPHORES.....	50
_POSIX_SHARED_MEMORY_OBJECTS.....	50
_POSIX_SOURCE.....	25, 64
_POSIX_SPAWN.....	51
_POSIX_SPIN_LOCKS.....	51
_POSIX_SPORADIC_SERVER.....	51
_POSIX_SYNCHRONIZED_IO.....	50
_POSIX_THREAD_CPUTIME.....	51
_POSIX_THREAD_PRIORITY_SCHEDULING.....	50
_POSIX_THREAD_PRIO_INHERIT.....	50
_POSIX_THREAD_PRIO_PROTECT.....	50
_POSIX_THREAD_SPORADIC_SERVER.....	51
_POSIX_TIMEOUTS.....	51
_POSIX_TIMERS.....	50
_POSIX_TRACE.....	51
_POSIX_TRACE_EVENT_FILTER.....	51
_POSIX_TRACE_INHERIT.....	51
_POSIX_TRACE_LOG.....	51
_POSIX_TYPED_MEMORY_OBJECTS.....	51
_setjmp.....	120
_tolower.....	120
_toupper.....	120
_XOPEN_SOURCE.....	25, 29-30, 65
a64l.....	120
abort.....	121
abs.....	121
absolute-value function.....	440
accept.....	122
access.....	122
acos, acosf, acosl.....	122
acosh, acoshf, acoshl.....	123
adb, rationale for omission.....	59
addch.....	70
addchstr.....	70
addnstr.....	70
addnwstr.....	70
address information.....	172
add_wch.....	70
add_wchnstr.....	70
admin.....	55-56, 345
advanced realtime.....	50
ADVANCED REALTIME...140-141, 231-238, 247	
.....	258, 265, 290, 408
advanced realtime threads.....	51
ADVANCED REALTIME THREADS 255-256, 261	
.....	273-274
aio.h.....	392
aio_cancel.....	123
aio_error.....	123
aio_fsync.....	124
aio_read.....	124
aio_return.....	125
aio_suspend.....	125
aio_write.....	125
alarm.....	126
alias.....	56, 345
anonymous aggregate.....	428
ANSI C.....	3
ar.....	56, 345
argument macros.....	432
arithmetic.....	432
arpa/inet.h.....	392
array declaration.....	430
array type compatibility.....	430
as, rationale for omission.....	59
asa.....	56, 346
asctime, asctime_r.....	126
asin, asinf, asinl.....	127
asinh, asinhf, asinhl.....	127
Aspen Group.....	5
assert.....	127
assert.h.....	392
at.....	56, 346
atan, atanf, atanl.....	128
atan2, atan2f, atan2l.....	128
atanh, atanhf, atanhhl.....	128
atexit.....	129
atof.....	129
atoi.....	129
atol, atoll.....	129
atoll.....	450
attributes, clock-resolution.....	239
attributes, creation-time.....	239
attributes, generation-version.....	239
attributes, inheritance.....	240
attributes, log-full-policy.....	240
attributes, log-max-size.....	241
attributes, max-data-size.....	241
attributes, stream-full-policy.....	240
attributes, stream-min-size.....	241
attributes, trace-name.....	239-240
atoff.....	70
attr_get.....	70
attr_off.....	70

Index

attr_on.....	70	cd.....	56, 349
attr_set.....	70	ceil, ceilf, ceill.....	136
awk.....	24, 29, 56, 347	cexp, cexpf, cexpl.....	137
banner, rationale for omission.....	59	cfgetispeed.....	137
basename.....	56, 130, 347	cfgetospeed.....	137
batch.....	56, 347	cflow.....	55-56, 349
baudrate.....	70	cfsetispeed.....	137
bc.....	56, 347	cfsetospeed.....	138
bcmp.....	130	change history.....	38, 57, 66
bcopy.....	130	chdir.....	138
beep.....	70	chgat.....	70
bg.....	56, 348	chgrp.....	56, 350
bind.....	131	chmod.....	56, 138, 350
bkgd.....	70	chown.....	56, 139, 351
bkgrnd.....	70	chroot, rationale for omission.....	59
block.....	431	cimag, cimagf, cimagl.....	139
Boolean.....	426	cksum.....	56, 351
border.....	70	classification macro.....	441
border_set.....	70	clear.....	70
box.....	70	clearerr.....	140
box_set.....	70	clearok.....	70
bsd_signal.....	131	clock.....	140
bsearch.....	131	clock-resolution attribute.....	239
btowc.....	132	clock_getcpuclockid.....	140
bzero.....	132	clock_getres.....	140
C-language development tools.....	55	clock_gettime.....	140
C-language support.....	25	clock_nanosleep.....	141
c89.....	56	clock_settime.....	140
c99.....	25, 348	clog, clogf, clogl.....	141
cabs, cabsf, cabsl.....	132	closedir.....	142
cacos, cacosf, cacosl.....	132	closelog.....	142
cacosh, cacoshf, cacoshl.....	133	clrtoebot.....	70
cal.....	56, 349	clrtoeol.....	70
calendar, rationale for omission.....	59	cmp.....	56, 351
calloc.....	133	col, rationale for omission.....	60
cancel, rationale for omission.....	59	color_content.....	70
can_change_color.....	70	color_set.....	70
carg, cargf, cargl.....	133	comm.....	56, 351
case-sensitive identifiers.....	432	command.....	56, 352
casin, casinl, casinf, casinl.....	134	comment.....	431
casinh, casinhf, casinhl.....	134	Common API Specification.....	1, 3
cat.....	56, 349	common devices.....	22
catan, catanf, catanl.....	134	common directories.....	22
catanh, catanhf, catanhl.....	135	comparison macro.....	447
catclose.....	30, 135	complex number.....	437
catgets.....	30, 135	complex.h.....	393
catopen.....	30, 135	compound literal.....	428
cbreak.....	70	compress.....	56, 352
cbrt, cbrtf, cbrtl.....	136	compression utilities.....	55
ccos, ccosh, ccoshf, ccoshl.....	136	conformance statement.....	12
		confstr.....	143

conj, conjf, conjl.....	143
connect	143
copysign, copysignf, copysignl.....	144
copywin.....	70
cos, cosf, cosl.....	144
cosh, coshf, coshl.....	144
cp.....	56, 352
cpio, rationale for omission	60
cpio.h	393
cpow, cpowf, cpowl.....	145
cpp, rationale for omission.....	60
cproj, cprojf, cprojl.....	145
creal, crealf, creall.....	145
creat.....	146
creation-time attribute.....	239
crontab.....	56, 352
CRYPT.....	146
crypt.....	146
CRYPT.....	151, 294
csin, csinf, csinl.....	146
csinh, csinhf, csinhl.....	146
csplit.....	24, 56, 353
csqrt, csqrtf, csqrtl.....	147
ctags.....	24, 56, 353
ctan, ctanf, ctanl.....	147
ctanh, ctanhf, ctanhl.....	147
ctermid.....	147
ctime, ctime_r.....	148
ctype.h.....	393
cu, rationale for omission.....	60
curscr.....	70
curs_set.....	70
cur_term.....	70
cut.....	56, 353
cxref.....	55-56, 353
date.....	56, 354
daylight.....	326
DBM.....	148
dbm_clearerr.....	148
dbm_close.....	148
dbm_delete.....	148
dbm_error.....	148
dbm_fetch.....	148
dbm_firstkey.....	148
dbm_nextkey.....	148
dbm_open.....	148
dbm_store.....	148
dc, rationale for omission.....	60
dd.....	56, 354
decimal integer constant.....	428
def_prog_mode.....	70
delay_output.....	70
delch.....	70
deleteln.....	70
delscreen.....	70
delta.....	55-56, 354
delwin.....	70
del_curterm.....	70
derwin.....	70
designated initializer.....	428
DEVELOPMENT.....	55-56, 99, 345, 349353-354, 361, 364, 366-367, 370373, 378, 381, 386-387, 389-390
df.....	56, 355
diff.....	56, 355
difftime.....	148
DIR.....	142, 230
dircmp, rationale for omission.....	60
dirent.h.....	394
dirname.....	56, 149, 355
dis.....	56
dis, rationale for omission.....	60
div.....	149
dlclose.....	149
dlderror.....	149
dlfcn.h.....	394
dlopen.....	150
dlsym.....	150
douppdate.....	70
drand48.....	150
du.....	56, 356
dup, dup2.....	151
dupwin.....	70
echo.....	56, 70, 356
echochar.....	70
echo_wchar.....	70
ecvt.....	151
ed.....	24, 56, 356
egrep.....	24
emacs, rationale for omission.....	60
empty argument macros.....	432
encrypt.....	151
encryption.....	50
endgrent.....	152
endhostent.....	152
endnetent.....	152
endprotoent.....	152
endpwent.....	153
endservent.....	153
endutxent.....	153
endwin.....	70
ENHANCED CURSES.....	30, 69

Index

enumeration specifier	429	fenv.h	395
env	56, 357	feof	162
environ	155	feraiseexcept	162
environment	449	ferror	163
environment variables	23	fesetenv	161
erand48	150	fesetexceptflag	161
erase	70	fesetround	162
erasechar	70	fetestexcept	163
erf, erff, erfl	154	feupdateenv	163
erfc, erfcf, erfcl	154	FE_ALL_EXCEPT	448
errno	154, 431	FE_DIVBYZERO	448
errno.h	394	FE_DOWNWARD	448
error descriptions	180	FE_INEXACT	448
error numbers	26	FE_INVALID	448
ex	24, 56, 357	FE_OVERFLOW	448
exact-width integer type	435	FE_TONEAREST	448
exception	448	FE_TOWARDZERO	448
execl, execl, execlp	155	FE_UNDERFLOW	448
execv, execve, execvp	155	FE_UPWARD	448
exit	156	fflush	163
exp, expf, expl	156	ffs	164
exp2, exp2f, exp2l	157	fg	56, 358
expand	56, 357	fgetc	164
expm1, expm1f, expm1l	157	fgetpos	164
exponential function	440, 443	fgets	165
expr	24, 56, 358	fgetwc	165
external variable		fgetws	165
quick reference	73	FIFO	218
fabs, fabsf, fabsl	157	file	56
false	56, 358	FILE	140, 159
fastest minimum-width integer type	435	file	359
fattach	158	file access	24
fc	56, 358	file format notation	28
fchdir	158	fileno	165
fchmod	158	filter	70
fchown	158	find	56, 359
fclose	159	flash	70
fcntl	159	flexible array member	430
fcntl.h	394	float.h	396
fcvt	151	floating-point constant	432
fdatasync	160	floating-point environment	447
fdetach	160	flockfile	166
fdim, fdimf, fdiml	160	floor, floorf, floorl	166
fdopen	160	flushinp	70
feature group	50	fma, fmaf, fmal	166
feature test macros	25, 64	fmax, fmaxf, fmaxl	167
feclearexcept	161	fmin, fminf, fminl	167
fegetenv	161	fmod, fmodf, fmodl	167
fegetexceptflag	161	fmsg	168
fegetround	162	fmsg.h	396
feholdexcept	162	fnmatch	168

fnmatch.h	396
fold	56, 360
fopen	168
for statement	431
fork	169
fort77	56, 360
FORTTRAN	55-56, 99, 353, 360
fpathconf	169
fpclassify	170, 441
fpos_t	450
fprintf	170, 452
fputc	171
fputs	171
fputwc	171
fputws	172
FP_INFINITE	441
FP_NAN	441
FP_NORMAL	441
FP_SUBNORMAL	441
FP_ZERO	441
fread	172
free	172
freeaddrinfo	172
freopen	173
frexp, frexpf, frexpl	174
fscanf	174, 452
fseek, fseeko	174
fsetpos	175
fstat	175
fstatvfs	176
fsync	176
ftell, ftello	176
ftime	177
ftok	177
truncate	177
trylockfile	166
ftw	178
FTW	228
ftw.h	396
function	
absolute-value	440
exponential	440, 443
hyperbolic	439, 442
logarithmic	440, 443
manipulation	440, 446
nearest integer	445
power	440
quick reference	73
remainder	446
trigonometric	438, 442
unsuffixed	450
wide-string numeric conversion	454
funlockfile	166
fuser	55, 360
fwide	178
fwprintf	178
fwrite	179
fwscanf	179
gai_strerror	180
gcvt	151
gencat	30, 56, 360
general terminal interface	27
generation-version attribute	239
generic parameter	449
get	55-56, 361
getaddrinfo	172
getbegyx	70
getbkgd	70
getbkgrnd	70
getc	180
getcchar	70
getch	70
getchar	180
getchar_unlocked	180
getconf	56, 361
getcontext	181
getcwd	181
getc_unlocked	180
getdate	181
getegid	181
getenv	182
geteuid	182
getgid	182
getgrent	152
getgrgid	182
getgrgid_r	182
getgrnam	183
getgrnam_r	183
getgroups	183
gethostbyaddr	184
gethostbyname	184
gethostent	152
gethostid	184
gethostname	184
getitimer	185
getlogin	185
getlogin_r	185
getmaxyx	70
getmsg	185
getnameinfo	186
getnetbyaddr	152
getnetbyname	152

Index

getnetent	152	grep	24, 56, 362
getnstr	70	grep -E	24
getn_wstr	70	grp.h	397
getopt	186	halfdelay	70
getopts	56, 362	hash	56, 362
getparyx	70	has_colors	70
getpeername	186	has_ic	70
getpgid	187	hcreate	193
getpgrp	187	hdestroy	193
getpid	187	head	56, 362
getpmsg	185	header	434
getppid	187	quick reference	105
getpriority	187	header rules	63
getprotobyname	152	headers	391
getprotobynumber	152	hexadecimal constant	432
getprotoent	152	hline	70
getpwent	153	hline_set	70
getpwnam	188	hsearch	193
getpwnam_r	188	htonl	193
getpwuid	188	htons	193
getpwuid_r	188	hyperbolic function	439, 442
getrlimit	189	hypot	194
getrusage	189	hypotf	194
gets	190	hypotl	194
getservbyname	153	h_errno	193
getservbyport	153	l18N	30
getservent	153	iconv	30, 56, 194, 362
getsid	190	iconv.h	397
getsockname	190	iconv_close	30, 194
getsockopt	190	iconv_open	30, 195
getstr	70	id	56, 363
getsubopt	190	idcok	70
gettimeofday	191	identifier	432
getuid	191	IEEE Std 1003.1	8
getutxent	153	if_freenameindex	195
getutxid	153	if_indextoname	195
getutxline	153	if_nameindex	195
getwc	191	if_nametoindex	195
getwchar	191	ilogb	196
getwd	192	ilogbf	196
getwin	70	ilogbl	196
getyx	70	imaxabs	196
get_wch	70	imaxdiv	196
get_wstr	70	immedok	70
glob	192	implicit declaration	429
glob.h	396	inch	70
globfree	192	inchnstr	70
gmtime	192	incomplete array	430
gmtime_r	192	index	196
grantpt	193	inet_addr	197
greatest-width integer type	435	inet_ntoa	197

inet_ntop	197	isless	200
inet_pton	197	islessequal	201
inheritance attribute	240	islessgreater	201
initscr	70	islower	201
initstate	197	isnam	441
init_color	70	isnan	201
inline	427	isnormal	201, 441
innstr	70	ISOC standard	3
innwstr	70	ISO C headers	63
insch	70	ISO C standard	425
insdelln	70	ISO/IEC 9945	8
insertln	70	iso646.h	397
insnstr	70	isprint	201
insque	197	ispunct	202
insstr	70	isspace	202
instr	70	isunordered	202
ins_nwstr	70	isupper	202
ins_wch	70	iswalnum	202
ins_wstr	70	iswalpha	203
integer type	434	iswblank	203, 451
exact-width	435	iswcntrl	203
fastest minimum-width	435	iswctype	203
greatest-width	435	iswdigit	203
holding object pointer	435	iswgraph	204
minimum-width	435	iswlower	204
specified-width limit	437	iswprint	204
interface count	24	iswpunct	204
internationalization	30	iswspace	204
interprocess communication	26	iswupper	205
intrflush	70	iswxdigit	205
inttypes.h	397	isxdigit	205
inwstr	70	is_linetouched	70
in_wch	70	j0	205
in_wchnstr	70	j1	205
ioctl	198	jn	205
ipcrm	55, 363	jobs	56, 363
ipcs	55, 363	join	56, 364
isalnum	198	jrandon48	150
isalpha	198	keyname	70
isascii	198	keypad	70
isastream	199	keyword	425
isatty	199	kill	56, 205, 364
isblank	199, 451	killchar	70
iscntrl	199	killpg	206
isdigit	199	l64a	120
isendwin	70	labs	206
isfinite	200, 441	langinfo.h	398
isgraph	200	lchown	206
isgreater	200, 447	lcong48	150
isgreaterequal	200, 447	ld, rationale for omission	60
isinf	200, 441	ldexp	207

Index

ldexpf	207	logbl	212
ldexpl	207	logf	210
ldiv	207	logger	56, 365
leaveok	70	login, rationale for omission	60
legacy	52	logl	210
lex	24, 56, 364	logname	56, 365
lfind	213	longjmp	212
lgamma	207	longname	70
lgammaf	207	lorder, rationale for omission	61
lgammal	207	lp	56, 366
libgen.h	398	lpstat, rationale for omission	61
library	450	lrand48	150
limits.h	398	lrint	213
line length	432	lrintf	213
line, rationale for omission	60	lrintl	213
link	55, 208, 364	lround	213
lint	55-56	lroundf	213
lint, rationale for omission	60	lroundl	213
lio_listio	208	ls	56, 366
listen	209	lsearch	213
llabs	206, 451	lseek	213
lldiv	207, 451	lstat	24, 214
llrint	209	m4	55-56, 366
llrintf	209	macro	437
llrintl	209	comparison	447
llround	209	quick reference	73
llroundf	209	mail, rationale for omission	61
llroundl	209	mailx	29, 56, 367
ln	56, 365	make	56, 367
locale	30, 56, 365	makecontext	214
locale.h	399	malloc	214
localeconv	209, 451	man	56, 368
localedef	30, 56, 365	manipulation function	440, 446
localtime	210	math.h	400
localtime_r	210	max-data-size attribute	241
lockf	210	mblen	215
log	210	mbrlen	215
log-full-policy attribute	240	mbrtowc	215
log-max-size attribute	241	mbsinit	215
log10	211	mbsrtowcs	216
log10f	211	mbstowcs	216
log10l	211	mbtowc	216
log1p	211	memccpy	216
log1pf	211	memchr	217
log1pl	211	memcmp	217
log2	212	memcpy	217
log2f	212	memmove	217
log2l	212	memset	217
logarithmic function	440, 443	mesg	56
logb	212	meta	70
logbf	212	min/max line length	432

minimum-width integer type	435	nanl	227
mkdir	56, 218, 368	nanosleep	227
mkfifo	56, 218, 368	napms	70
mknod	218	ndbm.h	401
mknod, rationale for omission	61	nearbyint	227
mkstemp	219	nearbyintf	227
mktemp	219	nearbyintl	227
mktime	219	nearest integer function	445
mlock	219	net/if.h	402
mlockall	220	netdb.h	402
mmap	220	netinet/in.h	402
modf	221	netinet/tcp.h	403
modff	221	networking services	29
modfl	221	Networking Services, Issue 5	5
monetary.h	401	newgrp	56, 369
more	24, 56, 369	newpad	70
move	70	news, rationale for omission	61
mprotect	221	newterm	70
mqqueue.h	401	newwin	70
mq_close	222	nextafter	228
mq_getattr	222	nextafterf	228
mq_notify	222	nextafterl	228
mq_open	223	nexttoward	228
mq_receive	223	nexttowardf	228
mq_send	224	nexttowardl	228
mq_setattr	224	nftw	228
mq_timedreceive	223	NGROUPS_MAX	369
mq_timedsend	224	nice	56, 229, 370
mq_unlink	225	nl	24, 56, 70, 370
rand48	150	nl_langinfo	229
msg	368	nl_types.h	403
msgctl	225	nm	56, 370
msgget	225	nodelay	70
msgrcv	225	nohup	56, 370
msgsnd	226	noqiflush	70
msync	226	notimeout	70
multiplicative operator	429	rand48	150
munlock	219	ntohl	193
munlockall	220	ntohs	193
munmap	226	od	56, 370
mv	56, 369	open	229
mvcur	70	Open Brand	11
mvderwin	70	opendir	230
mvprintw	70	openlog	142
mvscanw	70	optarg	186
mvwin	70	opterr	186
name information	186	optind	186
name space	25	option group	50
name space rules	63	option groups	22
nan	227	options policy	52
nanf	227	optopt	186

Index

OSF AES	3	posix_spawn_file_actions_init	235
overlay	70	posix_trace_attr_destroy	239
pack, rationale for omission	61	posix_trace_attr_getclockres	239
pair_content	70	posix_trace_attr_getcreatetime	239
passwd, rationale for omission	61	posix_trace_attr_getgenversion	239
paste	56, 371	posix_trace_attr_getinherited	240
patch	56, 371	posix_trace_attr_getlogfullpolicy	240
pathchk	56, 371	posix_trace_attr_getlogsize	240
pathconf	169	posix_trace_attr_getmaxdatasize	240
pause	230	posix_trace_attr_getmaxsystemeventsz	240
pax	24, 56, 371	posix_trace_attr_getmaxusereventsiz	240
pcat	56	posix_trace_attr_getname	239
pcat, rationale for omission	61	posix_trace_attr_getstreamfullpolicy	240
pclose	230	posix_trace_attr_getstreamsize	240
pechochar	70	posix_trace_attr_init	239
perror	230	posix_trace_attr_setinherited	240
pg	24	posix_trace_attr_setlogfullpolicy	240
pg, rationale for omission	61	posix_trace_attr_setlogsize	240
pipe	231	posix_trace_attr_setmaxdatasize	240
pnoutrefresh	70	posix_trace_attr_setname	239
poll	231	posix_trace_attr_setstreamfullpolicy	240
poll.h	403	posix_trace_attr_setstreamsize	240
popen	231	posix_trace_clear	242
portability codes	16	posix_trace_close	242
POSIX.1	3	posix_trace_create	242
posix_fadvise	231	posix_trace_create_withlog	242
posix_fallocate	232	posix_trace_event	243
posix_madvise	232	posix_trace_eventid_equal	244
posix_memalign	233	posix_trace_eventid_get_name	244
posix_mem_offset	233	posix_trace_eventid_open	243
posix_openpt	233	posix_trace_eventset_add	244
posix_spawn	234	posix_trace_eventset_del	244
posix_spawnattr_destroy	236	posix_trace_eventset_empty	244
posix_spawnattr_getflags	236	posix_trace_eventset_fill	244
posix_spawnattr_getpgroup	237	posix_trace_eventset_ismember	244
posix_spawnattr_getschedparam	237	posix_trace_eventtypelist_getnext_id	245
posix_spawnattr_getschedpolicy	237	posix_trace_eventtypelist_rewind	245
posix_spawnattr_getsigdefault	238	posix_trace_flush	242
posix_spawnattr_getsigmask	238	posix_trace_getnext_event	246
posix_spawnattr_init	236	posix_trace_get_attr	245
posix_spawnattr_setflags	236	posix_trace_get_filter	246
posix_spawnattr_setpgroup	237	posix_trace_get_status	245
posix_spawnattr_setschedparam	237	posix_trace_open	242
posix_spawnattr_setschedpolicy	237	posix_trace_rewind	242
posix_spawnattr_setsigdefault	238	posix_trace_set_filter	246
posix_spawnattr_setsigmask	238	posix_trace_shutdown	242
posix_spawnnp	234	posix_trace_start	247
posix_spawn_file_actions_addclose	235	posix_trace_stop	247
posix_spawn_file_actions_adddup2	235	posix_trace_timedgetnext_event	246
posix_spawn_file_actions_addopen	235	posix_trace_trid_eventid_open	244
posix_spawn_file_actions_destroy	235	posix_trace_trygetnext_event	246

posix_typed_mem_get_info	247	pthread_cleanup_pop	256
posix_typed_mem_open	247	pthread_cleanup_push	256
pow	248	pthread_condattr_destroy	258
power function	440	pthread_condattr_getclock	258
powf	248	pthread_condattr_getpshared	259
powl	248	pthread_condattr_init	258
pr	56, 373	pthread_condattr_setclock	258
pragma	433	pthread_condattr_setpshared	259
pread	277	pthread_cond_broadcast	257
predefined identifier	427	pthread_cond_destroy	257
predefined macro	432	pthread_cond_init	257
printf	56, 170, 373, 452	pthread_cond_signal	257
printw	70	pthread_cond_timedwait	257
process		pthread_cond_wait	257
setting real and effective user IDs	295	pthread_create	259
product standard	12	pthread_detach	260
prof, rationale for omission	61	pthread_equal	260
programming environment	24	pthread_exit	260
prs	55-56, 373	pthread_getconcurrency	261
ps	56, 374	pthread_getcpuclockid	261
pselect	248	pthread_getschedparam	261
pthread.h	403	pthread_getspecific	262
pthread_atfork	249	pthread_join	262
pthread_attr_destroy	249	pthread_join_unlock	274
pthread_attr_getdetachstate	250	pthread_key_create	263
pthread_attr_getguardsize	250	pthread_key_delete	263
pthread_attr_getinheritsched	251	pthread_kill	263
pthread_attr_getschedparam	251	pthread_mutexattr_destroy	265
pthread_attr_getschedpolicy	252	pthread_mutexattr_getprioceiling	266
pthread_attr_getscope	252	pthread_mutexattr_getprotocol	266
pthread_attr_getstack	253	pthread_mutexattr_getpshared	267
pthread_attr_getstackaddr	253	pthread_mutexattr_gettype	267
pthread_attr_getstacksize	254	pthread_mutexattr_init	265
pthread_attr_init	249	pthread_mutexattr_setprioceiling	266
pthread_attr_setdetachstate	250	pthread_mutexattr_setprotocol	266
pthread_attr_setguardsize	250	pthread_mutexattr_setpshared	267
pthread_attr_setinheritsched	251	pthread_mutexattr_settype	267
pthread_attr_setschedparam	251	pthread_mutex_destroy	263
pthread_attr_setschedpolicy	252	pthread_mutex_getprioceiling	264
pthread_attr_setscope	252	pthread_mutex_init	263
pthread_attr_setstack	253	pthread_mutex_lock	264
pthread_attr_setstackaddr	253	pthread_mutex_setprioceiling	264
pthread_attr_setstacksize	254	pthread_mutex_timedlock	265
pthread_barrierattr_destroy	255	pthread_mutex_trylock	264
pthread_barrierattr_getpshared	256	pthread_mutex_unlock	264
pthread_barrierattr_init	255	pthread_once	268
pthread_barrierattr_setpshared	256	pthread_rwlockattr_destroy	271
pthread_barrier_destroy	255	pthread_rwlockattr_getpshared	271
pthread_barrier_init	255	pthread_rwlockattr_init	271
pthread_barrier_wait	255	pthread_rwlockattr_setpshared	271
pthread_cancel	256	pthread_rwlock_destroy	268

Index

pthread_rwlock_init.....	268	raise.....	277	
pthread_rwlock_rdlock.....	269	rand.....	277	
pthread_rwlock_timedrdlock.....	269	random.....	197	
pthread_rwlock_timedwrlock.....	270	rand_r.....	277	
pthread_rwlock_tryrdlock.....	269	raw.....	70	
pthread_rwlock_trywrlock.....	270	RCS, rationale for omission.....	61	
pthread_rwlock_unlock.....	271	read.....	56, 277, 377	
pthread_rwlock_wrlock.....	270	readdir.....	278	
pthread_self.....	272	readdir_r.....	278	
pthread_setcancelsta.....	272	readlink.....	24, 279	
pthread_setcanceltype.....	272	readv.....	279	
pthread_setconcurrency.....	261	realloc.....	280, 452	
pthread_setschedparam.....	261	realpath.....	24, 280	
pthread_setschedprio.....	272	realtime.....	27, 50	
pthread_setspecific.....	262	REALTIME.....	123-125, 140, 160, 208219-220, 222-225, 227, 285-291297, 302-303, 322-323, 392, 401406-407	
pthread_sigmask.....	273	realtime threads.....	50	
pthread_spin_destroy.....	273	REALTIME THREADS.....	251-252, 261264, 266, 272	
pthread_spin_init.....	273	recv.....	280	
pthread_spin_lock.....	274	recvfrom.....	280	
pthread_spin_trylock.....	274	recvmsg.....	281	
pthread_testcancel.....	272	red, rationale for omission.....	61	
ptsname.....	275	redrawwin.....	70	
putc.....	275	refresh.....	70	
putchar.....	275	regcomp.....	281	
putchar_unlocked.....	180	regerror.....	281	
putc_unlocked.....	180	regex.h.....	406	
putenv.....	275	regexec.....	281	
putmsg.....	275	regfree.....	281	
putp.....	70	regular expressions.....	24	
putpmsg.....	275	rejected utilities.....	59	
puts.....	276	remainder.....	281	
pututxline.....	153	remainder function.....	446	
putwc.....	276	remainderf.....	281	
putwchar.....	276	remainderl.....	281	
putwin.....	70	remove.....	282	
pwd.....	56, 374	remque.....	197	
pwd.h.....	406	remquo.....	282	
pwrite.....	341	remquof.....	282	
qalter.....	56, 374	remquol.....	282	
qdel.....	56, 374	rename.....	282	
qhold.....	56, 375	renice.....	56, 377	
qiflush.....	70	reserved name.....	66 avoiding.....	65
qmove.....	56, 375	resetty.....	70	
qmsg.....	56, 375	reset_prog_mode.....	70	
qrerun.....	56, 375	restartterm.....	70	
qrls.....	56, 376	rewind.....	283	
qselect.....	56, 376			
qsig.....	56, 376			
qsort.....	276			
qstat.....	56, 376			
qsub.....	56, 377			

rewinddir.....	283	sem_close.....	288
rindex.....	283	sem_destroy.....	288
rint.....	283	sem_getvalue.....	289
rintf.....	283	sem_init.....	289
rintl.....	283	sem_open.....	289
ripoffline.....	70	sem_post.....	290
rm.....	56, 378	sem_timedwait.....	290
rmdel.....	55-56, 378	sem_trywait.....	291
rmdir.....	56, 284, 378	sem_unlink.....	291
round.....	284	sem_wait.....	291
roundf.....	284	send.....	292
rounding.....	448	sendmsg.....	292
roundl.....	284	sendto.....	292
rsh, rationale for omission.....	61	setbuf.....	293
sact.....	55-56, 378	setcchar.....	70
savetty.....	70	setcontext.....	181
SA_RESETHAND.....	131	setegid.....	293
SA_RESTART.....	131	setenv.....	293
scalb.....	284	seteuid.....	293
scalbln.....	285	setgid.....	294
scalblnf.....	285	setgrent.....	152
scalblnl.....	285	sethostent.....	152
scalbn.....	285	setitimer.....	185
scalbnf.....	285	setjmp.....	294
scalbnl.....	285	setjmp.h.....	407
scanf.....	174, 452	setkey.....	294
scanw.....	70	setlocale.....	294
sccs.....	55-56, 378	setlogmask.....	142
sched.h.....	406	setnetent.....	152
sched_getparam.....	285	setpgid.....	295
sched_getscheduler.....	286	setpgrp.....	295
sched_get_priority_max.....	285	setpriority.....	187
sched_get_priority_min.....	285	setprotoent.....	152
sched_rr_get_interval.....	286	setpwent.....	153
sched_setparam.....	286	setregid.....	295
sched_setscheduler.....	287	setreuid.....	295
sched_yield.....	287	setrlimit.....	189
scr1.....	70	setscreg.....	70
scrollok.....	70	setservent.....	153
scr_dump.....	70	setsid.....	296
sdb, rationale for omission.....	61	setsockopt.....	296
sdiff, rationale for omission.....	62	setstate.....	197
search.h.....	407	setuid.....	296
sed.....	24, 56, 379	setupterm.....	70
seed48.....	150	setutxent.....	153
seekdir.....	288	setvbuf.....	296, 452
select.....	248	set_curterm.....	70
semaphore.h.....	407	set_term.....	70
semctl.....	291	sh.....	56, 379
semget.....	292	shar, rationale for omission.....	62
semop.....	292	shell command language.....	28

Index

shl, rationale for omission	62	slk_color	70
shmat	297	slk_init	70
shmctl	298	slk_label	70
shmdt	298	slk_nooutrefresh	70
shmget	298	slk_refresh	70
shm_open	297	slk_restore	70
shm_unlink	297	slk_set	70
shutdown	298	slk_touch	70
sigaction	298	slk_wset	70
sigaddset	299	snprintf	170, 452
sigaltstack	299	socketatmark	305
sigdelset	299	socket	305
sigemptyset	300	socketpair	306
sigfillset	300	sort	56, 380
sighold	300	source file inclusion	432
sigignore	300	spawn.h	408
siginterrupt	301	Spec 1170	1
siglongjmp	301	specified-width integer type	437
signal	301	spell, rationale for omission	62
signal.h	408	split	56, 380
signbit	302, 442	sprintf	170, 452
sigpause	300	sqrt	306
sigpending	302	sqrtf	306
sigprocmask	273	sqrtd	306
sigqueue	302	srand	277
sigrelse	300	rand48	150
sigset	300	random	197
sigsetjmp	302	sscanf	174, 452
sigsuspend	303	standard I/O streams	26
sigtimedwait	303	standend	70
sigwait	304	start_color	70
sigwaitinfo	303	stat	306
sin	304	statvfs	176
sinf	304	stdarg.h	409
Single UNIX Environment	15	stdbool.h	410
Single UNIX Specification, Version 3	13	stddef.h	410
sinh	304	stderr	307
sinhf	304	stdint.h	410
sinhl	304	stdio.h	410
sinl	304	stdlib.h	411
SIOCATMARK	305	stdout	307
size, rationale for omission	62	stdscr	70
sizeof	429	strcasecmp	307
sleep	56, 305, 380	strcat	307
slk_attroff	70	strchr	307
slk_attron	70	strcmp	307
slk_attrset	70	strcoll	308
slk_attr_off	70	strcpy	308
slk_attr_on	70	strcspn	308
slk_attr_set	70	strdup	308
slk_clear	70	STREAM	275

stream-full-policy attribute	240
stream-min-size attribute	241
STREAMS	26, 160, 185, 198, 412
strerror	309
strerror_r	309
strfmon	309
strftime	309, 453
string literal	429
string.h	411
strings	56, 380
strings.h	412
strip	56, 381
strlen	310
strncasecmp	307
strncat	310
strncmp	310
strncpy	310
stropts.h	412
strpbrk	311
strptime	311
strchr	311
strspn	311
strstr	312
strtod	312, 453
strtof	312, 453
strtoimax	312
strtok	313
strtok_r	313
strtol	313
strtold	312, 453
strtoll	313, 453
strtol	313
strtoull	313, 453
strtoumax	312
structure member	
incomplete array	430
strxfrm	314
stty	56, 381
su, rationale for omission	62
subpad	70
subwin	70
sum, rationale for omission	62
superuser	59
SVID	3
swab	314
swapcontext	214
swprintf	178
swscanf	179
symbolic links	28
symlink	24, 315
sync	315
syncok	70
sys/ipc.h	412
sys/mman.h	412
sys/msg.h	413
sys/resource.h	413
sys/select.h	414
sys/sem.h	414
sys/shm.h	414
sys/socket.h	414
sys/stat.h	415
sys/statvfs.h	415
sys/time.h	416
sys/timeb.h	416
sys/times.h	416
sys/types.h	416
sys/uio.h	417
sys/un.h	417
sys/utsname.h	417
sys/wait.h	417
sysconf	315
syslog	142, 417
system	317
system interfaces	119
tabs	56, 381
tail	56, 381
talk	56, 382
tan	317
tanf	317
tanh	317
tanhf	317
tanhl	317
tanl	317
tar, rationale for omission	62
tar.h	418
tcdrain	318
tcflow	318
tcflush	318
tcgetattr	319
tcgetpgrp	319
tcgetsid	319
tcsendbreak	319
tcsetattr	320
tcsetpgrp	320
tdelete	320
tee	56, 382
telldir	321
tempnam	321
termattrs	70
terminal interfaces	30, 69
termios.h	418
termname	70

Index

term_attrs.....	70	tty.....	56, 384
test.....	56, 382	ttyname.....	326
testing.....	13	ttyname_r.....	326
tfind.....	320	twalk.....	320
tgamma.....	321	type.....	56, 384, 426
tgammaf.....	321	type qualifier.....	426
tgamma1.....	321	type-generic math.....	449
tgetent.....	70	typeahead.....	70
tgmth.h.....	418	tzname.....	326
threads.....	27	tzset.....	326
tigetflag.....	70	ualarm.....	326
time.....	56, 321, 382	UCN.....	427
time.h.....	419	ucontext.h.....	420
timeout.....	70	UINtN_MAX.....	437
timer_create.....	322	ulimit.....	56, 327, 384
timer_delete.....	322	ulimit.h.....	420
timer_getoverrun.....	323	umask.....	56, 327, 384
timer_gettime.....	323	unalias.....	56, 385
timer_settime.....	323	uname.....	56, 327, 385
times.....	323	uncompress.....	56, 385
timezone.....	326	unctrl.....	70
tmpfile.....	323	unexpand.....	56, 385
tmpnam.....	324, 453	ungetc.....	55-56, 386
TMP_MAX.....	453	ungetc.....	327, 453
toascii.....	324	ungetch.....	70
token pasting.....	433	ungetwc.....	327
tolower.....	324	uniq.....	56, 386
touch.....	56, 383	unistd.h.....	420
touchline.....	70	universal character name.....	427
toupper.....	324	unlink.....	55, 328, 386
towctrans.....	325	unlockpt.....	328
towlower.....	325	unpack, rationale for omission.....	62
toupper.....	325	unsetenv.....	328
tparam.....	70	unsuffixed function	
tput.....	56, 383	with C-prefixed counterpart.....	450
tputs.....	70	without C-prefixed counterpart.....	450
tr.....	56, 383	untouchwin.....	70
trace-name attribute.....	239-240	user ID	
trace.h.....	419	real and effective.....	295
tracing.....	51	setting real and effective.....	295
TRACING.....	239-240, 242-247	use_env.....	70
translation limit.....	433	usleep.....	328
translation-time arithmetic.....	432	utilities.....	344
trigonometric function.....	438, 442	utility	
true.....	56, 383	quick reference.....	99
trunc.....	325	utime.....	329
truncate.....	325	utime.h.....	423
truncf.....	325	utimes.....	329
truncl.....	325	utmpx.h.....	423
tsearch.....	320	uucp.....	55-56, 386
tsort.....	56, 384	uudecode.....	56, 386

uuencode.....	56, 387	wscmp.....	333
uupack.....	56	wscoll.....	333
uustat.....	55-56, 387	wscspy.....	333
uuto.....	55	wscspn.....	333
uux.....	55-56, 387	wcsftime.....	333
val.....	55-56, 387	wcslen.....	334
variable length array.....	430	wcsncat.....	334
variadic macro.....	433	wcsncmp.....	334
va_arg.....	329	wcsncpy.....	334
va_copy.....	329, 434	wcspbrk.....	334
va_end.....	329	wcsrchr.....	335
va_start.....	329	wcsrtombs.....	335
vfork.....	330	wcsspn.....	335
vfprintf.....	330	wcsstr.....	335
VFS.....	415	wcstod.....	336
vfscanf.....	330, 454	wcstof.....	336
vwprintf.....	331	wcstoimax.....	336
vwscanf.....	331, 454	wcstok.....	336
vi.....	24, 56, 388	wcstol.....	337
vidattr.....	70	wcstold.....	336
vid_attr.....	70	wcstoll.....	337
vid_puts.....	70	wcstombs.....	337
vline.....	70	wcstoul.....	337
vline_set.....	70	wcstoull.....	337
vprintf.....	330	wcstoumax.....	336
vscanf.....	330	wcswcs.....	338
vsnprintf.....	330	wcswidth.....	338
vsprintf.....	330	wcsxfrm.....	338
vsscanf.....	330	wctob.....	339
vswprintf.....	331	wctomb.....	339
vswscanf.....	331	wctrans.....	339
vwprintf.....	331	wctype.....	339
vwprintw.....	70	wctype.h.....	423
vwscanf.....	331	wcwidth.....	339
vwscanw.....	70	what.....	55-56, 389
vw_printw.....	70	who.....	56, 389
vw_scanw.....	70	wide-string numeric conversion function.....	454
wait.....	56, 331, 388	wmemchr.....	340
waitid.....	332	wmemcmp.....	340
waitpid.....	331	wmemcpy.....	340
wall, rationale for omission.....	62	wmemmove.....	340
wattr_get.....	70	wmemset.....	341
wattr_off.....	70	wordexp.....	341
wattr_on.....	70	wordexp.h.....	424
wattr_set.....	70	wordfree.....	341
wc.....	56, 388	wprintf.....	178
wchar.h.....	423	write.....	56, 341, 389
wcolor_set.....	70	writev.....	342
wcrtomb.....	332	wscanf.....	179
wcscat.....	332	wunctrl.....	70
wcschr.....	332	xargs.....	56, 389

Index

XCU environment	28
XCU options	55
XCURSES	
quick reference	109
XNS5	5
XPG4 Base	3
XSI headers	65
XSI streams	51
y0	342
y1	342
yacc	23, 56, 390
YACC grammar	23
yn	342
zcat	56, 390

