

IEEE Std 1003.13™-2003

(Revision of
IEEE Std 1003.13-1998)

1003.13™

IEEE Standard for Information Technology— Standardized Application Environment Profile (AEP)—POSIX™ Realtime and Embedded Application Support

IEEE Computer Society

Sponsored by the
Portable Applications Standards Committee



3 Park Avenue, New York, NY 10016-5997, USA

10 September 2004

Print: SH95191

PDF: 599-199



Documents

Document Sharing Hub

*Recognized as an
American National Standard (ANSI)*

IEEE Std 1003.13™-2003
(Revision of
IEEE Std 1003.13-1998)

IEEE Standard for Information Technology— Standardized Application Environment Profile (AEP)—POSIX™ Realtime and Embedded Application Support

Sponsor
Portable Applications Standards Committee
of the
IEEE Computer Society

Approved 12 May 2004
American National Standards Institute

Approved 10 December 2003
IEEE-SA Standards Board

Abstract: This standard is part of the POSIX series of standardized profiles for open systems. It defines environment profiles for portable realtime and embedded applications.

Keywords: AEP, application portability, data processing environment, embedded, open systems, operating system, portable application, POSIX profiles, realtime, realtime application environments

MS-DOS is a registered trademark of Microsoft Corporation.

QNX is a registered trademark of QNX Software Systems, Ltd.

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

pSOS is a registered trademark of Wind River Systems, Inc.

RTLinux is a product of FSMLabs, Inc.

RT-11 is a trademark of Mentec Inc.

RSX-11M is a trademark of Mentec Inc.

RTEMS is free software developed by OAR Corporation for the U.S. Army Missile Command.

UNIX is a registered trademark of The Open Group.

VRTX32 is a registered trademark of Mentor Graphics Corporation.

VxWorks is a registered trademark of Wind River Systems, Inc.

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

Copyright © 2004 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 10 September 2004. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

Print: ISBN 0-7381-3885-1 SH95191
PDF: ISBN 0-7381-3886-X SS95191

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

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

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

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

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

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

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

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

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

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

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

Contents

	PAGE
List of Figures	vii
List of Tables	viii
Introduction	x
Section 1: Overview	1
1.1 Scope	1
1.2 Taxonomy Position	2
1.2.1 Rationale for Positioning (informative)	2
1.3 Realtime System Profiles	2
1.3.1 Minimal Realtime System Profile (PSE51)	2
1.3.2 Realtime Controller System Profile (PSE52)	3
1.3.3 Dedicated Realtime System Profile (PSE53)	3
1.3.4 Multi-Purpose Realtime System Profile (PSE54)	3
1.4 Units of Functionality	4
1.5 Development Environment	4
1.6 Summary of Profile Features	17
Section 2: Normative References	23
2.1 Normative References	23
Section 3: Terms and Definitions	25
3.1 Terminology	25
3.2 Definitions	26
3.3 Rationale for Definitions (informative)	28
Section 4: Conventions and Abbreviations	31
4.1 Conventions	31
4.2 Abbreviations	32
Section 5: Conformance	33
5.1 Conformance	33
5.1.1 Implementation Conformance	33
5.1.2 Application Conformance	34
Section 6: Minimal Realtime System Profile (PSE51)	37
6.1 Introduction	37
6.1.1 Identification	37
6.1.2 Conformance	37
6.1.3 Options	38
6.1.4 The Compilation Environment (C language option)	38

COPYRIGHT © 2004 IEEE. ALL RIGHTS RESERVED.

6.2	Operating System Interface Requirements	39
6.2.1	POSIX.1 Interfaces (C Language Option)	39
6.2.2	POSIX.26 Interfaces (C Language Option)	40
6.2.3	POSIX.5c Interfaces (Ada Language Option)	40
6.3	Application Constraints	42
6.3.1	Constraints related to POSIX.1 Interfaces (C Language Option) . .	42
6.3.2	Constraints related to POSIX.5c Interfaces (Ada Language Option).	43
6.4	Shell and Utility Requirements	45
6.5	Development Platform Requirements	45
6.5.1	C Language Development Option	45
6.5.2	Ada Language Development Option	45
6.6	Rationale for Operating System Requirements (informative)	46
6.6.1	Operating System Interface Requirements	46
6.6.2	Shell and Utility Requirements	54
6.6.3	Development Platform Requirements	54
Section 7:	Realtime Controller System Profile (PSE52)	55
7.1	Introduction.	55
7.1.1	Identification.	55
7.1.2	Conformance	55
7.1.3	Options	56
7.1.4	The Compilation Environment (C language option).	56
7.2	Operating System Interface Requirements	57
7.2.1	POSIX.1 Interfaces (C language Option).	57
7.2.2	POSIX.26 Interfaces (C Language Option)	58
7.2.3	POSIX.5c Interfaces (Ada Language Option)	59
7.3	Application Constraints	60
7.3.1	Constraints Related to POSIX.1 Interfaces (C Language Option). .	61
7.3.2	Constraints related to POSIX.5c Interfaces (Ada Language Option).	61
7.4	Shell and Utility Requirements	62
7.5	Development Platform Requirements	62
7.5.1	C Language Development Option	62
7.5.2	Ada Language Development Option	63
7.6	Rationale for Operating System Requirements (informative)	63
7.6.1	Operating System Interface Requirements	63
7.6.2	Shell and Utility Requirements	71
7.6.3	Development Platform Requirements	72
Section 8:	Dedicated Realtime System Profile (PSE53)	73
8.1	Introduction.	73
8.1.1	Identification.	73
8.1.2	Conformance	73
8.1.3	Options	74
8.1.4	The Compilation Environment (C language option).	74
8.2	Operating System Interface Requirements	75
8.2.1	POSIX.1 Interfaces (C Language Option)	75
8.2.2	POSIX.26 Interfaces (C Language Option)	77

COPYRIGHT © 2004 IEEE. ALL RIGHTS RESERVED.

8.2.3 POSIX.5c Interfaces (Ada Language Option)	77
8.3 Application Constraints	79
8.3.1 Constraints related to POSIX.1 Interfaces (C Language Option) . .	79
8.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option).	79
8.4 Shell and Utility Requirements	80
8.5 Development Platform Requirements	80
8.5.1 C Language Development Option	80
8.5.2 Ada Language Development Option	81
8.6 Rationale for Operating System Requirements (informative)	81
8.6.1 Operating System Interface Requirements	81
8.6.2 Shell and Utility Requirements	89
8.6.3 Development Platform Requirements	90
Section 9: Multi-Purpose Realtime System Profile (PSE54)	91
9.1 Introduction.	91
9.1.1 Identification.	91
9.1.2 Conformance	91
9.1.3 Options	92
9.1.4 The Compilation Environment (C language option)	92
9.2 Operating System Interface Requirements.	93
9.2.1 POSIX.1 Interfaces (C Language Option)	93
9.2.2 POSIX.26 Interfaces (C Language Option)	95
9.2.3 POSIX.5c Interfaces (Ada Language Option)	95
9.3 Application Constraints	97
9.3.1 Constraints Related to POSIX.1 Interfaces (C Language Option) . .	97
9.3.2 Constraints Related to POSIX.5c Interfaces (Ada Language Option).	98
9.4 Shell and Utility Requirements	98
9.5 Development Platform Requirements	99
9.5.1 C Language Development Option	99
9.5.2 Ada Language Development Option	99
9.6 Rationale for Operating System Requirements (informative)	100
9.6.1 Operating System Interface Requirements	100
9.6.2 Shell and Utility Requirements	108
9.6.3 Development Platform Requirements	108
Annex A: POSIX Profiles Package (Ada Language).	109
Annex B: Description of Optional Interfaces	111
B.1 POSIX.1 Options.	111
B.2 POSIX.5c Options	120
Annex C: Bibliography.	123
C.1 Related Open Systems Standards	123
C.2 Other Documents	123
Alphabetic Topical Index.	125

COPYRIGHT © 2004 IEEE. ALL RIGHTS RESERVED.

List of Figures

Figure I.1 — Main Building Blocks of the Profiles.	xvii
--	------

Copyright © 2004 IEEE. All rights reserved.

List of Tables

Table 1-1 — POSIX.1 Units of Functionality	5
Table 1-2 — POSIX.5 Units of Functionality (Ada Language Support) . .	10
Table 1-3 — POSIX.5 Units of Functionality (Device IO)	11
Table 1-4 — POSIX.5 Units of Functionality (Device Specific)	11
Table 1-5 — POSIX.5 Units of Functionality (Event Management)	12
Table 1-6 — POSIX.5 Units of Functionality (FD Management)	12
Table 1-7 — POSIX.5 Units of Functionality (FIFO)	12
Table 1-8 — POSIX.5 Units of Functionality (File Attributes)	12
Table 1-9 — POSIX.5 Units of Functionality (File System)	13
Table 1-10 — POSIX.5 Units of Functionality (Job Control)	13
Table 1-11 — POSIX.5 Units of Functionality (Multi-Process)	14
Table 1-12 — POSIX.5 Units of Functionality (Networking)	14
Table 1-13 — POSIX.5 Units of Functionality (Pipes)	14
Table 1-14 — POSIX.5 Units of Functionality (Signals)	15
Table 1-15 — POSIX.5 Units of Functionality (Single Process)	15
Table 1-16 — POSIX.5 Units of Functionality (System Database)	16
Table 1-17 — POSIX.5 Units of Functionality (User Groups)	16
Table 1-18 — Units of Functionality Requirements	17
Table 1-19 — POSIX.1 Option Requirements	18
Table 1-21 — POSIX.1 Options vs. POSIX.5c Options	20
Table 1-20 — Requirements for Other Standards	20
Table 6-1 — POSIX.1 Units of Functionality Requirements	39
Table 6-2 — POSIX.1 Option Requirements	39
Table 6-3 — POSIX.5c Units of Functionality Requirements	40
Table 6-4 — POSIX.5c Option Requirements	41
Table 6-5 — Functions required to be async-signal-safe	42
Table 7-1 — POSIX.1 Units of Functionality Requirements	57
Table 7-2 — POSIX.1 Option Requirements	57
Table 7-3 — POSIX.5c Units of Functionality Requirements	59
Table 7-4 — POSIX.5c Option Requirements	59
Table 7-5 — Functions required to be async-signal-safe	61
Table 8-1 — POSIX.1 Units of Functionality Requirements	75
Table 8-2 — POSIX.1 Option Requirements	75
Table 8-3 — POSIX.5c Units of Functionality Requirements	77
Table 8-4 — POSIX.5c Option Requirements	77
Table 9-1 — POSIX.1 Units of Functionality Requirements	93
Table 9-2 — POSIX.1 Option Requirements	94
Table 9-3 — POSIX.1 Units of Functionality Requirements	95

Copyright © 2004 IEEE. All rights reserved.

Table 9-4 — POSIX.5c Option Requirements. 96

Table 9-5 — Shell and Utilities Option Requirements
(C Language Option) 98

Table 9-6 — Shell and Utilities Option Requirements
(Ada Language Option) 98

Table B-1 — Functions under each POSIX.1
System Interface Option 111

Table B-2 — Utilities under each POSIX.1 Shell
and Utilities Option 119

Table B-3 — Packages and Subprograms under
Each POSIX.5c Option. 120

Introduction

(This introduction is not a normative part of IEEE Std 1003.13™-2003, IEEE Standard for Information Technology—Standardized Application Environment Profile (AEP)—POSIX™ Realtime and Embedded Application Support.)

The purpose of this standard is to define realtime and embedded application environments based on the ISO/IEC 9945 series of standards. It is intended for realtime systems implementors and realtime applications software developers.

This standard is a revision of IEEE Std 1003.13-1998, where four realtime application environment profiles (or POSIX subsets) are defined. The goal of this revision is to update each of the four profiles according to implementation experience and to add the services defined in the following, newly approved POSIX standards:

- ISO/IEC 9945:2003 {3} (identical to IEEE Std 1003.1™-2003)
- IEEE 1003.26™-2003 {4}
- ISO/IEC 14519:2001 {5}

The base standard, ISO/IEC 9945:2003 {3}, allows profiling standards supporting functional requirements less than those required in the full base standard to subset both mandatory and optional functionality required for POSIX Conformance (see the Base Definitions volume of POSIX.1 {3}, Section 2.1.5.1). POSIX.13 articulates these subprofiling options through Units of Functionality, defined herein, and by use of named options defined in the base standard.

This standard specifies realtime profiles both for the C language and for the Ada language options. Because Ada Bindings to ISO/IEC 9945:2003 {3} are currently under development, the C language option contains more services than the Ada language option in the current draft. If these Ada Bindings are completed before this proposed standard is sent to ballot, the draft will be amended to incorporate them. Otherwise, an amendment of IEEE Std 1003.13-2003 will be produced in the future, to incorporate the added Ada language services.

This standard is designed to support building systems where not all the interconnected boxes use the same profile, for example, a hierarchical system where the bottom-level device controllers use the “minimal” profile, the next level up follows the larger “control” profile, and so on. There are interfaces called out for the smaller profiles that make no sense in an isolated box; those interfaces are there solely to support the construction of heterogeneous systems and systems of communicating peers. Such systems are very common in practice.

To summarize, this standard is embedded in a much larger and widely supported set of standards, which yields benefits during code development, as much

Copyright © 2004 IEEE. All rights reserved.

development and testing is done on the larger and more comfortable systems. It also may be used in the construction of large and heterogeneous systems.

Four profiles have been defined to reflect the wide range of system requirements presented by realtime designs. The intent is to provide a meaningful and coherent set of interfaces that will provide software vendors and consumers with a uniform framework for describing and specifying operating system capabilities. This allows an application writer to construct an application that may be easily moved to a different system that supports the same profile. Similarly, it allows a vendor to claim conformance with an established standard, even if that vendor's implementation does not support the full POSIX feature set.

Initially, the focus of this standard is to provide standardized environments supporting the C language. Options are provided for bindings to the Ada programming language as well as for the C language. Bindings for other languages to these services may be developed and this standard will be updated as appropriate.

Within this standard, the term “POSIX.13” refers to this standard, IEEE Std 1003.13-2003.

Organization of this Standard

This standard is divided into nine elements:

- (1) Overview (Section 1)
- (2) Normative references (Section 2)
- (3) Terms and definitions (Section 3)
- (4) Conventions and abbreviations (Section 4)
- (5) Conformance (Section 5)
- (6) The various realtime profiles (Section 6, Section 7, Section 8, and Section 9)
- (7) The POSIX Profiles package in Ada (Annex A)
- (8) A description of the optional interfaces of the base standards (Annex B)
- (9) Bibliography (Annex C)

References are provided to direct the reader to other related sections.

Informative annexes are not normative parts of the standard and are provided for information only. They are provided for guidance and to help understanding.

Copyright © 2004 IEEE. All rights reserved.

Base Documents

The various realtime application environments described herein are based on the ISO/IEC 9945 and IEEE 1003 family of documents as well as ISO 9899 (C99 Language) and 8652 (Ada95 Language).

Scenario

This standard is based directly on existing small and/or realtime [typically non-UNIX^{®1)}] kernel practice as well as the growing body of practice with POSIX conformant kernels having realtime features. The general approach taken in this standard is to specify interfaces (taken from POSIX) sufficient to deliver the functionality typical of current realtime systems (see Table 1-18 through Table 1-21).

Each profile is specified with full features, to give users clear direction. Vendors may provide means to configure out those parts that are not needed by specific applications. Vendors wishing to expand on the specified profiles are strongly encouraged to take the added interfaces from other POSIX.13 profiles or from the base standards, rather than invent new interfaces.

For each profile, the minimum hardware typically required is specified. This is the hardware assumed to be present; implementations may, of course, have more, but nothing in the profile requires—either directly or indirectly—more than the specified minimum hardware model.

Audience

The intended audience for this class of profiles is all persons concerned with an industry-wide standard realtime application environment based on the POSIX suite of standards. This includes at least four groups of people:

- (1) Persons buying hardware and software systems.
- (2) Persons managing companies that are deciding on future corporate computing directions.
- (3) Persons implementing realtime operating systems.
- (4) Persons developing realtime applications where portability is a primary objective.

1) UNIX is a registered trademark of The Open Group.

Copyright © 2004 IEEE. All rights reserved.

Rationale on Background

This clause contains rationale common to all four realtime profiles.

The developers of POSIX.13 represent a cross section of hardware manufacturers, vendors of operating systems and other software development tools, software designers, consultants, academics, authors, applications programmers, and others. In the course of their deliberations, the developers reviewed related U.S. and international standards, both published and in progress.

Conceptually, POSIX.13 describes a set of application environment profiles (AEPs) needed for the construction and execution of portable realtime application programs.

The developers of this standard have tried to capture the functionality of existing realtime systems in a reasonable number of profiles that specify predominate application environments. It is felt that these profiles, although not optimum, are a best fit to existing classes of applications and systems.

Features of several commercial realtime kernels were considered during the development of the 1998 version of POSIX.13. These included **pSOS**^{™, 2)}, **VRTX32**^{®, 3)} and **VxWorks**^{®, 4)}. Since these products were commercially successful, they must have addressed a significant market segment. In addition, the uniprocessor subset of VITA's **ORKID** specification, NGCR's "**Tiny Real Time**" (**TRT**), and the **uITRON** specification were examined. These were all proposed standard interfaces for small realtime embedded systems.

Features of other commercial realtime kernels such as **RTLinux**⁵⁾ and **QNX**^{®, 6)} as well as free software products such as **RTEMS**⁷⁾ were considered during the development of the current revision of POSIX.13.

The following is a list of features that are representative of current realtime systems and highlights the range of system requirements. While some concepts are common to virtually all implementations (e.g., preemptive, priority-based scheduling), some only apply to smaller systems (e.g., a single address space), and some only to more full-featured systems (e.g., network support, self-hosting).

Basic Realtime Multitasking and Synchronization

- Multiple flows of control
- Preemptive priority scheduling of flows of control
- One address space for all flows of control

2) **pSOS** is a registered trademark of Wind River Systems, Inc.

3) **VRTX32** is a registered trademark of Mentor Graphics Corporation.

4) **VxWorks** is a registered trademark of Wind River Systems, Inc.

5) **RTLinux** is a product of FSMLabs, Inc.

6) **QNX** is a registered trademark of QNX Software Systems, Ltd.

7) **RTEMS** is free software developed by OAR Corporation for the U.S. Army Missile Command.

Copyright © 2004 IEEE. All rights reserved.

- Direct control of location of memory areas
- Inter-thread communications mechanism via message passing (queues)
- Binary and counting semaphores, without priority inheritance
- Mutual exclusion, with optional priority inheritance or priority ceiling protocols
- Local or global event flags (one thread awaits multiple things)
- Multiple memory areas, with both fixed- and variable-sized block allocation policies
- System time in units of clock ticks
- Timeouts on all blocking services in units of clock ticks
- Hardware interrupt control and support for user interrupt handlers
- Signals
- Exception handling
- Minimal synchronous I/O interface: *open()*, *close()*, *read()*, *write()*, *ioctl()*, *posix_devctl()*
- Debugger interface
- No memory protection
- Application runs in privileged (supervisor) mode, if applicable
- Direct I/O, rather than via kernel
- System executable size and memory requirements are major constraints

I/O

Realtime systems supporting I/O generally provide the following features:

- Named I/O devices
- Support for serial I/O lines
- Pipes
- Installable user device drivers
- Memory-mapped I/O

Copyright © 2004 IEEE. All rights reserved.

Local File System

Realtime systems supporting a file system generally provide the following features:

- Named files
- Hierarchical file system (directories)
- Contiguous preallocation of disk space
- May provide media compatibility with another file system [e.g., **MSDOS**⁸⁾ or **RT-11**⁹⁾]
- No user IDs or file protection

Historically, file systems for embedded realtime systems typically have had a one-level name space, contiguous allocation of disk space, and relatively short filenames. They have not supported an arbitrary hierarchy of named directories, non-contiguous allocation of disk space, or long filenames. They may have had numbered directories [e.g., **RSX-11M**¹⁰⁾], or only contiguous allocation of disk space (e.g., **RT-11**TM)

However, recent commercial offerings have supported multilevel named directories and both contiguous and non-contiguous disk space allocation. In these implementations, the support of these features with potentially nondeterministic performance does not preclude an application from restricting itself to features with deterministic performance. For example, it is still possible to use contiguous files exclusively. Because it is relatively easy to implement both, and need not interfere with deterministic performance, the working group did not make a distinction between realtime and time-sharing file systems in this AEP.

Although few embedded systems had a hard drive and a file system, present flash memory technology has enabled embedded systems, even those with strict vibration requirements, to have a file system resident on this kind of nonvolatile media. This has caused the POSIX.13 profile designed for large embedded systems, the Dedicated Realtime System Profile (PSE53), to incorporate a simplified file system in this new revision of the standard.

Traditional implementations of POSIX.1 file systems employ a disk buffer cache to improve average performance by reducing the number of physical media accesses and by reordering the accesses to take advantage of the characteristics of rotating media. These implementations have not made a distinction between the buffering of data transfers [*read()* and *write()*], and directory operations [*creat()*, *link()*, *unlink()*, *mkdir()*, *rmdir()*, *rename()*]. A result of this is that a system crash at an unexpected moment can leave

8) **MS-DOS** is a registered trademark of Microsoft Corporation.

9) **RT-11** is a trademark of Mentec Inc.

10) **RSX-11M** is a trademark of Mentec Inc.

Copyright © 2004 IEEE. All rights reserved.

the file system in a corrupted state. This situation is usually corrected at the next system reboot by a file system checker and recovery program, such as `fsck`. The checking and correcting of a corrupted file system may take a long and variable amount of time to perform, may require a human operator to monitor and control its progress, and may nonetheless fail to repair the file system. Any one of these characteristics would make a file system check unacceptable for some embedded realtime applications. It was therefore suggested that such applications limit their use of directory operations to *safe* times, and that implementations maintain the file system in such a way that a file system check during reboot is avoided. This was considered, but rejected on the grounds that not all applications would require the capability, and that it was neither specifiable nor testable.

Network Communication

Realtime systems supporting networking generally provide the following features:

- Compatibility with a protocol stack (e.g., TCP/IP)
- May support applications such as FTP, TELNET, TFTP, rcp

Distributed File System

Realtime systems supporting a distributed (non-local) file system generally provide the following features:

- Remote access to a file system
- Performance not realtime

Memory Protection

Realtime systems supporting memory protection (typically requiring a memory management unit) generally provide the following features:

- Memory mapping and protection
- Ability to map to special areas of memory (I/O page, frame buffer)
- Typically do not have demand paging for realtime parts

Multiprocessor Support

Realtime systems supporting multiprocessing generally provide one of the following methods:

- **network**
Non-transparent access to remote objects, remote procedure calls
- **distributed**
Transparent access to objects, no load-balancing

Copyright © 2004 IEEE. All rights reserved.

- **symmetric**
Presence of a global task scheduling queue (may also have local scheduling queues)

Self-Hosting

Realtime systems supporting the capability for program development, text editing, compilation, etc., generally provide the following features:

- Shell
- Text editor
- Compiler, assembler, linker, debugger
- May have user ID protection

Only the larger profiles (i.e., PSE54) are likely to be self-hosted.

Overview of the Profiles Structure (Rationale)

This clause contains rationale common to all four realtime profiles.

The four profiles defined in this standard are designed to make applications upwards compatible to higher profiles. Figure I.1 shows the main building blocks of each of the four profiles specified in this standard. Please note that the full differences between the different profiles are more complex than those appearing on this figure. See 1.6 for a full description of the differences between the profiles.

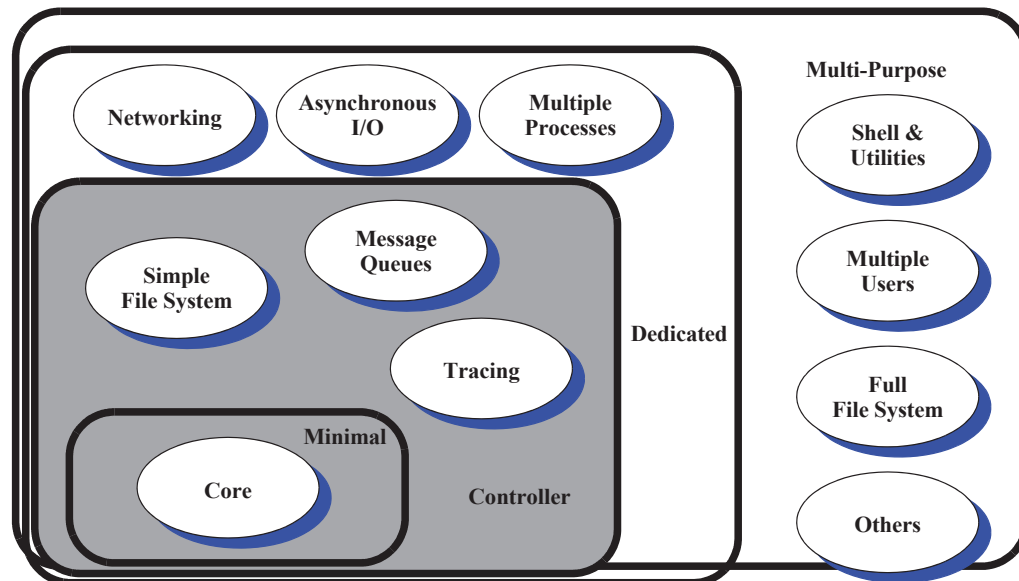


Figure I.1 — Main Building Blocks of the Profiles

Copyright © 2004 IEEE. All rights reserved.

The “core” building block in Figure I.1 refers to the Units of Functionality and options required in all four profiles. See 6.2 for a description of the core services. Profiles with only one implicit process (PSE51 and PSE52) are shaded in the figure, to highlight this major difference with the larger profiles, which require support for multiple processes (and thus require having an MMU).

Related Standards Activities

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

The following areas are under active consideration at this time or are expected to become active in the near future¹¹⁾:

- (1) Additional system application program interfaces (APIs) in C language
- (2) Ada language bindings
- (3) Additional realtime facilities
- (4) Fault tolerance
- (5) Profiles describing application- or user-specific combinations of Open Systems standards

If you have interest in participating in the Portable Application Standards Committee (PASC) working groups addressing these issues, please send your name, address, and telephone number to

*Secretary, IEEE Standards Board
Institute of Electrical and Electronics Engineers, Inc.
445 Hoes Lane
Piscataway, NJ 08854
USA*

When writing, ask to have your letter forwarded to the chair of the appropriate PASC working group.

If you have interest in participating in this work at the international level, contact your ISO/IEC national body.

11) A Standards Status Report that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone: +1 202 371-0101; FAX: +1 202 728-9614. Working drafts of POSIX standards under development are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854 (<http://www.standards.ieee.org/>).

Copyright © 2004 IEEE. All rights reserved.

Notice to Users

Errata

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

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

Patents

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

Participants in the 2003 Version

IEEE Std 1003.13-2003 was prepared by the System Services Working Group—Realtime, sponsored by the Portable Application Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the System Services Working Group—Realtime was as follows:

Portable Application Standards Committee

Chair:	Lowell G. Johnson
Vice Chair:	Joseph M. Gwinn
Functional Vice Chairs:	Jay Ashford Andrew Josey Curtis Royster, Jr.
Secretary:	Nick Stoughton

Copyright © 2004 IEEE. All rights reserved.

IEEE System Services Working Group—Realtime

Chair:	Joseph M. Gwinn
Secretary:	Karen D. Gordon
Technical Editor:	Michael González
Ballot Coordinator:	James T. Oblinger
Technical Reviewer:	Michael González

Working Group

At the time this standard was completed, the IEEE System Services Working Group—Realtime had the following members:

Pierre-Jean Arcos	Dave Emery	François Riche
Ted Baker	Andrew Josey	Doug Robinson
Bob Barned	Jim Litchfield	Greg Shelton
Donald Cragun	Franklin C. Prindle	Peter van der Veen
	Ray Richards	

Balloting Committee

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Alejandro Alonso	Scott Gudgel	François Riche
Pierre-Jean Arcos	Joseph M. Gwinn	John Riley
Mario Barbacci	Barry Hedquist	Curtis Royster, Jr.
Mitchell Bonnett	Charles Hammons	Diane Schleicher
Mark Brown	Karl Heubaum	Stephen Schwarm
Paul Buerger	Lowell G. Johnson	Yuriy Sheynin
Donald Cragun	Andrew Josey	Gil Shultz
John Davies	Joerg Kampmann	Keld Simonsen
Juan Antonio de la Puente	Bill LaPlant	Arun Subbarao
Sourav Dutta	Roger J. Martin	Mark-Rene Uchida
Yaacov Fenster	George Miao	Srinivasa Vemuru
Michel Gien	James T. Oblinger	Rina Walach
Michael González	Peter Petrov	Oren Yuen
Karen D. Gordon	Franklin C. Prindle	Janusz Zalewski
	Vikram Punj	

Copyright © 2004 IEEE. All rights reserved.

When the IEEE-SA Standards Board approved this standard on 10 December 2003, it had the following membership:

Don Wright, Chair
Howard M. Frazier, Vice Chair
Judith Gorman, Secretary

H. Stephen Berger	Donald N. Heirman	Daleep C. Mohla
Joseph A. Bruder	Laura Hitchcock	William J. Moylan
Bob Davis	Richard H. Hulett	Paul Nikolich
Richard DeBlasio	Anant Kumar Jain	Gary S. Robinson
Julian Forster*	Lowell G. Johnson	Malcolm V. Thaden
Toshio Fukuda	Joseph L. Koepfinger*	Geoffrey O. Thompson
Arnold M. Greenspan	Tom McGean	Doug Topping
Raymond Hapeman	Steve M. Mills	Howard L. Wolfman

*Member Emeritus

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

Alan Cookson, *NIST Representative*
Satish K. Aggarwal, *NRC Representative*

Savoula Amanatidis
IEEE Standards Managing Editor

Participants in the 1998 Version

IEEE Std 1003.13-1998 was prepared by the System Services Working Group—Realtime, sponsored by the Portable Application Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the System Services Working Group—Realtime was as follows:

Portable Application Standards Committee

Chair:	Lowell G. Johnson
Vice Chair:	Joseph M. Gwinn
Functional Vice Chairs:	Jay Ashford
	Andrew Josey
	Curtis Royster, Jr.
Secretary:	Nick Stoughton

Copyright © 2004 IEEE. All rights reserved.

IEEE System Services Working Group—Realtime

Chair:	Joseph M. Gwinn Susan Corwin (to 1995)
Secretary:	Karen D. Gordon Franklin C. Prindle (1996) Lee Schermerhorn (to 1994)
Technical Editor:	Bob Luken
Ballot Coordinators:	Andrew E. Wheeler, Jr. James T. Oblinger
Technical Reviewers:	Andrew E. Wheeler, Jr. Joseph M. Gwinn Karen D. Gordon

Working Group

Ray Alderman	Michael Feustel	Dave Lungert
Larry Anderson	Bill Gallmeister	Bill Maes
Pierre-Jean Arcos	Michael González	James T. Oblinger
Charles R. Arnold	Karen D. Gordon	Offer Pazy
V. Raj Avula	Randy Greene	Carolyn Petersen
Theodore P. Baker	Rick Greer	Dave Plauger
Todd Bargorek	Joseph M. Gwinn	Arlan Pool
Robert Barned	Steven A. Haaser	Franklin C. Prindle
Richard M. Bergman	Barbara Haleen	François Riche
Nawaf Bitar	Geoffrey R. Hall	Robert Rose
Steve Brosky	Patrick Hebert	Gordon W. Ross
David Butenhof	Mary R. Hermann	Barry Ruzek
Hans Pietter Christiansen	David Hughes	Webb Scales
Dave Cooper	Duane Hughes	Lee Schermerhorn
Susan Corwin	Michael B. Jones	Del Swanson
Bill Cox	Steven Kleiman	Barry Traylor
June R. Curtis	Robert Knighten	Stephen R. Walli
Peter Dibble	C. Douglass Locke	Andrew E. Wheeler, Jr.
Christoph Eck	Kent Long	David Wilner
Maryland R. Edwards	Robert D. Luken	John Zolnowsky

Balloting Group

The following persons were members of the 1003.13 Balloting Group that approved the standard for submission to the IEEE Standards Board:

Norman Aaronson	Andy R. Bihain	Lee Damico
Alejandro Alonso-Muñoz	Shirley Bockstahler-Brandt	Christoph Eck
Pierre-Jean Arcos	Steve Case	James A. Eiler
Charles R. Arnold	Hu Cheng	Philip H. Enslow
Theodore P. Baker	Hans Pietter Christiansen	Donna K. Fisher
Robert Barned	Susan Corwin	Michel Gien
Jason Behm	Donald Cragun	John Gilbert
Richard M. Bergman	June R. Curtis	Michael González

Copyright © 2004 IEEE. All rights reserved.

Karen D. Gordon
Mars J. Gralia
Randy Greene
Joseph M. Gwinn
Steven A. Haaser
Geoffrey R. Hall
Patrick Hebert
Hans H. Heilborn
Duane Hughes
Hal Jespersen
Michael B. Jones
Joe Kelsey
Judy Kerner
Lawrence J. Kilgallen

Martin J. Kirk
Thomas M. Kurihara
Kevin Lewis
C. Douglass Locke
Kent Long
James P. Lonjers
Lee W. Lucas
Robert D. Luken
Dave Lunger
Marshall McKusick
Craig B. Meyers
Diana Norwood
James T. Oblinger
Dave Plauger
Arlan Pool

Franklin C. Prindle
Paul Rabin
Wendy Rauch
Henry H. Robbins
Steven Schwarm
Del Swanson
Sandra Swearingen
James G. Tanner
Mark-Rene Uchida
Andrew E. Wheeler, Jr.
David Wilner
Oren Yuen
John J. Zenor
John Zolnowsky

When the IEEE-SA Standards Board approved IEEE 1003.13 on 19 March 1998, it had the following membership:

Richard J. Holleman *Chair*
Donald N. Heirman, *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Clyde R. Camp
James T. Carlo
Gary R. Engmann
Harold E. Epstein
Julian Forster*
Thomas F. Garrity
Ruben D. Garzon

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

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

*Member Emeritus

Noelle Humenick
IEEE Standards Project Editor

Copyright © 2004 IEEE. All rights reserved.

This page is left intentionally blank.

IEEE Standard for Information Technology— Standardized Application Environment Profile (AEP)—POSIX™ Realtime and Embedded Application Support

Section 1: Overview

1.1 Scope

This standard establishes a set of Realtime and Embedded Environment Profiles based on ISO/IEC 9945:2003 {3}, IEEE Std 1003.26™-2003 {4}, ISO/IEC 14519:2001 {5}, and related standards specifying foundations for realtime applications. It is a revision of the previous IEEE Std 1003.13™-1998, which established Realtime Profiles based on ISO/IEC 9945-1:1990, as amended by IEEE Std 1003.1b™-1993, IEEE Std 1003.5b™-1996, and ISO/IEC 9945-2:1993. Both C {2} and Ada {1} language application program interfaces are addressed in this standard.

The Application Environment Profiles (AEPs) specified herein are appropriate for the development and execution of realtime or embedded applications using the services and utilities provided by standards called out in this standard.

Copyright © 2004 IEEE. All rights reserved.

1.2 Taxonomy Position

P— OSE Profiles

AEP— Application Environment Profiles

PS— System Profiles

PSE— Generic Environment Profiles

PSE5— Realtime Environments

PSE51— Minimal Realtime System Profile

PSE52— Realtime Controller System Profile

PSE53— Dedicated Realtime System Profile

PSE54— Multi-Purpose Realtime System Profile

1.2.1 Rationale for Positioning (informative)

(This subclause is not a normative part of IEEE Std 1003.13-2003.)

This standard contains requirements for Application Program Interfaces and Units of Functionality necessary to support four instances of the Generic Realtime Environment class of applications. It specifies the behavior to be observed at the interfaces of the Application Platform on which the class of applications can run. This subset of an Open System Environment (OSE) profile is complete and coherent within the context of the class of applications supported. As such, it is a System Profile class of AEP.

1.3 Realtime System Profiles

This standard describes four realtime profiles and their minimum hardware requirements.

1.3.1 Minimal Realtime System Profile (PSE51)

PSE51 systems are typically embedded in larger systems dedicated to unattended control of one or more special I/O devices. Neither user interaction nor a file system (mass storage) is required. The programming model is that of a single (implicit)

Copyright © 2004 IEEE. All rights reserved.

POSIX process (corresponding to the processor's hardware address space), containing one or more threads of control (POSIX.1 threads or Ada tasks). Although there is only one process, a Message Passing interface is provided for communications among threads of control and between PSE5X instantiations. Special devices are operated and controlled either by memory-mapped I/O or by the basic I/O interface, which provides a standard way to access the intrinsically nonstandard I/O hardware and its nonportable control code.

The hardware model for this profile assumes a single processor with its memory, but no memory management unit (MMU) or common I/O devices are required. (If there are in fact multiple processors, typically there are multiple instantiations of the operating system, perhaps communicating via shared memory or a backplane channel, perhaps isolated.)

1.3.2 Realtime Controller System Profile (PSE52)

These systems are an extension of the Minimal Realtime System Profile. Support for a file system interface and asynchronous (nonblocking) I/O interfaces has been added.

The hardware model for this profile assumes a single processor and memory space (an MMU is not required). Mass storage devices are not required; the file system may, for instance, be implemented in memory (RAM disk or flash memory).

1.3.3 Dedicated Realtime System Profile (PSE53)

These systems are an extension of the Realtime Controller System Profile. Support for multiple processes has been added. Although these are usually embedded systems, flash memory technology enables presence of a simplified file system, even in those systems with mechanical or environmental requirements that preclude a rotating-media hard drive. Since memory management hardware may be provided, the functionality of memory locking is provided.

The hardware model for this profile assumes one or more processors, each with its own MMU, in the same system.

1.3.4 Multi-Purpose Realtime System Profile (PSE54)

These systems include all the functionality of the other three profiles. They provide comprehensive functionality and run a mix of differing realtime and non-realtime tasks. This functionality includes most of POSIX.1 {3} and/or POSIX.5c {5}. Since

Copyright © 2004 IEEE. All rights reserved.

users may conduct interactive sessions on those systems, all the mandatory elements of the Shell and Utilities volume of POSIX.1 {3} are also included. Support for multiple multithreaded processes is required so that multitasking may be done by threads (POSIX.1 threads or Ada tasks), processes, or both.

The hardware model for this profile assumes one or more processors with memory management units, high-speed storage devices, special interfaces, network support, and display devices. The system supports a mix of realtime and non-realtime tasks, some being interactive user tasks.

1.4 Units of Functionality

Some of the profiles specified in this standard do not require support for all the functionality specified in a referenced standard. In this case, if that referenced standard does not contain options for specifying just the required functionality, only those Units of Functionality referenced by the profile may be used by a strictly conforming application.

Table 1-1 shows the Units of Functionality defined for POSIX.1 {3}; each of these units represents a Subprofiling Option Group (see the Base Definitions volume of POSIX.1 {3}, Section 2.1.5.1) and is a set of interfaces that represents a separately implementable element of POSIX.1 {3}. Table 1-2 through Table 1-17 show the Units of Functionality defined for POSIX.5c {5}.

1.5 Development Environment

Although the Shell and Utilities part of POSIX.1 {3} is not required for the execution environment of PSE51, PSE52, or PSE53, option POSIX2_SW_DEV is required in the development environments for all four profiles. The option POSIX2_C_DEV is required for C language development environments.

Copyright © 2004 IEEE. All rights reserved.

Table 1-1 — POSIX.1 Units of Functionality

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

Copyright © 2004 IEEE. All rights reserved.

Table 1-1 — POSIX.1 Units of Functionality (Continued)

Unit of Functionality	Included Interfaces
POSIX_C_LANG_SUPPORT	<i>abs()</i> , <i>asctime()</i> , <i>asctime_r()</i> , <i>atof()</i> , <i>atoi()</i> , <i>atol()</i> , <i>atoll()</i> , <i>bsearch()</i> , <i>calloc()</i> , <i>ctime()</i> , <i>ctime_r()</i> , <i>difftime()</i> , <i>div()</i> , <i>feclearexcept()</i> , <i>fegetenv()</i> , <i>fegetexceptflag()</i> , <i>fegetround()</i> , <i>fehldexcept()</i> , <i>feraiseexcept()</i> , <i>fesetenv()</i> , <i>fesetexceptflag()</i> , <i>fesetround()</i> , <i>fetestexcept()</i> , <i>feupdateenv()</i> , <i>free()</i> , <i>gmtime()</i> , <i>gmtime_r()</i> , <i>imaxabs()</i> , <i>imaxdiv()</i> , <i>isalnum()</i> , <i>isalpha()</i> , <i>isblank()</i> , <i>iscntrl()</i> , <i>isdigit()</i> , <i>isgraph()</i> , <i>islower()</i> , <i>isprint()</i> , <i>ispunct()</i> , <i>isspace()</i> , <i>isupper()</i> , <i>isxdigit()</i> , <i>labs()</i> , <i>ldiv()</i> , <i>llabs()</i> , <i>lldiv()</i> , <i>localeconv()</i> , <i>localtime()</i> , <i>localtime_r()</i> , <i>malloc()</i> , <i>memchr()</i> , <i>memcmp()</i> , <i>memcpy()</i> , <i>memmove()</i> , <i>memset()</i> , <i>mktime()</i> , <i>qsort()</i> , <i>rand()</i> , <i>rand_r()</i> , <i>realloc()</i> , <i>setlocale()</i> , <i>snprintf()</i> , <i>sprintf()</i> , <i>srand()</i> , <i>sscanf()</i> , <i>strcat()</i> , <i>strchr()</i> , <i>strcmp()</i> , <i>strcoll()</i> , <i>strcpy()</i> , <i>strcspn()</i> , <i>strerror()</i> , <i>strerror_r()</i> , <i>strftime()</i> , <i>strlen()</i> , <i>strncat()</i> , <i>strncmp()</i> , <i>strncpy()</i> , <i>strpbrk()</i> , <i>strrchr()</i> , <i>strspn()</i> , <i>strstr()</i> , <i>strtod()</i> , <i>strtof()</i> , <i>strtoimax()</i> , <i>strtok()</i> , <i>strtok_r()</i> , <i>strtol()</i> , <i>strtold()</i> , <i>strtoll()</i> , <i>strtoul()</i> , <i>strtoull()</i> , <i>strtoumax()</i> , <i>strxfrm()</i> , <i>time()</i> , <i>tolower()</i> , <i>toupper()</i> , <i>tzname</i> , <i>tzset()</i> , <i>va_arg()</i> , <i>va_copy()</i> , <i>va_end()</i> , <i>va_start()</i> , <i>vsprintf()</i> , <i>vsscanf()</i>
POSIX_C_LANG_WIDE_CHAR	<i>btowc()</i> , <i>iswalnum()</i> , <i>iswalphabet()</i> , <i>iswblank()</i> , <i>iswcntrl()</i> , <i>iswctype()</i> , <i>iswdigit()</i> , <i>iswgraph()</i> , <i>iswlower()</i> , <i>iswprint()</i> , <i>iswpunct()</i> , <i>iswspace()</i> , <i>iswupper()</i> , <i>iswxdigit()</i> , <i>mblen()</i> , <i>mbrlen()</i> , <i>mbtowc()</i> , <i>mbsinit()</i> , <i>mbsrtowcs()</i> , <i>mbstowcs()</i> , <i>mbtowc()</i> , <i>swprintf()</i> , <i>swscanf()</i> , <i>towctrans()</i> , <i>towlower()</i> , <i>towupper()</i> , <i>vwprintf()</i> , <i>vwscanf()</i> , <i>wctomb()</i> , <i>wcscat()</i> , <i>wcschr()</i> , <i>wscmp()</i> , <i>wscoll()</i> , <i>wscpy()</i> , <i>wcscspn()</i> , <i>wcsftime()</i> , <i>wcslen()</i> , <i>wcsncat()</i> , <i>wcsncmp()</i> , <i>wcsncpy()</i> , <i>wcspbrk()</i> , <i>wcsrchr()</i> , <i>wcsrtombs()</i> , <i>wcsspn()</i> , <i>wcsstr()</i> , <i>wcstod()</i> , <i>wcstof()</i> , <i>wcstoimax()</i> , <i>wcstok()</i> , <i>wcstol()</i> , <i>wcstold()</i> , <i>wcstoll()</i> , <i>wcstombs()</i> , <i>wcstoul()</i> , <i>wcstoull()</i> , <i>wcstoumax()</i> , <i>wcsxfrm()</i> , <i>wctob()</i> , <i>wctomb()</i> , <i>wctrans()</i> , <i>wctype()</i> , <i>wmemchr()</i> , <i>wmemcmp()</i> , <i>wmemcpy()</i> , <i>wmemmove()</i> , <i>wmemset()</i>
POSIX_DEVICE_IO	<i>clearerr()</i> , <i>close()</i> , <i>fclose()</i> , <i>fdopen()</i> , <i>feof()</i> , <i>ferror()</i> , <i>fflush()</i> , <i>fgetc()</i> , <i>fgets()</i> , <i>fileno()</i> , <i>fopen()</i> , <i>fprintf()</i> , <i>fputc()</i> , <i>fputs()</i> , <i>fread()</i> , <i>freopen()</i> , <i>fscanf()</i> , <i>fwrite()</i> , <i>getc()</i> , <i>getchar()</i> , <i>gets()</i> , <i>open()</i> , <i>perror()</i> , <i>printf()</i> , <i>putc()</i> , <i>putchar()</i> , <i>puts()</i> , <i>read()</i> , <i>scanf()</i> , <i>setbuf()</i> , <i>setvbuf()</i> , <i>stderr</i> , <i>stdin</i> , <i>stdout</i> , <i>ungetc()</i> , <i>vfprintf()</i> , <i>vfprintf()</i> , <i>vscanf()</i> , <i>vprintf()</i> , <i>vscanf()</i> , <i>write()</i>

Copyright © 2004 IEEE. All rights reserved.

Table 1-1 — POSIX.1 Units of Functionality (Continued)

Unit of Functionality	Included Interfaces
POSIX_DEVICE_SPECIFIC	<i>cfgetispeed(), cfgetospeed(), cfsetispeed(), cfsetospeed(), ctermid(), isatty(), tcdrain(), tcflow(), tcflush(), tcgetattr(), tcsetattr(), ttyname(), ttyname_r()</i>
POSIX_EVENT_MGMT	<i>FD_CLR(), FD_ISSET(), FD_SET(), FD_ZERO(), pselect(), select()</i>
POSIX_FD_MGMT	<i>dup(), dup2(), fcntl(), fgetpos(), fseek(), fseeko(), fsetpos(), ftell(), ftello(), ftruncate(), lseek(), rewind()</i>
POSIX_FIFO	<i>mkfifo()</i>
POSIX_FILE_ATTRIBUTES	<i>chmod(), chown(), fchmod(), fchown(), umask()</i>
POSIX_FILE_LOCKING	<i>flockfile(), ftrylockfile(), funlockfile(), getc_unlocked(), getchar_unlocked(), putc_unlocked(), putchar_unlocked()</i>
POSIX_FILE_SYSTEM	<i>access(), chdir(), closedir(), creat(), fpathconf(), fstat(), getcwd(), link(), mkdir(), opendir(), pathconf(), readdir(), readdir_r(), remove(), rename(), rewinddir(), rmdir(), stat(), tmpfile(), tmpnam(), unlink(), utime()</i>
POSIX_FILE_SYSTEM_EXT	<i>glob(), globfree()</i>
POSIX_JOB_CONTROL⁽¹⁾	<i>setpgid(), tcgetpgrp(), tcsetpgrp()</i>
POSIX_MULTI_PROCESS	<i>_Exit(), _exit(), assert(), atexit(), clock(), execl(), execle(), execlp(), execv(), execve(), execvp(), exit(), fork(), getpgrp(), getpid(), getppid(), setsid(), sleep(), times(), wait(), waitpid()</i>
POSIX_NETWORKING	<i>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()</i>
POSIX_PIPE	<i>pipe()</i>
POSIX_REGEX⁽²⁾	<i>regcomp(), regerror(), regexec(), regfree()</i>

Copyright © 2004 IEEE. All rights reserved.

Table 1-1 — POSIX.1 Units of Functionality (Continued)

Unit of Functionality	Included Interfaces
POSIX_RW_LOCKS⁽³⁾	<i>pthread_rwlock_destroy()</i> , <i>pthread_rwlock_init()</i> , <i>pthread_rwlock_rdlock()</i> , <i>pthread_rwlock_timedrdlock()</i> ⁽⁴⁾ , <i>pthread_rwlock_timedwrlock()</i> ^d , <i>pthread_rwlock_tryrdlock()</i> , <i>pthread_rwlock_trywrlock()</i> , <i>pthread_rwlock_unlock()</i> , <i>pthread_rwlock_wrlock()</i> , <i>pthread_rwlockattr_destroy()</i> , <i>pthread_rwlockattr_getpshared()</i> ⁽⁵⁾ , <i>pthread_rwlockattr_init()</i> , <i>pthread_rwlockattr_setpshared()</i> ^e
POSIX_SHELL_FUNC	<i>pclose()</i> , <i>popen()</i> , <i>system()</i> , <i>wordexp()</i> , <i>wordfree()</i>
POSIX_SIGNALS	<i>abort()</i> , <i>alarm()</i> , <i>kill()</i> , <i>pause()</i> , <i>raise()</i> , <i>sigaction()</i> , <i>sigaddset()</i> , <i>sigdelset()</i> , <i>sigemptyset()</i> , <i>sigfillset()</i> , <i>sigismember()</i> , <i>signal()</i> , <i>sigpending()</i> , <i>sigprocmask()</i> , <i>sigsuspend()</i> , <i>sigwait()</i>
POSIX_SIGNAL_JUMP	<i>siglongjmp()</i> , <i>sigsetjmp()</i>
POSIX_SINGLE_PROCESS	<i>confstr()</i> , <i>environ</i> , <i>errno</i> , <i>getenv()</i> , <i>setenv()</i> , <i>sysconf()</i> , <i>uname()</i> , <i>unsetenv()</i>
POSIX_STRING_MATCHING	<i>fnmatch()</i> , <i>getopt()</i> , <i>optarg</i> , <i>optind</i> , <i>opterr</i> , <i>optopt</i>
POSIX_SYMBOLIC_LINKS	<i>lstat()</i> , <i>readlink()</i> , <i>symlink()</i>
POSIX_SYSTEM_DATABASE	<i>getgrgid()</i> , <i>getgrgid_r()</i> , <i>getgrnam()</i> , <i>getgrnam_r()</i> , <i>getpwnam()</i> , <i>getpwnam_r()</i> , <i>getpwuid()</i> , <i>getpwuid_r()</i>
POSIX_THREADS_BASE⁽⁶⁾	<i>pthread_atfork()</i> , <i>pthread_attr_destroy()</i> , <i>pthread_attr_getdetachstate()</i> , <i>pthread_attr_getschedparam()</i> , <i>pthread_attr_init()</i> , <i>pthread_attr_setdetachstate()</i> , <i>pthread_attr_setschedparam()</i> , <i>pthread_cancel()</i> , <i>pthread_cleanup_pop()</i> , <i>pthread_cleanup_push()</i> , <i>pthread_cond_broadcast()</i> , <i>pthread_cond_destroy()</i> , <i>pthread_cond_init()</i> , <i>pthread_cond_signal()</i> , <i>pthread_cond_timedwait()</i> , <i>pthread_cond_wait()</i> , <i>pthread_condattr_destroy()</i> , <i>pthread_condattr_init()</i> , <i>pthread_create()</i> , <i>pthread_detach()</i> , <i>pthread_equal()</i> , <i>pthread_exit()</i> , <i>pthread_getspecific()</i> , <i>pthread_join()</i> , <i>pthread_key_create()</i> , <i>pthread_key_delete()</i> , <i>pthread_kill()</i> , <i>pthread_mutex_destroy()</i> , <i>pthread_mutex_init()</i> , <i>pthread_mutex_lock()</i> , <i>pthread_mutex_trylock()</i> , <i>pthread_mutex_unlock()</i> , <i>pthread_mutexattr_destroy()</i> , <i>pthread_mutexattr_init()</i> , <i>pthread_once()</i> , <i>pthread_self()</i> , <i>pthread_setcancelstate()</i> , <i>pthread_setcanceltype()</i> , <i>pthread_setspecific()</i> , <i>pthread_sigmask()</i> , <i>pthread_testcancel()</i>

Copyright © 2004 IEEE. All rights reserved.

Table 1-1 — POSIX.1 Units of Functionality (Continued)

Unit of Functionality	Included Interfaces
POSIX_USER_GROUPS	<i>getegid(), geteuid(), getgid(), getgroups(), getlogin(), getlogin_r(), getuid(), setegid(), seteuid(), setgid(), setuid()</i>
POSIX_WIDE_CHAR_IO	<i>fgetwc(), fgetws(), fputwc(), fputws(), fwide(), fwprintf(), fwscanf(), getwc(), getwchar(), putwc(), putwchar(), ungetwc(), vfwprintf(), vfwscanf(), vwprintf(), vwscanf(), wprintf(), wscanf()</i>
XSI_C_LANG_SUPPORT	<i>_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(), mrand48(), nrand48(), random(), remque(), seed48(), setcontext(), setstate(), signgam(), srand48(), srandom(), strcasecmp(), strdup(), strfmon(), strncasecmp(), strptime(), swab(), swapcontext(), tdelete(), tfind(), time zone, toascii(), tsearch(), twalk()</i>
XSI_DBM	<i>dbm_clearerr(), dbm_close(), dbm_delete(), dbm_error(), dbm_fetch(), dbm_firstkey(), dbm_nextkey(), dbm_open(), dbm_store()</i>
XSI_DEVICE_IO	<i>fmsg(), poll(), pread(), pwrite(), readv(), writev()</i>
XSI_DEVICE_SPECIFIC	<i>grantpt(), posix_openpt(), ptsname(), unlockpt()</i>
XSI_DYNAMIC_LINKING	<i>dlclose(), dlderror(), dlopen(), dlsym()</i>
XSI_FD_MGMT	<i>truncate()</i>
XSI_FILE_SYSTEM	<i>basename(), dirname(), fchdir(), fstatvfs(), ftw(), lchown(), lockf(), mknod(), mkstemp(), nftw(), realpath(), seekdir(), statvfs(), sync(), telldir(), tempnam()</i>
XSI_I18N	<i>catclose(), catgets(), catopen(), nl_langinfo()</i>
XSI_IPC	<i>ftok(), msgctl(), msgget(), msgrcv(), msgsnd(), semctl(), semget(), semop(), shmat(), shmctl(), shmdt(), shmget()</i>
XSI_JOB_CONTROL	<i>tcgetsid()</i>
XSI_JUMP	<i>_longjmp(), _setjmp()</i>
XSI_MATH	<i>j0(), j1(), jn(), scalb(), y0(), y1(), yn()</i>
XSI_MULTI_PROCESS	<i>getpgid(), getpriority(), getrlimit(), getrusage(), getsid(), nice(), setpgid(), setpriority(), setrlimit(), ulimit(), usleep(), vfork(), waitid()</i>
XSI_SIGNALS	<i>bsd_signal(), killpg(), sigaltstack(), sighold(), sigignore(), siginterrupt(), sigpause(), sigrelse(), sigset(), ualarm()</i>
XSI_SINGLE_PROCESS	<i>gethostid(), gettimeofday(), putenv()</i>
XSI_SYSTEM_DATABASE	<i>endpwent(), getpwent(), setpwent()</i>
XSI_SYSTEM_LOGGING	<i>closelog(), openlog(), setlogmask(), syslog()</i>

Copyright © 2004 IEEE. All rights reserved.

Table 1-1 — POSIX.1 Units of Functionality (Continued)

Unit of Functionality	Included Interfaces
XSI_THREAD_MUTEX_EXT	<i>pthread_mutexattr_gettype()</i> , <i>pthread_mutexattr_settype()</i>
XSI_THREADS_EXT	<i>pthread_attr_getguardsize()</i> , <i>pthread_attr_getstack()</i> , <i>pthread_attr_setguardsize()</i> , <i>pthread_attr_setstack()</i> , <i>pthread_getconcurrency()</i> , <i>pthread_setconcurrency()</i>
XSI_TIMERS	<i>getitimer()</i> , <i>setitimer()</i>
XSI_USER_GROUPS	<i>endgrent()</i> , <i>endutxent()</i> , <i>getgrent()</i> , <i>getutxent()</i> , <i>getutxid()</i> , <i>getutxline()</i> , <i>pututxline()</i> , <i>setgrent()</i> , <i>setregid()</i> , <i>setreuid()</i> , <i>setutxent()</i>
XSI_WIDE_CHAR	<i>wcswidth()</i> , <i>wcwidth()</i>

- (1) There was a `_POSIX_JOB_CONTROL` option in an earlier version of the POSIX standards that specified these functions. All of these functions are mandatory in POSIX.1 {3}.
- (2) There was a `_POSIX_REGEX` option in an earlier version of the POSIX standards that specified these functions. All of these functions are mandatory in POSIX.1 {3}.
- (3) There was a `_POSIX_READER_WRITER_LOCKS` option in an earlier version of the POSIX standards that specified these functions. All of these functions are part of the `_POSIX_THREADS` option in POSIX.1 {3}.
- (4) Dependent on the `_POSIX_TIMEOUTS` option.
- (5) Dependent on the `_POSIX_THREAD_PROCESS_SHARED` option.
- (6) `POSIX_THREADS_BASE` is the same as the `_POSIX_THREADS` option, but without the functions belonging to the `POSIX_RW_LOCKS` Unit of Functionality.

Table 1-2 — POSIX.5 Units of Functionality (Ada Language Support)

POSIX_ADA_LANG_SUPPORT	
Package	Subprograms
System	Extra requirements specified in POSIX.5c {5}, Section 2.8.
System_Storage_Elements	All ⁽¹⁾
POSIX_Page_Alignment	All
POSIX_Supplement_To_Ada_IO	All
Ada_Task_Identification	All
Ada_Streams	All

- (1) *All* indicates all subprograms in a package are required to be supported. Where overloaded versions of a subprogram exist, each instance is required, except as noted. All Image and Value functions must be supported for all packages provided by the implementation.

Copyright © 2004 IEEE. All rights reserved.

Table 1-3 — POSIX.5 Units of Functionality (Device IO)

POSIX_DEVICE_IO	
Package	Subprograms
POSIX_IO	Open Close Read Write Generic_Read Generic_Write Is_Open

Table 1-4 — POSIX.5 Units of Functionality (Device Specific)

POSIX_DEVICE_SPECIFIC	
Package	Subprograms
POSIX_Terminal_Functions	Get_Terminal_Characteristics Get_Controlling_Terminal_Name Set_Terminal_Characteristics Terminal_Modes_Of Define_Terminal_Modes Bits_Per_Character_Of Define_Bits_Per_Character Special_Control_Character_Of Define_Special_Control_Character Disable_Control_Character Input_Time_Of Define_Input_Time Minimum_Input_Count_Of Define_Minimum_Input_Count Input_Baud_Rate_Of Output_Baud_Rate_Of Define_Input_Baud_Rate Define_Output_Baud_Rate Send_Break Drain Discard_Data Flow
POSIX_IO	Is_A_Terminal Get_Terminal_Name

Copyright © 2004 IEEE. All rights reserved.

Table 1-5 — POSIX.5 Units of Functionality (Event Management)

POSIX_EVENT_MGMT	
Package	Subprograms
POSIX_Event_Management ⁽¹⁾	Make_Empty Add Remove In_Set Select_File For_Every_File_In

⁽¹⁾ The subprograms listed in this table are those under the Select option in POSIX.5c {5}. But instead of using this option, a Unit of Functionality has been created because there is no equivalent option in POSIX.1 {3}.

Table 1-6 — POSIX.5 Units of Functionality (FD Management)

POSIX_FD_MGMT	
Package	Subprograms
POSIX_File_Locking	All
POSIX_IO	Duplicate Duplicate_And_Close Get_File_Control Set_File_Control Get_Close_On_Exec Set_Close_On_Exec Seek File_Size File_Position

Table 1-7 — POSIX.5 Units of Functionality (FIFO)

POSIX_FIFO	
Package	Subprograms
POSIX_Files	Create_FIFO

Table 1-8 — POSIX.5 Units of Functionality (File Attributes)

POSIX_FILE_ATTRIBUTES	
Package	Subprograms
POSIX_Permissions	Set_Allowed_Process_Permissions Get_Allowed_Process_Permissions
POSIX_Files	Change_Owner_And_Group Change_Permissions

Copyright © 2004 IEEE. All rights reserved.

Table 1-9 — POSIX.5 Units of Functionality (File System)

POSIX_FILE_SYSTEM	
Package	Subprograms
POSIX_Configurable_File_Limits	All
POSIX_File_Status	All
POSIX_Files	For_Every_Directory_Entry Create_Directory Unlink Remove_Directory Rename Accessibility Is_Accessible Existence Is_File_Present Set_File_Times Link Filename_Of Is_File Is_Directory Is_FIFO Is_Character_Special_File Is_Block_Special_File Is_Socket
POSIX_Process_Environment	Change_Working_Directory Get_Working_Directory
POSIX_IO	Open_Or_Create

Table 1-10 — POSIX.5 Units of Functionality (Job Control)

POSIX_JOB_CONTROL ⁽¹⁾	
Package	Subprograms
POSIX_Process_Identification	Set_Process_Group_Id Create_Process_Group
POSIX_Terminal_Functions	Get_Process_Group_Id Set_Process_Group_Id
POSIX_Signals	Set_Stopped_Child_Signal Stopped_Child_Signal_Enabled

⁽¹⁾ The subprograms listed in this table are those under the Job Control option in POSIX.5c {5}. But instead of using this option, a Unit of Functionality has been created because the equivalent option in POSIX.1 {3} does not specify the functions that fall under it.

Table 1-11 — POSIX.5 Units of Functionality (Multi-Process)

POSIX_MULTI_PROCESS	
Package	Subprograms
POSIX_Process_Primitives	All
POSIX_Unsafe_Process_Primitives	All
POSIX_Process_Times	All
POSIX_Process_Identification	Get_Process_Id Get_Parent_Process_Id

Table 1-12 — POSIX.5 Units of Functionality (Networking)

POSIX_NETWORKING	
Package	Subprograms
POSIX_IO	Get_Owner Set_Socket_Process_Owner Set_Socket_Group_Owner Set_Buffer Get_Buffer
POSIX_Sockets	All ⁽¹⁾
POSIX_Sockets_Local	All ^a
POSIX_Sockets_Internet	All ⁽²⁾

⁽¹⁾ The POSIX_Sockets and POSIX_Sockets_Local packages depend on the Sockets Detailed Network Interface option (and partly on the Network Management option) defined in POSIX.5c {5}, but they are included here because there are no equivalent options in POSIX.1 {3}.

⁽²⁾ The POSIX_Sockets_Internet package depends on the Sockets Detailed Network Interface option (and partly on the Internet Protocol, Internet Datagram, and Internet Stream options) defined in POSIX.5c {5}, but it is included here because there are no equivalent options in POSIX.1 {3}.

Table 1-13 — POSIX.5 Units of Functionality (Pipes)

POSIX_PIPES	
Package	Subprograms
POSIX_IO	Create_Pipe

Copyright © 2004 IEEE. All rights reserved.

Table 1-14 — POSIX.5 Units of Functionality (Signals)

POSIX_SIGNALS	
Package	Subprograms
POSIX_Signals	Add_Signal Add_All_Signals Delete_Signal Delete_All_Signals Is_Member Send_Signal Set_Blocked_Signals Block_Signals Unblock_Signals Blocked_Signals Ignore_Signal Unignore_Signal Is_Ignored Install_Empty_Handler Pending_Signals Await_Signal ⁽¹⁾ Await_Signal_Or_Timeout ^a Interrupt_Task Get_Signal ⁽²⁾ Set_Signal ^b Get_Notification Set_Notification Get_Data ^b Set_Data ^b

(1) Return type Signal .
 (2) Operation on type Signal_Event .

Table 1-15 — POSIX.5 Units of Functionality (Single Process)

POSIX_SINGLE_PROCESS	
Package	Subprograms
POSIX	All
POSIX_Limits	All
POSIX_Options	All
POSIX_Profiles	All ⁽¹⁾
POSIX_Configurable_System_Limits	All
POSIX_Calendar	All

Copyright © 2004 IEEE. All rights reserved.

Table 1-15 — POSIX.5 Units of Functionality (Single Process)

POSIX_SINGLE_PROCESS	
Package	Subprograms
POSIX_Process_Environment	Argument_List Copy_From_Current_Environment Copy_To_Current_Environment Copy_Environment Clear_Environment Set_Environment_Variable Delete_Environment_Variable Length For_Every_Environment_Variable For_Every_Current_Environment_Variable Environment_Value_Of Is_Environment_Variable

(1) The POSIX_Profiles package is defined in Annex A of this standard.

Table 1-16 — POSIX.5 Units of Functionality (System Database)

POSIX_SYSTEM_DATABASE	
Package	Subprograms
POSIX_Group_Database	All
POSIX_User_Database	All

Table 1-17 — POSIX.5 Units of Functionality (User Groups)

POSIX_USER_GROUPS	
Package	Subprograms
POSIX_Process_Identification	Get_Real_User_ID Get_Effective_User_ID Get_Real_Group_ID Get_Effective_Group_ID Set_User_ID Create_Session Set_Group_ID Get_Groups Get_Login_Name Get_Process_Group_ID

Copyright © 2004 IEEE. All rights reserved.

1.6 Summary of Profile Features

Table 1-18 through Table 1-20 summarize the requirements of the four profiles using an X character to represent a required item and a short dash (–) to represent an item that is not required. Since POSIX.1 {3} and/or POSIX.5c {5} does not provide sufficient options to remove features unnecessary for some profiles, Units of Functionality have been developed and are described in Table 1-1 through Table 1-17.

Table 1-18 — Units of Functionality Requirements

Unit of Functionality	PSE51	PSE52	PSE53	PSE54
POSIX_ADA_LANG_SUPPORT ⁽¹⁾	X	X	X	X
POSIX_C_LANG_JUMP ⁽²⁾	X	X	X	X
POSIX_C_LANG_MATH ^b	–	X	X	X
POSIX_C_LANG_SUPPORT ^b	X	X	X	X
POSIX_C_LANG_WIDE_CHAR ^b	–	–	–	X
POSIX_DEVICE_IO	X	X	X	X
POSIX_DEVICE_SPECIFIC	–	–	–	X
POSIX_EVENT_MGMT	–	–	X	X
POSIX_FD_MGMT	–	X	X	X
POSIX_FIFO	–	–	–	X
POSIX_FILE_ATTRIBUTES	–	–	–	X
POSIX_FILE_LOCKING ^b	X	X	X	X
POSIX_FILE_SYSTEM	–	X	X	X
POSIX_FILE_SYSTEM_EXT ^b	–	–	–	X
POSIX_JOB_CONTROL	–	–	–	X
POSIX_MULTI_PROCESS	–	–	X	X
POSIX_NETWORKING	–	–	X	X
POSIX_PIPE	–	–	X	X
POSIX_REGEX ^b	–	–	–	X
POSIX_RW_LOCKS ^b	–	–	–	–
POSIX_SHELL_FUNC ^b	–	–	–	X
POSIX_SIGNALS	X	X	X	X
POSIX_SIGNAL_JUMP ^b	–	–	X	X
POSIX_SINGLE_PROCESS	X	X	X	X
POSIX_STRING_MATCHING ^b	–	–	–	X
POSIX_SYMBOLIC_LINKS ^b	–	–	–	X
POSIX_SYSTEM_DATABASE	–	–	–	X
POSIX_THREADS_BASE ^b	X	X	X	X
POSIX_USER_GROUPS	–	–	–	X
POSIX_WIDE_CHAR_IO ^b	–	–	–	X

Copyright © 2004 IEEE. All rights reserved.

Table 1-18 — Units of Functionality Requirements (Continued)

Unit of Functionality	PSE51	PSE52	PSE53	PSE54
XSI_C_LANG_SUPPORT ^b	—	—	—	—
XSI_DBM ^b	—	—	—	—
XSI_DEVICE_IO ^b	—	—	—	—
XSI_DEVICE_SPECIFIC ^b	—	—	—	—
XSI_DYNAMIC_LINKING ^b	—	—	—	X
XSI_FD_MGMT ^b	—	—	—	—
XSI_FILE_SYSTEM ^b	—	—	—	—
XSI_I18N ^b	—	—	—	—
XSI_IPC ^b	—	—	—	—
XSI_JOB_CONTROL ^b	—	—	—	—
XSI_JUMP ^b	—	—	—	—
XSI_MATH ^b	—	—	—	—
XSI_MULTI_PROCESS ^b	—	—	—	—
XSI_SIGNALS ^b	—	—	—	—
XSI_SINGLE_PROCESS ^b	—	—	—	—
XSI_SYSTEM_DATABASE ^b	—	—	—	—
XSI_SYSTEM_LOGGING ^b	—	—	—	X
XSI_THREAD_MUTEX_EXT ^b	X	X	X	X
XSI_THREADS_EXT ^b	X	X	X	X
XSI_TIMERS ^b	—	—	—	—
XSI_USER_GROUPS ^b	—	—	—	—
XSI_WIDE_CHAR ^b	—	—	—	—

(1) Required only for the Ada language option.

(2) Required only for the C language option.

Table 1-19 — POSIX.1 Option Requirements

Option	PSE51	PSE52	PSE53	PSE54
_POSIX_ADVISORY_INFO	—	—	—	X
_POSIX_ASYNCHRONOUS_IO	—	—	X	X
_POSIX_BARRIERS	—	—	—	—
_POSIX_CHOWN_RESTRICTED	—	—	—	X
_POSIX_CLOCK_SELECTION	X	X	X	X
_POSIX_CPUTIME	—	—	X	X
_POSIX_FSYNC	X	X	X	X
_POSIX_IPV6	—	—	—	—
_POSIX_MAPPED_FILES	—	X	X	X
_POSIX_MEMLOCK	X	X	X	X
_POSIX_MEMLOCK_RANGE	X	X	X	X

Copyright © 2004 IEEE. All rights reserved.

Table 1-19 — POSIX.1 Option Requirements (Continued)

Option	PSE51	PSE52	PSE53	PSE54
_POSIX_MEMORY_PROTECTION	—	—	X	X
_POSIX_MESSAGE_PASSING	—	X	X	X
_POSIX_MONOTONIC_CLOCK	X	X	X	X
_POSIX_NO_TRUNC	X	X	X	X
_POSIX_PRIORITIZED_IO	—	—	X	X
_POSIX_PRIORITY_SCHEDULING	—	—	X	X
_POSIX_RAW_SOCKETS	—	—	X	X
_POSIX_REALTIME_SIGNALS	X	X	X	X
_POSIX_SAVED_IDS	—	—	—	X
_POSIX_SEMAPHORES	X	X	X	X
_POSIX_SHARED_MEMORY_OBJECTS	X	X	X	X
_POSIX_SPAWN	—	—	X	X
_POSIX_SPIN_LOCKS	—	—	—	—
_POSIX_SPORADIC_SERVER	—	—	X	X
_POSIX_SYNCHRONIZED_IO	X	X	X	X
_POSIX_THREAD_ATTR_STACKADDR	X	X	X	X
_POSIX_THREAD_ATTR_STACKSIZE	X	X	X	X
_POSIX_THREAD_CPU_TIME	X	X	X	X
_POSIX_THREAD_PRIO_INHERIT	X	X	X	X
_POSIX_THREAD_PRIO_PROTECT	X	X	X	X
_POSIX_THREAD_PRIORITY_SCHEDULING	X	X	X	X
_POSIX_THREAD_PROCESS_SHARED	—	—	X	X
_POSIX_THREAD_SAFE_FUNCTIONS	See Units of Functionality			
_POSIX_THREAD_SPARADIC_SERVER	X	X	X	X
_POSIX_THREADS	See Units of Functionality			
_POSIX_TIMEOUTS	X	X	X	X
_POSIX_TIMERS	X	X	X	X
_POSIX_TRACE	—	X	X	X
_POSIX_TRACE_EVENT_FILTER	—	X	X	X
_POSIX_TRACE_INHERIT	—	—	—	—
_POSIX_TRACE_LOG	—	X	X	X
_POSIX_TYPED_MEMORY_OBJECTS	—	—	—	—
_POSIX_VDISABLE	—	—	—	X
_POSIX2_C_DEV ⁽¹⁾	X ^b	X ^b	X ^b	X
_POSIX2_CHAR_TERM	—	—	—	X
_POSIX2_FORT_DEV	—	—	—	—
_POSIX2_FORT_RUN	—	—	—	X
_POSIX2_LOCALEDEF	—	—	—	—
_POSIX2_PBS	—	—	—	—
_POSIX2_PBS_ACCOUNTING	—	—	—	—
_POSIX2_PBS_CHECKPOINT	—	—	—	—

Copyright © 2004 IEEE. All rights reserved.

Table 1-19 — POSIX.1 Option Requirements (Continued)

Option	PSE51	PSE52	PSE53	PSE54
_POSIX2_PBS_LOCATE	—	—	—	—
_POSIX2_PBS_MESSAGE	—	—	—	—
_POSIX2_PBS_TRACK	—	—	—	—
_POSIX2_SW_DEV	X ⁽²⁾	X ^b	X ^b	X
_POSIX2_UPE	—	—	—	X
_XOPEN_CRYPT	—	—	—	—
_XOPEN_ENH_I18N	No interfaces fall under this option			
_XOPEN_LEGACY	—	—	—	—
_XOPEN_REALTIME	See individual suboptions			
_XOPEN_REALTIME_THREADS	See individual suboptions			
_XOPEN_SHM	No interfaces fall under this option			
_XOPEN_STREAMS	—	—	—	—
_XOPEN_UNIX	See Units of Functionality			

(1) Required only for the C language option.

(2) Required only for the development platform, which will often differ from the execution platform.

Table 1-20 — Requirements for Other Standards

Standard	PSE51	PSE52	PSE53	PSE54
POSIX.26 {4}	X	X	X	X

The correspondence between the options listed in Table 1-19 and the options described in POSIX.5c {5}, Section 2.5, are shown in Table 1-21.

Table 1-21 — POSIX.1 Options vs. POSIX.5c Options

POSIX.1 Option	POSIX.5c Option
_POSIX_ADVISORY_INFO	none
_POSIX_ASYNCHRONOUS_IO	Asynchronous I/O
_POSIX_BARRIERS	none
_POSIX_CHOWN_RESTRICTED	Change Owner Restriction
_POSIX_CLOCK_SELECTION	none
_POSIX_CPUTIME	none
_POSIX_FSYNC	File Synchronization
_POSIX_IPV6	none
_POSIX_MAPPED_FILES	Memory Mapped Files
_POSIX_MEMLOCK	Memory Locking
_POSIX_MEMLOCK_RANGE	Memory Range Locking
_POSIX_MEMORY_PROTECTION	Memory Protection
_POSIX_MESSAGE_PASSING	Message Queues
_POSIX_MONOTONIC_CLOCK	none

Copyright © 2004 IEEE. All rights reserved.

Table 1-21 — POSIX.1 Options vs. POSIX.5c Options (Continued)

POSIX.1 Option	POSIX.5c Option
_POSIX_NO_TRUNC	Filename Truncation ⁽¹⁾
_POSIX_PRIORITIZED_IO	Prioritized I/O
_POSIX_PRIORITY_SCHEDULING	Priority Process Scheduling
_POSIX_RAW_SOCKETS	none
_POSIX_REALTIME_SIGNALS	Realtime Signals
_POSIX_SAVED_IDS	Saved IDs Support
_POSIX_SEMAPHORES	Semaphores
_POSIX_SHARED_MEMORY_OBJECTS	Shared Memory Objects
_POSIX_SPAWN	C language-specific
_POSIX_SPIN_LOCKS	none
_POSIX_SPORADIC_SERVER	none
_POSIX_SYNCHRONIZED_IO	Synchronized I/O
_POSIX_THREAD_ATTR_STACKADDR	C language-specific
_POSIX_THREAD_ATTR_STACKSIZE	C language-specific
_POSIX_THREAD_CPU_TIME	none
_POSIX_THREAD_PRIO_INHERIT	Mutex Priority Inheritance
_POSIX_THREAD_PRIO_PROTECT	Mutex Priority Ceiling
_POSIX_THREAD_PRIORITY_SCHEDULING	C language-specific
_POSIX_THREAD_PROCESS_SHARED	Process Shared
_POSIX_THREAD_SAFE_FUNCTIONS	C language-specific
_POSIX_THREAD_SPAWNSERVER	none
_POSIX_THREADS	C language-specific
_POSIX_TIMEOUTS	none
_POSIX_TIMERS	Timers
_POSIX_TRACE	none
_POSIX_TRACE_EVENT_FILTER	none
_POSIX_TRACE_INHERIT	none
_POSIX_TRACE_LOG	none
_POSIX_TYPED_MEMORY_OBJECTS	none
_POSIX_VDISABLE	C language-specific
_POSIX2_C_DEV	not applicable
_POSIX2_CHAR_TERM	not applicable
_POSIX2_FORT_DEV	not applicable
_POSIX2_FORT_RUN	not applicable
_POSIX2_LOCALEDEF	not applicable
_POSIX2_PBS	not applicable
_POSIX2_PBS_ACCOUNTING	not applicable
_POSIX2_PBS_CHECKPOINT	not applicable
_POSIX2_PBS_LOCATE	not applicable
_POSIX2_PBS_MESSAGE	not applicable
_POSIX2_PBS_TRACK	not applicable
_POSIX2_SW_DEV	not applicable
_POSIX2_UPE	not applicable

Copyright © 2004 IEEE. All rights reserved.

Table 1-21 — POSIX.1 Options vs. POSIX.5c Options (Continued)

POSIX.1 Option	POSIX.5c Option
<code>_XOPEN_CRYPT</code>	none
<code>_XOPEN_ENH_I18N</code>	none
<code>_XOPEN_LEGACY</code>	none
<code>_XOPEN_REALTIME</code>	none
<code>_XOPEN_REALTIME_THREADS</code>	none
<code>_XOPEN_SHM</code>	none
<code>_XOPEN_STREAMS</code>	none
<code>_XOPEN_UNIX</code>	none
Ada language-specific (mutexes are included under the <code>_POSIX_THREADS</code> option)	Mutexes

⁽¹⁾ Note that the POSIX.5c Filename Truncation option has the opposite sense relative to the POSIX.1 option `_POSIX_NO_TRUNC`.

In all profiles that do not support the `POSIX_JOB_CONTROL` Unit of Functionality, the subprogram `POSIX_Signals.Set_Stopped_Child_Signal` shall fail silently.

In all profiles that do not support the `POSIX_JOB_CONTROL` Unit of Functionality, the subprogram `POSIX_Signals.Stopped_Child_Signal_Enabled` shall return `False`.

`POSIX_Limits.Groups_Maxima'First` shall be zero for PSE51, PSE52, and PSE53. For PSE54 it shall be greater than or equal to eight.

`POSIX_Terminal_Functions.Disable_Control_Character` (which corresponds to `_POSIX_VDISABLE`) is not supported in PSE51, PSE52, and PSE53. For PSE54, `POSIX_Terminal_Functions.Disable_Control_Character` shall not raise `POSIX_Error` with an error code of `Operation_Not_Implemented`.

For PSE51 and PSE52, the blocking behavior of all reentrant operations defined by POSIX.5c {5} shall be per task, i.e., a blocked task cannot prevent any other task from executing. Therefore, the corresponding `Blocking_Behavior` constants shall have the value `Tasks`. (See POSIX.5c {5}, Section 2.4.1.5.)

Copyright © 2004 IEEE. All rights reserved.

Section 2: Normative References

2.1 Normative References

The following standards contain provisions which, through references in this text, constitute provisions of this standard.¹⁾ At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this profile of IEEE and ISO are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

- {1} ISO/IEC 8652:1995, *Information technology—Programming Languages—Ada*.²⁾
- {2} ISO/IEC 9899:1999, *Programming Languages—C*.
- {3} ISO/IEC 9945:2003, *Information technology—Portable Operating System Interface (POSIX™)*.³⁾
- {4} IEEE Std 1003.26-2003, *IEEE Standard for Information Technology—Portable Operating System Interface (POSIX™)—Part 26: Device Control Application Program Interface (API) [C Language]*.⁴⁾
- {5} ISO/IEC 14519:2001, *Information technology—POSIX™ Ada Language Interfaces—Binding for System Application Program Interface (API)*.
- {6} ISO/IEC TR 10000-1:1998, *Information technology—Framework and taxonomy of International Standardized Profiles—Part 1: General principles and documentation framework*.
- {7} ISO/IEC TR 10000-3:1998, *Information technology—Framework and Taxonomy of International Standardized Profiles—Part 3: Principles and Taxonomy for Open System Environment Profiles*.

1) Other references to related standards and other documents can be found in Annex C of this standard. Common names for these standards can be found in 4.2.

2) ISO/IEC documents can be obtained from the ISO office, 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iso.ch/>) and from the IEC office, 3 rue de Varembé, Case Postale 131, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iec.ch/>). ISO/IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

3) Identical to IEEE Std 1003.1™-2003.

4) IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (<http://standards.ieee.org/>).

Copyright © 2004 IEEE. All rights reserved.

This page is left intentionally blank.

Section 3: Terms and Definitions

3.1 Terminology

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

3.1.1 implementation defined: Describes a value or behavior that is not defined by the standard, but is selected by an implementor. The value or behavior may vary among implementations that conform to POSIX.13. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations.

The implementor shall document such a value or behavior in the conformance document, so that it can be used correctly by an application.

3.1.2 may: Describes a feature or behavior that is optional for an implementation that conforms to POSIX.13. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

To avoid ambiguity, the opposite of *may* is expressed as *need not*, instead of *may not*.

3.1.3 shall: For an implementation that conforms to POSIX.13, describes a feature or behavior that is mandatory. An application can rely on the existence of the feature or behavior.

For an application or user, describes a behavior that is mandatory.

3.1.4 should: For an implementation that conforms to POSIX.13, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

Copyright © 2004 IEEE. All rights reserved.