

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

Sponsor

Portable Applications Standards Committee
of the
IEEE Computer Society

Unapproved draft

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, open systems, operating system, portable application, POSIX profiles, realtime application environments, realtime, embedded

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

IEEE P1003.13/D2.1 February 2003

Copyright © 2003 by the Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue
New York, NY 10016-5997, USA
All rights reserved.

This document is an unapproved draft of a proposed IEEE Standard. As such, this document is subject to change. USE AT YOUR OWN RISK. Because this is an unapproved draft, this document must not be utilized for any conformance/compliance purposes. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only. Prior to submitting this document to another standard development organization for standardization activities, permission must first be obtained from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department. Other entities seeking permission to reproduce this document, in whole or in part, must obtain permission from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department.

IEEE Standards Activities Department
Standards Licensing and Contracts
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA

+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

3/2/03

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 *Editor's Notes*

2
3 This section will not appear in the final document. It is used for editorial comments
4 concerning this draft. Please consult the balloting instructions document and the
5 cover letter for the ballot that accompanied this draft for information on how the
6 balloting process is accomplished.
7

8 This is the first draft of the POSIX.13 revision; POSIX.13-1998 defined four real-
9 time application environment profiles (or POSIX subsets), based on the ISO/IEC
10 9945-1:1996 (POSIX.1) and the IEEE Std 1003.5c-1998 (POSIX.5c) standards. The
11 goal of this revision is to update the profiles according to implementation experi-
12 ence, and to add the services defined in the new revised IEEE Std 1003.1-2001
13 (which incorporates among other services the recently approved POSIX amend-
14 ments POSIX.1d, POSIX.1g, POSIX.1j, and POSIX.1q) and the POSIX.5c amend-
15 ment. Also in the scope is to incorporate any new POSIX Ada bindings that might
16 get developed and approved before the completion of this revision. The POSIX.13
17 revision project incorporates and supersedes work developed previously in the
18 POSIX.13a and POSIX.13b projects

19 Changes to the previous standard have been marked with side bars like that af-
20 fecting this sentence. These side bars are for information only. Small numbers are
21 printed at the left margin of each page of the document to ease making references
22 to specific text during the ballot process. These numbers may not match actual
23 lines, and are only used as an approximate reference.
24

25 Please report typographical errors to:

26
27 Michael González Harbour
28 Dpto. de Electrónica y Computadores
29 Universidad de Cantabria
30 Avenida de los Castros s/n
31 39005 - Santander SPAIN
32 TEL: +34 942 201483
33 FAX: +34 942 201402
34 Email: mgh@unican.es (*Electronic mail is preferred.*)
35

36 The copying and distribution of IEEE balloting drafts is accomplished by the Stan-
37 dards Office. To report problems with reproduction of your copy, or to request ad-
38 ditional copies of this draft, contact:

39
40 Tracy Woods
41 IEEE Computer Society,
42 1730 Massachusetts Avenue, NW,
43 Washington DC 20036-1992, USA.
44 Phone: +1-202-371-1013
45 Fax: +1-202-728-0884
46 E-mail: twoods@computer.org
47 Web page: <http://www.computer.org/standard/draftstd.htm>
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

POSIX.13 Change History

This section is provided to track major changes between drafts.

Draft 2 [July 2002] First ballot draft

Draft 1.2 [July 2002] First complete draft, for internal SSWG-RT use.

- Added new limits
- Eliminated requirement for reader/writer locks
- Added Annex B
- Added the alphabetical topical index
- Added requirement for priority ranges

Draft 1.1 [April 2002] Second draft for internal SSWG-RT use.

- Minor fixes and additions
- Incorporated some changes from discussions at the Open Group's Real-Time Forum.

Draft 1.0 [February 2002] First draft, incomplete, for internal SSWG-RT use.

- Scope of P1003.1a and P1003.1b (amendments to IEEE Std 1003.13-1998) included in this revision
- Targets the newly approved POSIX.1 revision (IEEE Std 1003.1-2001) and POSIX.5c
- Uses text developed for POSIX.13b, but reformatted as a revision to POSIX.13
- Standard's title changed to add support for embedded applications

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Contents

Introduction	xi
Section 1: Overview	1
1.1 Scope	1
1.2 Taxonomy Position	1
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	16
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	29
Section 4: Conventions and Abbreviations	31
4.1 Conventions	31
4.2 Abbreviations	32
Section 5: Conformance	35
5.1 Conformance	35
5.1.1 Implementation Conformance	35
5.1.2 Application Conformance	36
Section 6: Minimal Realtime System Profile (PSE51)	39
6.1 Introduction	39
6.1.1 Identification	39
6.1.2 Conformance	39
6.1.3 Options	40
6.2 Operating System Interface Requirements	40
6.2.1 POSIX.1 Requirements (C Language Option)	40
6.2.2 POSIX.5c Requirements (Ada Language Option)	42
6.3 Application Constraints	43
6.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)	43
6.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)	43

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1	45
2	6.4 Shell and Utility Requirements.....	46
3	6.5 Development Platform Requirements.....	46
4	6.5.1 C Language Development Option.....	46
5	6.5.2 Ada Language Development Option.....	47
6	6.6 Rationale for Operating System Requirements (informative).....	47
7	6.6.1 Operating System Interface Requirements.....	47
8	6.6.2 Shell and Utility Requirements.....	55
9	6.6.3 Development Platform Requirements.....	55
10		
11	Section 7: Realtime Controller System Profile (PSE52).....	57
12	7.1 Introduction.....	57
13	7.1.1 Identification.....	57
14	7.1.2 Conformance.....	57
15	7.1.3 Options.....	58
16	7.2 Operating System Interface Requirements.....	58
17	7.2.1 POSIX.1 Requirements (C language Option).....	58
18	7.2.2 POSIX.5c Requirements (Ada Language Option).....	60
19	7.3 Application Constraints.....	61
20	7.3.1 Constraints related to POSIX.1 Interfaces (C Language Option) ..	61
21	7.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)	
22	62
23	7.4 Shell and Utility Requirements.....	63
24	7.5 Development Platform Requirements.....	63
25	7.5.1 C Language Development Option.....	63
26	7.5.2 Ada Language Development Option.....	64
27	7.6 Rationale for Operating System Requirements (informative).....	64
28	7.6.1 Operating System Interface Requirements.....	64
29	7.6.2 Shell and Utility Requirements.....	72
30	7.6.3 Development Platform Requirements.....	73
31		
32	Section 8: Dedicated Realtime System Profile (PSE53).....	75
33	8.1 Introduction.....	75
34	8.1.1 Identification.....	75
35	8.1.2 Conformance.....	75
36	8.1.3 Options.....	76
37	8.2 Operating System Interface Requirements.....	76
38	8.2.1 POSIX.1 Requirements (C Language Option).....	76
39	8.2.2 POSIX.5c Requirements (Ada Language Option).....	78
40	8.3 Application Constraints.....	80
41	8.3.1 Constraints related to POSIX.1 Interfaces (C Language Option) ..	80
42	8.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)	
43	80
44	8.4 Shell and Utility Requirements.....	81
45	8.5 Development Platform Requirements.....	81
46	8.5.1 C Language Development Option.....	81
47	8.5.2 Ada Language Development Option.....	82
48	8.6 Rationale for Operating System Requirements (informative).....	82
49	8.6.1 Operating System Interface Requirements.....	82

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 8.6.2 Shell and Utility Requirements 91
2 8.6.3 Development Platform Requirements 91
3
4 Section 9: Multi-Purpose Realtime System Profile (PSE54) 93
5 9.1 Introduction. 93
6 9.1.1 Identification. 93
7 9.1.2 Conformance 93
8 9.1.3 Options 94
9 9.2 Operating System Interface Requirements. 94
10 9.2.1 POSIX.1 Requirements (C Language Option). 94
11 9.2.2 POSIX.5c Requirements (Ada Language Option). 97
12 9.3 Application Constraints 99
13 9.3.1 Constraints related to POSIX.1 Interfaces (C Language Option) . . 99
14 9.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)
15 99
16 9.4 Shell and Utility Requirements 99
17 9.5 Development Platform Requirements 100
18 9.5.1 C Language Development Option 100
19 9.5.2 Ada Language Development Option 100
20 9.6 Rationale for Operating System Requirements (informative) 101
21 9.6.1 Operating System Interface Requirements. 101
22 9.6.2 Shell and Utility Requirements 109
23 9.6.3 Development Platform Requirements 109
24
25 Annex A: POSIX Profiles Package (Ada Language). 111
26
27 Annex B: Description of Optional Interfaces 113
28 B.1 POSIX.1 Options. 113
29 B.2 POSIX.5c Options 123
30
31 Annex C: Bibliography. 127
32 C.1 Related Open Systems Standards 127
33 C.2 Other Documents 127
34
35 Alphabetic Topical Index. 129
36
37
38
39
40
41
42
43
44
45
46
47
48
49

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

List of Figures

Figure I.1. Main Building Blocks of the Profiles xix

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

List of Tables

Table 1-1: POSIX.1 Units of Functionality	4
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)	11
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)	12
Table 1-10: POSIX.5 Units of Functionality (Job Control).	13
Table 1-11: POSIX.5 Units of Functionality (Multi-Process).	13
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 (Priority Ranges)	14
Table 1-15: POSIX.5 Units of Functionality (Signals)	15
Table 1-16: POSIX.5 Units of Functionality (Single Process)	15
Table 1-17: POSIX.5 Units of Functionality (System Database)	16
Table 1-18: POSIX.5 Units of Functionality (User Groups).	16
Table 1-19: Units of Functionality Requirements	17
Table 1-20: POSIX.1 Option Requirements	18
Table 1-21: POSIX.1 Options vs. POSIX.5c Options	20
Table 6-1: POSIX.1 Units of Functionality Requirements	40
Table 6-2: POSIX.1 Option Requirements	41
Table 6-3: POSIX.5c Units of Functionality Requirements	42
Table 6-4: POSIX.5c Option Requirements	42
Table 6-5: Functions required to be async-signal-safe	44
Table 7-1: POSIX.1 Units of Functionality Requirements	58
Table 7-2: POSIX.1 Option Requirements	59
Table 7-3: POSIX.5c Units of Functionality Requirements	60
Table 7-4: POSIX.5c Option Requirements	60
Table 7-5: Functions required to be async-signal-safe	62
Table 8-1: POSIX.1 Units of Functionality Requirements	76
Table 8-2: POSIX.1 Option Requirements	77
Table 8-3: POSIX.5c Units of Functionality Requirements	78
Table 8-4: POSIX.5c Option Requirements	79
Table 9-1: POSIX.1 Units of Functionality Requirements	94
Table 9-2: POSIX.1 Option Requirements	95
Table 9-3: POSIX.1 Units of Functionality Requirements	97

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1	Table 9-4: POSIX.5c Option Requirements	97
2	Table 9-5: Shell and Utilities Option Requirements	
3	(C Language Option)	99
4	Table 9-6: Shell and Utilities Option Requirements	
5	(Ada Language Option)	100
6	Table B-1: Functions under each POSIX.1	
7	System Interface Option	113
8	Table B-2: Utilities under each POSIX.1 Shell and Utilities Option	121
9	Table B-3: Packages and Subprograms under	
10	each POSIX.5c Option.	123
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
41		
42		
43		
44		
45		
46		
47		
48		
49		

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Introduction

(This introduction is not a normative part of IEEE Std P1003.13, Information technology—Standardized Application Environment Profile—POSIX Realtime and Embedded Application Support (AEP)).

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 newly approved POSIX standards:

- IEEE Std 1003.1-2001, Standard for *Information Technology—Portable Operating System Interface (POSIX)* (which includes among others the revised ISO/IEC 9945-1: 1996 as amended by IEEE Std 1003.1d-1999, IEEE Std 1003.1j-2000, IEEE Std 1003.1g-2000, and IEEE Std 1003.1q-2000)
- and the amendment to IEEE Std 1003.5-1992, *IEEE Standard for Information Technology—POSIX Ada Language Interfaces—Part 1: Binding for System Application Programming Interface (API)* and IEEE Std 1003.5b-1996, *IEEE Standard for Information Technology—POSIX Ada Language Interfaces—Binding for System Application Program Interface (API)—Amendment 1: Realtime Extensions*; this amendment is: IEEE Std 1003.5c-1998.

The base standard, IEEE Std 1003.1-2001, 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, Section 2.1.5.1, “Subprofiling Considerations”). The POSIX.13 standard articulates these subprofiling options through units of functionality, defined herein, and by use of named options defined in the base standard.

This standard specifies four realtime profiles both for the C Language and for the Ada Language options. Because Ada Bindings to IEEE Std 1003.1-2001 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 1003.13 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

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 bottom-level device controllers use the “minimal” profile, the next level up follows
2 the larger “control” profile, and so on. There are interfaces called out for the small-
3 er profiles that make no sense in an isolated box; those interfaces are there solely
4 to support the construction of heterogeneous systems, and systems of communicat-
5 ing peers. Such systems are very common in practice.

6
7 To summarize, this standard is embedded in a much larger and widely supported
8 set of standards, which yields benefits during code development, as much develop-
9 ment and testing is done on the larger and more comfortable systems. It also may
10 be used in the construction of large and heterogeneous systems.

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

19
20 Initially, the focus of this standard is to provide standardized environments sup-
21 porting the C language. Options are provided for bindings to the Ada programming
22 language as well as for the C language. Bindings for other languages to these ser-
23 vices may be developed and this standard will be updated as appropriate.

24 Within this document, the term “POSIX.13” refers to this standard, IEEE Std
25 1003.13-200x.

26
27 *Editor's note: 200x will be changed to match the year the 1003.13*
28 *revision is approved as a standard.*
29

30 31 **Organization of This Standard**

32
33
34 This Standard is divided into eight elements:

- 35 (1) General (Section 1)
- 36 (2) Normative references (Section 2)
- 37 (3) Definitions (Section 3)
- 38 (4) Conventions and abbreviations (Section 4)
- 39 (5) Conformance (Section 5)
- 40 (6) The various realtime profiles (Sections 6 through 9)
- 41 (7) ISPICS requirements (C) (Annex A)
- 42 (8) ISPICS requirements (Ada) (Annex B)

43
44
45
46
47
48
49 References are provided to direct the reader to other related sections.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

3
4 In publishing this Standard, its developers simply intend to provide a yardstick
5 against which various operating system implementations can be measured for con-
6 formance. It is not the intent of the developers to measure or rate any products, to
7 reward or sanction any vendors of products for conformance or lack of conformance
8 to this Standard, or to attempt to enforce this Standard by these or any other
9 means. The responsibility for determining the degree of conformance or lack there-
10 of with this Standard rests solely with the individual who is evaluating a product
11 claiming to be in conformance with this Standard.

12 13 **Base Documents**

14
15
16 The various realtime application environments described herein are based on the
17 ISO/IEC 9945 and IEEE 1003 family of documents as well as ISO 9899 (C99 Lan-
18 guage) and 8652 (Ada95 Language).

19 20 21 **Scenario**

22
23 This standard is based directly on existing small and/or realtime (typically non-
24 UNIX^{TM1}) kernel practice as well as the growing body of practice with POSIX con-
25 formant kernels having realtime features. The general approach taken in this
26 standard is to specify interfaces (taken from POSIX) sufficient to deliver the func-
27 tionality typical of current realtime systems, (see Table 1-19 through Table 1-21).

28
29 Each profile is specified with full features, to give users clear direction. Vendors
30 may provide means to configure out those parts that are not needed by specific ap-
31 plications. Vendors wishing to expand on the specified profiles are strongly encour-
32 aged to take the added interfaces from other POSIX.13 profiles or from the base
33 standards, rather than invent new interfaces.

34
35 For each profile, the minimum hardware typically required is specified. This is the
36 hardware assumed to be present; implementations may, of course, have more, but
37 nothing in the profile requires—either directly or indirectly—more than the spec-
38 ified minimum hardware model.

39 40 41 **Audience**

42
43 The intended audience for this class of profiles is all persons concerned with an in-
44 dustry-wide standard realtime application environment based on the POSIX suite
45 of standards. This includes at least four groups of people:

-
- 46
47 1. UNIX is a registered trademark of The Open Group in the United States of America and
48 other countries.

49

- 1 (1) Persons buying hardware and software systems.
- 2
- 3 (2) Persons managing companies that are deciding on future corporate comput-
4 ing directions.
- 5 (3) Persons implementing realtime operating systems.
- 6
- 7 (4) Persons developing realtime applications where portability is a primary ob-
8 jective.
- 9

10 Rationale on Background

11
12
13 *This subclause contains rationale common to all four realtime profiles.*

14
15 The developers of POSIX.13 represent a cross section of hardware manufacturers,
16 vendors of operating systems and other software development tools, software de-
17 signers, consultants, academics, authors, applications programmers, and others.
18 In the course of their deliberations, the developers reviewed related U.S. and in-
19 ternational standards, both published and in progress.

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

22
23 The developers of this standard have tried to capture the functionality of existing
24 realtime systems in a reasonable number of profiles that specify predominate ap-
25 plication environments. It is felt that these profiles, although not optimum, are a
26 best fit to existing classes of applications and systems.

27
28 Features of several commercial realtime kernels were considered during the devel-
29 opment of the 1998 version of POSIX.13. These included **pSOS**^{TM1}, **VRTX32**^{TM2},
30 and **VxWorks**^{TM3}. Since these products were commercially successful, they must
31 have addressed a significant market segment. In addition, the uniprocessor subset
32 of VITA's **ORKID** specification, NGCR's "**Tiny Real Time**" (TRT), and the
33 **uITRON** specification were examined. These were all proposed standard interfac-
34 es for small realtime embedded systems.

35 Features of other commercial realtime kernels such as **RT-Linux**⁴ and **QNX**⁵, as
36 well as free software products such as **RTEMS**⁶ were considered during the devel-
37 opment of the current revision of POSIX.13.

38
39 The following is a list of features that are representative of current realtime sys-
40 tems and highlights the range of system requirements. While some concepts are

-
- 41
 - 42 1. **pSOS** is now a registered trademark of Wind River Systems, Inc.
 - 43 2. **VRTX32** is now a registered trademark of Mentor Graphics.
 - 44 3. **VxWorks** is a registered trademark of Wind River Systems, Inc.
 - 45 4. **RT-Linux** is... FSM Labs
 - 46 5. **QNX** is ...
 - 47 6. **RTEMS** is ...
 - 48
 - 49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 common to virtually all implementations (e.g., preemptive, priority-based schedul-
 2 ing), some only apply to smaller systems (e.g., a single address space), and some
 3 only to more full-featured systems (e.g., network support, self-hosting).
 4

5 **Basic Realtime Multitasking and Synchronization**

- 6 — Multiple flows of control
- 7 — Preemptive priority scheduling of flows of control
- 8 — One address space for all flows of control
- 9 — Direct control of location of memory areas
- 10 — Inter-thread communications mechanism via message passing
- 11 (queues)
- 12 — Binary and counting semaphores, without priority inheritance
- 13 — Mutual exclusion, with optional priority inheritance or priority ceiling
- 14 protocols
- 15 — Local or global event flags (one thread awaits multiple things)
- 16 — Multiple memory areas, with both fixed- and variable-sized block allo-
- 17 cation policies
- 18 — System time in units of clock ticks
- 19 — Timeouts on all blocking services in units of clock ticks
- 20 — Hardware interrupt control and support for user interrupt handlers
- 21 — Signals
- 22 — Exception handling
- 23 — Minimal synchronous I/O interface: *open()*, *close()*, *read()*, *write()*, *ioctl()*
- 24 — Debugger interface
- 25 — No memory protection
- 26 — Application runs in privileged (supervisor) mode, if applicable
- 27 — Direct I/O, rather than via kernel
- 28 — System executable size and memory requirements are major con-
- 29 straints

30 **I/O**

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

- 32 — Named I/O devices

33 Copyright © 2003 IEEE. All rights reserved.

34 This is an unapproved IEEE Standards Draft, subject to change.

- 1 — Support for serial I/O lines
- 2
- 3 — Pipes
- 4
- 5 — Installable user device drivers
- 6
- 7 — Memory mapped I/O

8 **Local File System**

9
10 Realtime systems supporting a file system generally provide the following
11 features:

- 12 — Named files
- 13
- 14 — Hierarchical filesystem (directories)
- 15
- 16 — Contiguous preallocation of disk space
- 17
- 18 — May provide media compatibility with another filesystem (e.g.
19 **MSDOS**^{TM1}, or **RT-11**^{TM2})
- 20 — No user IDs or file protection

21 Historically, filesystems for embedded realtime systems typically have had
22 a one-level name space, contiguous allocation of disk space, and relatively
23 short filenames. They have not supported an arbitrary hierarchy of named
24 directories, non-contiguous allocation of disk space, or long filenames.
25 They may have had numbered directories (e.g. **RSX-11M**^{TM3}), or only con-
26 tiguous allocation of disk space (e.g., **RT-11**TM)

27
28 However, recent commercial offerings have supported multilevel named
29 directories and both contiguous and non-contiguous disk space allocation.
30 In these implementations, the support of these features with potentially
31 non-deterministic performance does not preclude an application from
32 restricting itself to features with deterministic performance. For example,
33 it is still possible to use contiguous files exclusively. Because it is relatively
34 easy to implement both, and need not interfere with deterministic perfor-
35 mance, the working group did not make a distinction between realtime
36 and time-sharing file systems in this AEP.

37
38 Although few embedded systems had a hard drive and a file system,
39 present flash memory technology has enabled embedded systems, even
40 those with strict vibration requirements, to have a file system resident on
41 this kind of non-volatile media. This has caused the POSIX.13 profile
42 designed for large embedded systems, the Dedicated Realtime System Pro-
43 file (PSE53), to incorporate a simplified file system in this new revision of
44 the standard.

-
- 45
 - 46 1. **MS-DOS** is a registered trademark of Microsoft Corporation.
 - 47 2. **RT-11** is now a registered trademark of Compaq.
 - 48 3. **RSX-11M** is now a registered trademark of Compaq.
 - 49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 Traditional implementations of POSIX.1 filesystems employ a disk buffer
2 cache to improve average performance by reducing the number of physical
3 media accesses, and by reordering the accesses to take advantage of the
4 characteristics of rotating media. These implementations have not made a
5 distinction between the buffering of data transfers [*read()* and *write()*], and
6 directory operations [*creat()*, *link()*, *unlink()*, *mkdir()*, *rmdir()*, *rename()*]. A
7 result of this is that a system crash at an unexpected moment can leave
8 the filesystem in a corrupted state. This situation is usually corrected at
9 the next system reboot by a filesystem checker and recovery program, such
10 as *fsck*. The checking and correcting of a corrupted filesystem may take a
11 long and variable amount of time to perform, may require a human opera-
12 tor to monitor and control its progress, and may nonetheless fail to repair
13 the filesystem. Any one of these characteristics would make a filesystem
14 check unacceptable for some embedded realtime applications. It was there-
15 fore suggested that such applications limit their use of directory opera-
16 tions to *safe* times, and that implementations maintain the filesystem in
17 such a way that a filesystem check during reboot is avoided. This was con-
18 sidered, but rejected on the grounds that not all applications would require
19 the capability, and that it was neither specifiable nor testable.

21 Network Communication

22
23 Realtime systems supporting networking generally provide the following
24 features:

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

29 Distributed File System

30
31 Realtime systems supporting a distributed (non-local) file system gener-
32 ally provide the following features:

- 33 — Remote access to a filesystem
- 34 — Performance not realtime

37 Memory Protection

38
39 Realtime systems supporting memory protection (typically requiring a
40 memory management unit) generally provide the following features:

- 41 — Memory mapping and protection
- 42 — Ability to map to special areas of memory (I/O page, frame buffer)
- 43 — 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
- **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 subclause 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 subclause 1.6, “Summary of Profile Features”, for a full description of the differences between the profiles.

The “core” building block in Figure I.1 refers to the units of functionality and options required in all four profiles. See subclause 6.2, “Operating System Interface Requirements”, 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 a MMU).

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

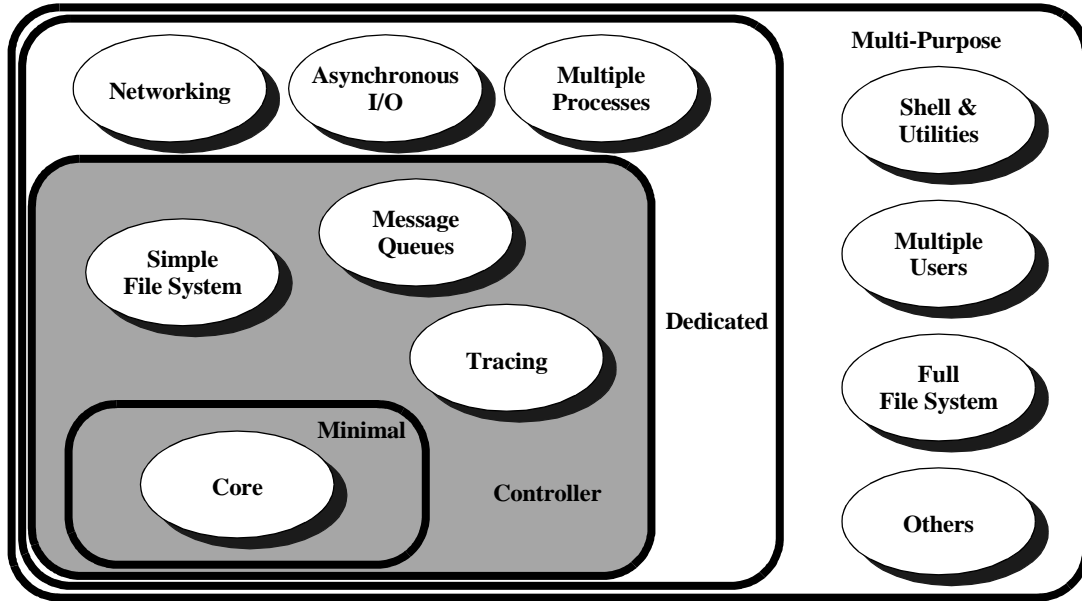


Figure I.1. Main Building Blocks of the Profiles

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 future¹:

- (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

1. A Standards Status Report that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone: +1 202 371-0101; FAX: +1 202 728-9614. Working drafts of POSIX standards under development are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331 (<http://www.standards.ieee.org/>).

1 If you have interest in participating in the Portable Application Standards Com-
2 mittee (PASC) working groups addressing these issues, please send your name, ad-
3 dress, and phone number to

4 Secretary, IEEE Standards Board
5 Institute of Electrical and Electronics Engineers, Inc.
6 P.O. Box 1331
7 445 Hoes Lane
8 Piscataway, NJ 08855-1331
9 USA
10

11 When writing, ask to have your letter forwarded to the chairperson of the appro-
12 priate PASC working group.
13

14 If you have interest in participating in this work at the international level, contact
15 your International Organization for Standardization/International Electrotechni-
16 cal Committee (ISO/IEC) national body.
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 IEEE Std 1003.13-1998 was prepared by the System Services Working Group—Re-
 2 realtime, sponsored by the Portable Application Standards Committee of the IEEE
 3 Computer Society. At the time this standard was approved, the membership of the
 4 System Services Working Group—Realtime was as follows:
 5

6
 7 **Portable Application Standards Committee**

8
 9 Chair: Lowell Johnson
 10 Vice Chair: Joseph M. Gwinn
 11
 12 Functional Vice Chairs: Jay Ashford
 13 Andrew Josey
 14 Curtis Royster, Jr.
 15
 16 Secretary: Nick Stoughton
 17

18
 19 **IEEE System Services Working Group—Realtime**

20
 21 Chair: Joseph M. Gwinn
 22 Susan Corwin (to 1995)
 23
 24 Secretary: Karen D. Gordon
 25 Frank Prindle (1996)
 26 Lee Schermerhorn (to
 27 1994)
 28
 29 Technical Editor: Bob Luken
 30
 31 Ballot Coordinators: Andrew E. Wheeler, Jr.
 32 James T. Oblinger
 33
 34 Technical Reviewers: Andrew E. Wheeler, Jr.
 35 Joseph M. Gwinn
 36 Karen D. Gordon
 37

38
 39 **Working Group**

40 Ray Alderman	Michael Feustel	Dave Lunger
41 Larry Anderson	Bill Gallmeister	Bill Maes
42 Pierre-Jean Arcos	Michael González	James T. Oblinger
43 Charles R. Arnold	Karen D. Gordon	Offer Pazy
44 V. Raj Avula	Randy Greene	Carolyn Petersen
45 Theodore P. Baker	Rick Greer	Dave Plauser
46 Todd Bargorek	Joseph M. Gwinn	Arlan Pool
47 Robert Barked	Steven A. Haaser	Franklin C. Prindle
48 Richard M. Bergman	Barbara Haleen	François Riche
49 Nawaf Bitar	Geoffrey R. Hall	Robert Rose

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1	Steve Brosky	Patrick Hebert	Gordon W. Ross
2	David Butenhof	Mary R. Hermann	Barry Ruzek
3	Hans Petter Christiansen	David Hughes	Webb Scales
4	Dave Cooper	Duane Hughes	Lee Schermerhorn
5	Susan Corwin	Michael B. Jones	Del Swanson
6	Bill Cox	Steven Kleiman	Barry Traylor
7	June R. Curtis	Robert Knighten	Stephen R. Walli
8	Peter Dibble	C. Douglass Locke	Andrew E. Wheeler, Jr.
9	Christoph Eck	Kent Long	David Wilner
10	Maryland R. Edwards	Robert D. Luken	John Zolnowsky
11			
12			
13			

Ballot Group

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

18	Norman Aaronson	John Gilbert	Robert D. Luken
19	Alejandro Alonso-Muñoz	Michael González	Dave Lungert
20	Pierre-Jean Arcos	Karen D. Gordon	Marshall McKusick
21	Charles R. Arnold	Mars J. Gralia	Craig B. Meyers
22	Theodore P. Baker	Randy Greene	Diana Norwood
23	Robert Barned	Joseph M. Gwinn	James T. Oblinger
24	Jason Behm	Steven A. Haaser	Dave Plauger
25	Richard M. Bergman	Geoffrey R. Hall	Arlan Pool
26	Andy R. Bihain	Patrick Hebert	Franklin C. Prindle
27	Shirley Bockstahler-Brandt	Hans H. Heilborn	Paul Rabin
28	Steve Case	Duane Hughes	Wendy Rauch
29	Hu Cheng	Hal Jespersen	Henry H. Robbins
30	Hans Pietter Christiansen	Michael B. Jones	Steven Schwarm
31	Susan Corwin	Joe Kelsey	Del Swanson
32	Donald Cragun	Judy Kerner	Sandra Swearingen
33	June R. Curtis	Lawrence J. Kilgallen	James G. Tanner
34	Lee Damico	Martin J. Kirk	Mark-Rene Uchida
35	Christoph Eck	Thomas M. Kurihara	Andrew E. Wheeler, Jr.
36	James A. Eiler	Kevin Lewis	David Wilner
37	Philip H. Enslow	C. Douglass Locke	Oren Yuen
38	Donna K. Fisher	Kent Long	John J. Zenor
39	Michel Gien	James P. Lonjers	John Zolnowsky
40		Lee W. Lucas	
41			
42			
43			

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

46	Richard J. Holleman,	Judith Gorman,	Donald N. Heirman,
47	<i>Chair</i>	<i>Secretary</i>	<i>Vice Chair</i>
48			
49			

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

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

James H. Gurney
 Jim D. Isaak
 Lowell G. Johnson
 Robert Kennelly
 E.G. “Al” Kiener
 Joseph L. Koepfinger^a
 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

a. Member emeritus

Noelle Humenick
IEEE Standards Project Editor

1 IEEE P1003.13 was prepared by the System Services Working Group—Realtime,
2 sponsored by the Portable Application Standards Committee of the IEEE Comput-
3 er Society. At the time this standard was approved, the membership of the System
4 Services Working Group—Realtime was as follows:
5
6

7 **Portable Application Standards Committee**

8
9 Chair: Lowell Johnson
10 Vice Chair: Joseph M. Gwinn
11
12 Functional Vice Chairs: Jay Ashford
13 Andrew Josey
14 Curtis Royster Jr.
15
16 Secretary: Nick Stoughton
17

18 **IEEE System Services Working Group—Realtime**

19
20
21 Chair: Joseph M. Gwinn
22 Secretary: Karen D. Gordon
23
24 Technical Editor: Michael González
25
26 Ballot Coordinator: Jim Oblinger
27
28 Technical Reviewers: Michael González
29

30 **Working Group**

31
32 <to be added later>
33

34 **Ballot Group**

35
36 <to be added later>
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

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

Section 1: Overview

1.1 Scope

This standard establishes a set of Realtime and Embedded Environment Profiles based on IEEE Std 1003.1-2001, IEEE Std 1003.5-1992 as amended by IEEE Std 1003.5b-1996 and IEEE Std 1003.5c-1998, 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, and ISO/IEC 9945-2:1993.

The Application Environment Profiles 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 document.

1.2 Taxonomy Position

P— OSE Profiles

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 P1003.13)

This document 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 OSE profile is complete and coherent within the context of the class of applications supported. As such, it is a System Profile class of Application Environment Profile (AEP).

1.3 Realtime System Profiles

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

1.3.1 Minimal Realtime System Profile (PSE51)

These systems are typically embedded in 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) 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

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 and controlled either by memory-mapped I/O or by the basic I/O interface, which
2 provides a standard way to access the intrinsically nonstandard I/O hardware and
3 its non-portable control code.

4
5 The hardware model for this profile assumes a single processor with its memory,
6 but no memory management unit (MMU) or common I/O devices are required. (If
7 there are in fact multiple processors, typically there are multiple instantiations of
8 the operating system, perhaps communicating via shared memory or a backplane
9 channel, perhaps isolated).

10 11 12 **1.3.2 Realtime Controller System Profile (PSE52)**

13
14
15 These systems are an extension of the Minimal Realtime System Profile. Support
16 for a file system interface and asynchronous (non-blocking) I/O interfaces has been
17 added.

18
19 The hardware model for this profile assumes a single processor and memory space
20 (a MMU is not required). Mass storage devices are not required; the file system
21 may, for instance, be implemented in memory (RAM disk or flash memory).

22 23 24 **1.3.3 Dedicated Realtime System Profile (PSE53)**

25
26
27 These systems are an extension of the Realtime Controller System Profile. Support
28 for multiple processes has been added. Although these are usually embedded sys-
29 tems, flash memory technology enables presence of a simplified file system, even
30 in those systems with mechanical or environmental requirements that preclude a
31 rotating-media hard drive. Since memory management hardware may be provid-
32 ed, the functionality of memory locking is provided.

33
34 The hardware model for this profile assumes one or more processors, each with its
35 own MMU, in the same system.

36 37 38 **1.3.4 Multi-Purpose Realtime System Profile (PSE54)**

39
40
41 These systems include all the functionality of the other three profiles. They provide
42 comprehensive functionality and run a mix of differing realtime and non-realtime
43 tasks. This functionality includes most of POSIX.1 and/or POSIX.5c. Since users
44 may conduct interactive sessions on those systems, all the mandatory elements of
45 the Shell and Utilities volume of POSIX.1 are also included. Support for multiple
46 multi-threaded processes is required so that multi-tasking may be done by threads
47 (POSIX.1 threads or Ada tasks), processes, or both.

48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 The hardware model for this profile assumes one or more processors with memory
 2 management units, high-speed storage devices, special interfaces, network sup-
 3 port, and display devices. The system supports a mix of realtime and non-realtime
 4 tasks, some being interactive user tasks.
 5
 6
 7

8 **1.4 Units of Functionality**

9
 10
 11 Some of the profiles specified in this standard do not require support for all the
 12 functionality specified in a referenced standard. In this case, if that referenced
 13 standard does not contain options for specifying just the required functionality,
 14 only those Units of Functionality referenced by the profile may be used by a strictly
 15 conforming application.
 16

17 Table 1-1 shows the Units of Functionality defined for POSIX.1; each of these units
 18 represents a Subprofiling Option Group (See the Base Definitions Volume of
 19 POSIX.1 {3}, Section 2.1.5.1, “Subprofiling Considerations”), and is a set of func-
 20 tions that represents a separately implementable element of POSIX.1. Table 1-2
 21 through Table 1-18 show the Units of Functionality defined for POSIX.5c.
 22

23 **Table 1-1: POSIX.1 Units of Functionality**

Unit of Functionality	Included Functions
POSIX_C_LANG_JUMP	<i>longjmp()</i> , <i>setjmp()</i>

24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

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

Unit of Functionality	Included Functions
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>asinhhl()</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>casinhhl()</i> , <i>casinl()</i> , <i>catan()</i> , <i>catanf()</i> , <i>catanh()</i> , <i>catanhf()</i> , <i>catanhl()</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>ceilhl()</i> , <i>cexp()</i> , <i>cexpf()</i> , <i>cexpl()</i> , <i>cimag()</i> , <i>cimagf()</i> , <i>cimagl()</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>csinhhl()</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>ctanhl()</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>fabsl()</i> , <i>fdim()</i> , <i>fdimf()</i> , <i>fdiml()</i> , <i>floor()</i> , <i>floorf()</i> , <i>floorl()</i> , <i>fma()</i> , <i>fmaf()</i> , <i>fmal()</i> , <i>fmax()</i> , <i>fmaxf()</i> , <i>fmaxl()</i> , <i>fmin()</i> , <i>fminf()</i> , <i>fminl()</i> , <i>fmod()</i> , <i>fmodf()</i> , <i>fmodl()</i> , <i>fpclassify()</i> , <i>frexp()</i> , <i>frexpf()</i> , <i>frexpl()</i> , <i>hypot()</i> , <i>hypotf()</i> , <i>hypotl()</i> , <i>ilogb()</i> , <i>ilogbf()</i> , <i>ilogbl()</i> , <i>isfinite()</i> , <i>isgreater()</i> , <i>isgreaterequal()</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>log10l()</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>rint()</i> , <i>rintf()</i> , <i>rintl()</i> , <i>lround()</i> , <i>lroundf()</i> , <i>lroundl()</i> , <i>modf()</i> , <i>modff()</i> , <i>modfl()</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>nextafterl()</i> , <i>nexttoward()</i> , <i>nexttowardf()</i> , <i>nexttowardl()</i> , <i>pow()</i> , <i>powf()</i> , <i>powl()</i> , <i>remainder()</i> , <i>remainderf()</i> , <i>remainderl()</i> , <i>remquo()</i> , <i>remquof()</i> , <i>remquol()</i> , <i>rint()</i> , <i>rintf()</i> , <i>rintl()</i> , <i>round()</i> , <i>roundf()</i> , <i>roundl()</i> , <i>scalbln()</i> , <i>scalblnf()</i> , <i>scalblnl()</i> , <i>scalbn()</i> , <i>scalbnf()</i> , <i>scalbnl()</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>tanhl()</i> , <i>tanl()</i> , <i>tgamma()</i> , <i>tgammaf()</i> , <i>tgammaf()</i> , <i>trunc()</i> , <i>truncf()</i> , <i>truncl()</i>

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

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

Unit of Functionality	Included Functions
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>feholdexcept()</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>strtolu()</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>vsprintf()</i> , <i>vswscanf()</i> , <i>wctomb()</i> , <i>wcscat()</i> , <i>wcschr()</i> , <i>wscmp()</i> , <i>wscoll()</i> , <i>wscopy()</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>ungetc()</i> , <i>vfprintf()</i> , <i>vscanf()</i> , <i>vprintf()</i> , <i>vscanf()</i> , <i>write()</i>

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

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

Unit of Functionality	Included Functions
POSIX_DEVICE_SPECIFIC	<i>cfgetispeed(), cfgetospeed(), cfsetispeed(), cfsetospeed(), ctermid(), isatty(), tcdrain(), tcflow(), tcflush(), tcgetattr(), tcsendbreak(), 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^a	<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(), 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_PRIORITY_RANGES	<i>sched_get_priority_max(), sched_get_priority_min(), sched_rr_get_interval()</i>
POSIX_REGEX^b	<i>regcomp(), regerror(), regexexec(), regfree()</i>

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

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

Unit of Functionality	Included Functions
POSIX_RW_LOCKS^c	<i>pthread_rwlock_destroy()</i> , <i>pthread_rwlock_init()</i> , <i>pthread_rwlock_rdlock()</i> , <i>pthread_rwlock_timedrdlock()</i> ^d , <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> ^e , <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>getenv()</i> , <i>setenv()</i> , <i>sysconf()</i> , <i>uname()</i> , <i>unsetenv()</i>
POSIX_STRING_MATCHING	<i>fnmatch()</i> , <i>getopt()</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^f	<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 © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

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

Unit of Functionality	Included Functions
POSIX_USER_GROUPS	<i>getegid(), geteuid(), getgid(), getgroups(), getlogin(), getlogin_r(), getuid(), setegid(), seteuid(), setgid(), setuid()</i>
POSIX_WIDE_CHAR_IO	<i>fgetwc(), fgetwts(), fputwc(), fputwts(), 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(), srand48(), srandom(), strcasecmp(), strdup(), strfmon(), strncasecmp(), strptime(), swab(), swapcontext(), tdelete(), tfind(), timezone(), 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>fntmsg(), 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(), setpgrp(), 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 © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

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

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

- a. There is a matching option in POSIX.1 called `_POSIX_JOB_CONTROL`, but that standard does not describe which functions fall under that option.
- b. There is a matching option in POSIX.1 called `_POSIX_REGEX`, but that standard does not describe which functions fall under that option.
- c. There is a matching option in POSIX.1 called `_POSIX_READER_WRITER_LOCKS`, but that standard does not describe which functions fall under that option.
- d. Dependent on the `_POSIX_TIMEOUTS` option.
- e. Dependent on the `_POSIX_THREAD_PROCESS_SHARED` option.
- f. `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, Section 2.8
System_Storage_Elements	All ^a
POSIX_Page_Alignment	All
POSIX_Supplement_To_Ada_IO	All
Ada_Task_Identification	All
Ada_Streams	All

- a. *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 © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

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

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

POSIX_EVENT_MGMT	
Package	Subprograms
POSIX_Event_Management ^a	Make_Empty Add Remove In_Set Select_File For_Every_File_In

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

- a. The subprograms listed in this table are those under the Select option in POSIX.5c. But instead of using this option, a unit of functionality has been created because there is no equivalent option in POSIX.1.

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

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

POSIX_FILE_SYSTEM	
Package	Subprograms
POSIX_Configurable_File_Limits	All
POSIX_File_Status	All

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

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

POSIX_FILE_SYSTEM	
Package	Subprograms
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 ^a	
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

a. The subprograms listed in this table are those under the Job Control option in POSIX.5c. But instead of using this option, a unit of functionality has been created because the equivalent option in POSIX.1 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

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 ^a
POSIX_Sockets_Local	All ^a
POSIX_Sockets_Internet	All ^b

- a. 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, but they are included here because there are no equivalent options in POSIX.1.
- b. 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, but it is included here because there are no equivalent options in POSIX.1.

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

POSIX_PIPES	
Package	Subprograms
POSIX_IO	Create_Pipe

Table 1-14: POSIX.5 Units of Functionality (Priority Ranges)

POSIX_PRIORITY_RANGES	
Package	Subprograms
POSIX_Process_Scheduling	Get_Maximum_Priority Get_Minimum_Priority Get_Round_Robin_Interval

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Table 1-15: 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 ^a Await_Signal_Or_Timeout ^a Interrupt_Task Get_Signal ^b Set_Signal ^b Get_Notification Set_Notification Get_Data ^b Set_Data ^b

- a. Return type Signal
- b. Operation on type Signal_Event

Table 1-16: POSIX.5 Units of Functionality (Single Process)

POSIX_SINGLE_PROCESS	
Package	Subprograms
POSIX	All
POSIX_Limits	All
POSIX_Options	All
POSIX_Profiles	All ^a
POSIX_Configurable_System_Limits	All
POSIX_Calendar	All

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Table 1-16: POSIX.5 Units of Functionality (Single Process) (Continued)

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

- a. The POSIX_Profiles package is defined in Annex B of this document

Table 1-17: POSIX.5 Units of Functionality (System Database)

POSIX_SYSTEM_DATABASE	
Package	Subprograms
POSIX_Group_Database	All
POSIX_User_Database	All

Table 1-18: 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

1.5 Development Environment

Although the Shell and Utilities part of POSIX.1 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 options POSIX2_C_BIND and POSIX2_C_DEV are required for C-Language development environments.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1.6 Summary of Profile Features

The following tables summarize the requirements of the four profiles. Since POSIX.1, and/or POSIX.5 do 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 1-18 and defined by Annex A and Annex B.

Table 1-19: Units of Functionality Requirements

Unit of Functionality	PSE51	PSE52	PSE53	PSE54
POSIX_ADA_LANG_SUPPORT ^a	X	X	X	X
POSIX_C_LANG_JUMP ^b	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_PRIORITY_RANGES	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
XSI_C_LANG_SUPPORT ^b	-	-	-	-
XSI_DBM ^b	-	-	-	-

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

Table 1-19: Units of Functionality Requirements (Continued)

Unit of Functionality	PSE51	PSE52	PSE53	PSE54
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	-	-	-	-

a. Required only for the Ada-Language option

b. Required only for the C-Language option

Table 1-20: 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_JOB_CONTROL	-	-	-	X
_POSIX_MAPPED_FILES	-	X	X	X
_POSIX_MEMLOCK	X	X	X	X
_POSIX_MEMLOCK_RANGE	X	X	X	X
_POSIX_MEMORY_PROTECTION	-	-	X	X
_POSIX_MESSAGE_PASSING	-	X	X	X

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

Table 1-20: POSIX.1 Option Requirements (Continued)

 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Option	PSE51	PSE52	PSE53	PSE54
_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_READER_WRITER_LOCKS	X	X	X	X
_POSIX_REALTIME_SIGNALS	X	X	X	X
_POSIX_REGEX	-	-	-	X
_POSIX_SAVED_IDS	-	-	-	X
_POSIX_SEMAPHORES	X	X	X	X
_POSIX_SHARED_MEMORY_OBJECTS	X	X	X	X
_POSIX_SHELL	-	-	-	X
_POSIX_SPAWN	-	-	X	X
_POSIX_SPIN_LOCKS	-	-	-	-
_POSIX_SPARADIC_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_BIND^a	X ^b	X ^b	X ^b	X
_POSIX2_C_DEV^a	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	-	-	-	-

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

Table 1-20: POSIX.1 Option Requirements (Continued)

Option	PSE51	PSE52	PSE53	PSE54
_POSIX2_PBS_CHECKPOINT	-	-	-	-
_POSIX2_PBS_LOCATE	-	-	-	-
_POSIX2_PBS_MESSAGE	-	-	-	-
_POSIX2_PBS_TRACK	-	-	-	-
_POSIX2_SW_DEV	X ^b	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			

a. Required only for the C-language option.

b. Required only for the development platform, which will often differ from the execution platform.

The correspondence between the options listed in Table 1-20 and the options described in POSIX.5c, clause 2.5, are as follows:

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
_POSIX_NO_TRUNC	Filename Truncation^a
_POSIX_PRIORITIZED_IO	Prioritized I/O
_POSIX_PRIORITY_SCHEDULING	Priority Process Scheduling
_POSIX_RAW_SOCKETS	none
_POSIX_READER_WRITER_LOCKS	none
_POSIX_REALTIME_SIGNALS	Realtime Signals

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

POSIX.1 Option	POSIX.5c Option
_POSIX_REGEX	none
_POSIX_SAVED_IDS	Saved IDs Support
_POSIX_SEMAPHORES	Semaphores
_POSIX_SHARED_MEMORY_OBJECTS	Shared Memory Objects
_POSIX_SHELL	not applicable
_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_SPORADIC_SERVER	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_BIND	not applicable
_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
_XOPEN_CRYPT	none
_XOPEN_ENH_I18N	none

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

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

POSIX.1 Option	POSIX.5c Option
<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

- a. 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 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, clause 2.4.1.5.)

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 2: Normative References

2.1 Normative References

The following standards contain provisions which, through references in this text, constitute provisions of this standard¹. 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 8652:1995², *Information Technology—Programming languages—Ada*, including Technical Corrigendum No.1.
- {2} ISO/IEC 9899:1999, *Information processing systems—Programming languages—C*, including Technical Corrigendum No. 1.
- {3} IEEE Std 1003.1:2001, *Information Technology—Portable Operating System Interface (POSIX®)* (Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992)³.
- {4} IEEE Std 1003.5-1992, *IEEE Standard for Information Technology—POSIX Ada Language Interfaces—Part 1: Binding for System Application Program Interface (API)*.
- {5} IEEE Std 1003.5b-1996, *IEEE Standard for Information Technology—POSIX Ada Language Interfaces—Part 1: Binding for System Application Program Interface (API)—Amendment 1: Realtime Extensions*.

-
- 1. Other references to related standards and other documents can be found in Annex C of this document. Common names for these standards can be found in 4.2, “Abbreviations”.
 - 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, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).
 - 3. Includes IEEE Std 1003.1d-1999, IEEE Std 1003.1j-1999, and IEEE Std 1003.1q-2000

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

- 1 {6} IEEE Std 1003.5c-1998, *IEEE Standard for Information Technology—*
2 *POSIX Ada Language Interfaces—Part 1: Binding for System Application*
3 *Program Interface (API)—Amendment 2: Protocol Independent Interfaces.*
4
5 {7} ISO/IEC TR 10000-1:1998 *Information technology -- Framework and taxon-*
6 *omy of International Standardized Profiles -- Part 1: General principles and*
7 *documentation framework.*
8
9 {8} ISO/IEC TR 10000-3:1998 *Information technology -- Framework and taxon-*
10 *omy of International Standardized Profiles -- Part 3: Principles and Taxono-*
11 *my for Open System Environment Profiles.*
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

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.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 For an application, describes a feature or behavior that is recommended program-
2 ming practice for optimum portability.
3
4

5 **3.1.5 undefined:** Describes the nature of a value or behavior not defined by
6 POSIX.13 which results from use of an invalid program construct or invalid data
7 input.
8

9 The value or behavior may vary among implementations that conform to
10 POSIX.13. An application should not rely on the existence or validity of the value
11 or behavior. An application that relies on any particular value or behavior cannot
12 be assured to be portable across conforming implementations.
13
14

15 **3.1.6 unspecified:** Describes the nature of a value or behavior not specified by
16 POSIX.13 which results from use of a valid program construct or valid data input.
17

18 The value or behavior may vary among implementations that conform to
19 POSIX.13. An application should not rely on the existence or validity of the value
20 or behavior. An application that relies on any particular value or behavior cannot
21 be assured to be portable across conforming implementations.
22
23
24

25 **3.2 Definitions**

26
27

28 For the purposes of this standard, the following definitions apply:
29
30
31

32 **3.2.1 Application Environment Profile (AEP):** An OSE profile which speci-
33 fies a complete and coherent subset of the Open System Environment. [ISO/IEC
34 TR 10000-3:1998 {8}]
35
36

37 **3.2.2 Application Platform:** A set of resources on which an application will
38 run.
39
40

41 **3.2.3 Base Standard:** An approved IEEE, National, Regional, or International
42 Standard which defines and describes basic functionality and capability. [ISO/IEC
43 TR 10000-1:1998 {7}]
44
45

46 **3.2.4 Component Profile:** An Application Environment Profile that specifies a
47 unit of functionality in terms of the interfaces that it supports and the interfaces
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 that it uses, and the relationships between these interfaces. [ISO/IEC TR 10000-
2 3:1998 {8}]
3

4
5 **3.2.5 Conformance Document:** A document provided by an implementor that
6 contains implementation details as described in 5.1.1.2.
7

8
9 **3.2.6 Development Platform:** A system used to prepare an application for ex-
10 ecution. Such a system is possibly distinct from the system on which the applica-
11 tion will execute.
12

13
14 **3.2.7 Embedded Computer System:** A computer (and its software) is consid-
15 ered *embedded* if it is an integral component of a larger system and is used to con-
16 trol and/or directly monitor that system, using special hardware devices.
17

18
19 **3.2.8 Generic Application Environment Profile:** An Application Environ-
20 ment Profile which is not specific to a particular community of use [ISO/IEC TR
21 10000-3:1998 {8}]
22

23
24 **3.2.9 Generic Interface Profile:** An Interface Profile which is not specific to a
25 particular community of use. [ISO/IEC TR 10000-3:1998 {8}]
26

27
28 **3.2.10 Industry Specific Application Environment Profile:** An Application
29 Environment Profile which deals with specific industry requirements. [ISO/IEC
30 TR 10000-3:1998 {8}]
31

32
33 **3.2.11 Industry Specific Interface Profile:** An Interface Profile which deals
34 with specific industry requirements. [ISO/IEC TR 10000-3:1998 {8}]
35

36
37 **3.2.12 Interface Profile (IP):** An OSE Profile defining one interface of the Open
38 System Environment. [ISO/IEC TR 10000-3:1998 {8}]
39

40
41 **3.2.13 International Standardized Profile (ISP):** An internationally agreed-
42 to, harmonized document which identifies a standard or group of standards, to-
43 gether with options and parameters, necessary to accomplish a function or set of
44 functions. [ISO/IEC TR 10000-1:1998 {8}]
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 **3.2.14 Open System Environment (OSE):** The comprehensive set of interfac-
2 es, services, and supporting formats for interoperability and/or for portability of
3 applications, data or people, as specified by information technology standards and
4 profiles. [ISO/IEC TR 10000-3:1998 {8}]
5

6
7 **3.2.15 Priority Inversion:** A condition in which a thread that is waiting for a
8 shared resource (including a CPU) is prevented from executing while a thread with
9 a lower application-specified priority is running. The delays caused by priority in-
10 version can be extremely large in the case of unbounded priority inversion (see def-
11 inition). But there are mechanisms to bound these delays to small predictable
12 intervals.
13

14 See also 3.2.21, “Unbounded Priority Inversion”.
15

16
17 **3.2.16 Profile (for ISO standardization):** A set of one or more base standards
18 (and where applicable) chosen classes, subsets, options, and parameters of those
19 base standards to accomplish a function. [ISO/IEC TR 10000-1:1998 {7}]
20
21

22 **3.2.17 Realtime Environment Profile:** A profile designed to support applica-
23 tions requiring bounded response.
24

25
26
27 **3.2.18 System Documentation:** All documentation provided with an imple-
28 mentation, except the conformance document.

29 Electronically distributed documents for an implementation are considered part of
30 the system documentation.
31

32
33 **3.2.19 Subprofiling Option Group:** A unit of functionality (See 3.2.22).
34
35

36
37 **3.2.20 System Profile:** An Application Environment Profile that specifies a set
38 of functions necessary to support a class of applications. It specifies the behavior
39 to be observed at the interfaces of the application platform on which the class of
40 applications can run. [ISO/IEC TR 10000-3:1998 {8}]
41

42 *NOTE: A system profile is defined in terms of component profiles that specify units of func-*
43 *tionality that can be combined to realize the application platform.*
44

45
46 **3.2.21 Unbounded Priority Inversion:** A priority inversion condition in which
47 the delay caused to the waiting thread cannot be bounded by the duration of the
48 intervals during which lower priority threads hold the shared resource. For exam-
49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 ple, this can happen when a lower priority thread is holding a lock also requested
 2 by the high priority thread, and then one or more medium priority threads request
 3 execution, thus preempting the lower priority thread

4
 5 See also 3.2.15, “Priority Inversion”.

6
 7
 8 **3.2.22 Unit of Functionality:** A separately implementable element of an OSE
 9 system. [ISO/IEC TR 10000-3:1998 {8}]

10
 11
 12
 13 **3.3 Rationale for definitions**

14
 15
 16 *(This subclause is not a normative part of IEEE Std P1003.13)*

17
 18 **Embedded Computer System.** For the definition of an embedded computer system
 19 the following canonical examples were taken into account:

- 20 — Are programs that understand physics and/or hardware embedded? For ex-
 21 ample one that uses finite-element methods to predict fluid flow over air-
 22 plane wings? No. These programs are never considered to be embedded
 23 because they are not an integral component of a larger system.
- 24 — Is the internal microprocessor controlling a disk drive an example of an em-
 25 bedded system? Yes, regardless of what the disk drive is used for. The soft-
 26 ware (firmware, actually) within the disk drive controls the HDA (head disk
 27 assembly) hardware, and is hard realtime as well.
- 28 — I/O drivers control hardware, so does presence of an I/O driver imply that
 29 the computer executing the driver is embedded? No, because that computer
 30 may be a general-purpose computer that is not part of a larger system.
- 31 — Is a PDA (Personal Digital Assistant) an embedded system? No. People of-
 32 ten say that PDAs are embedded because they are very small and con-
 33 strained, and because PDA OS and application software is kept in non-
 34 volatile memory, but PDAs parallel the desktop systems used to run office
 35 productivity applications, and no special hardware is being controlled.
- 36 — Is the microprocessor controlling a cellphone an embedded system? Yes.
 37 The firmware in the cellphone is controlling the radio hardware.
- 38 — Are the computers in a big phased-array radar considered embedded?
 39 These radars are ten-story buildings with one to three 100-foot diameter ra-
 40 diating patches on the sloped sides of the building. Yes. These computers
 41 are generally some of the most powerful computers available when the sys-
 42 tem was built, live in a large computer room occupying almost one whole
 43 floor of a building, may be hundreds of meters away from the radar hard-
 44 ware.

45
 46
 47
 48
 49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

- 1 ware, but these computers are still an integral component of a larger sys-
2 tem.
3
4 — Is a traditional Flight Management System (FMS) built into an airplane
5 cockpit considered embedded? If the FMS is not connected to the avionics,
6 and is used only for logistics computations, a function readily performed on
7 a laptop, then the FMS is clearly not embedded.
8
9 — Are the computers in a hardware-in-the-loop (HIL) simulator embedded?
10 Yes, both in the simulator, and in the thing being tested in the HIL simu-
11 lator. Hardware is being controlled on both sides.
12
13 — Is the computer controlling a pacemaker in a person’s chest an embedded
14 computer? Yes. In this case the “system” is the combination of the pacemak-
15 er and the person’s heart.
16
17 — Is the computer controlling fuel injection in an automobile engine embed-
18 ded? Yes. It is part of a larger system, the engine, and it is directly monitor-
19 ing and controlling the engine through special hardware.
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 4: Conventions and Abbreviations

4.1 Conventions

This document uses the following typographic conventions:

(1) The *italic* font is used for:

- Symbolic parameters that are generally substituted with real values by the application
- C language data types and function names
- Global external variable names
- Function families; references to groups of closely related functions

(2) The bold font is used with a word in all capital letters, such as

PATH

to represent an environment variable. It is also used for the term “NULL pointer.”

Sometimes it is used in tables to enhance visibility of option names.

(3) The constant-width (Courier) font is used:

- For references to utility names and C language headers
- For names of attributes in attributes objects
- For references to Ada identifiers.

(4) Symbolic constants returned by many functions as error numbers are represented as:

[ERRNO]

(5) Symbolic constants or limits defined in certain headers are represented as:

`_POSIX_AEP_REALTIME_`

In some cases tabular information is presented “inline”; in others it is presented in a separately labeled table. This arrangement was employed purely for ease of type-setting and there is no normative difference between these two cases.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 The conventions listed previously are for ease of reading only. Editorial inconsis-
2 tencies in the use of typography are unintentional and have no normative meaning
3 in this standard.

4 Notes provided as parts of labeled tables and figures are integral parts of this Stan-
5 dard (normative). Footnotes and notes within the body of the text are for informa-
6 tion only (informative).

7 Numerical quantities are presented in international style: comma is used as a dec-
8 imal sign and units are from the International System (SI).

11 4.2 Abbreviations

12 For the purposes of this document the following abbreviations apply:

13
14
15
16
17
18
19
20 **4.2.1 Ada95 RM:** ISO/IEC 8652:1995, *Information Technology—Programming*
21 *languages—Ada [Revision of the first edition (ISO 8652:1987)]*, including Technical
22 Corrigendum No. 1.

23
24
25 **4.2.2 C99 Standard:** ISO/IEC 9899:1999, *Information processing systems—*
26 *Programming languages—C*, including Technical Corrigendum No. 1.

27
28
29
30 **4.2.3 MMU:** Memory management unit.

31
32
33 **4.2.4 POSIX.1:** IEEE Std 1003.1:2001, *Information Technology—Portable Op-*
34 *erating System Interface (POSIX®)* (Revision of IEEE Std 1003.1-1996 and IEEE
35 Std 1003.2-1992).

36
37
38 **4.2.5 POSIX.5c:** IEEE Std 1003.5-1992, *IEEE Standard for Information Tech-*
39 *nology—POSIX Ada Language Interfaces—Part 1: Binding for System Application*
40 *Program Interface (API)* as amended by IEEE Std 1003.5b-1996, *IEEE Standard*
41 *for Information Technology—POSIX Ada Language Interfaces—Part 1: Binding*
42 *for System Application Program Interface (API)—Amendment 1: Realtime Exten-*
43 *sions* and IEEE Std 1003.5c-1998, *IEEE Standard for Information Technology—*
44 *POSIX Ada Language Interfaces—Part 1: Binding for System Application Program*
45 *Interface (API)—Amendment 2: Protocol Independent Interfaces.*

46
47
48
49
Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 1 **4.2.6 POSIX.13:** This standard. |
- 2
- 3
- 4 **4.2.7 AEP:** Application Environment Profile.
- 5
- 6
- 7
- 8 **4.2.8 ISP:** International Standardized Profile.
- 9
- 10
- 11 **4.2.9 OSE:** Open System Environment.
- 12
- 13
- 14 **4.2.10 PSE:** Generic Environment Profile.
- 15
- 16
- 17
- 18 **4.2.11 PSE51:** The Minimal Realtime System Profile defined herein.
- 19
- 20
- 21 **4.2.12 PSE52:** The Realtime Controller System Profile defined herein.
- 22
- 23
- 24 **4.2.13 PSE53:** The Dedicated Realtime System Profile defined herein.
- 25
- 26
- 27
- 28 **4.2.14 PSE54:** The Multi-Purpose Realtime System Profile defined herein.
- 29
- 30
- 31 **4.2.15 PSE5X:** Any one of the PSE51, PSE52, PSE53 or PSE54 profiles. |
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 5: Conformance

5.1 Conformance

5.1.1 Implementation Conformance

5.1.1.1 Requirements

An implementation may claim conformance to one or more of the profiles defined by this standard. For any given profile a conforming implementation shall meet all of the following criteria:

- (1) The system shall support all required interfaces referenced in the appropriate standardized profile. These interfaces shall support the functional behavior described in the appropriate base standard, and any additional constraints or options described herein.
- (2) The system may provide additional functions or facilities not required by this standard. Nonstandard extensions should be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of functions or facilities defined in the appropriate base standard. The conformance document shall define an environment in which an application can be run with predictable behavior specified by the referenced standards. In no case shall such an environment require modification of a Strictly Conforming POSIX.13 Application.

5.1.1.2 Documentation

An implementation conforming to one or more of the profiles defined by this standard shall provide a conformance document that shall document conformance in one of two specific manners:

- (1) If the implementation is fully conformant to the referenced base standard(s), then that implementation may cite the separate conformance doc-

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 uments that document the base standard conformance. This will primarily
2 apply to implementations that support the PSE53 or PSE54 Profiles.

- 3
4 (2) If the implementation does not fully conform to one or more of the refer-
5 enced base standards, or if separate base standard conformance documents
6 are not cited, the implementation shall document the specific extent of con-
7 formance to each such base standard. This specification shall include:
- 8 — A complete list of interfaces from the base standard that are present in
9 the implementation.
 - 10 — Limit values whose specification is normally required in a conformance
11 document for the base standard (e.g. the limit values found in the
12 <limits.h> and <unistd.h> headers), stating values, the conditions
13 under which those values may change, and the limits of such varia-
14 tions, if any.
 - 15 — A description of the behavior of the implementation for all implemen-
16 tation-defined features specified by those portions of the base standard
17 that the implementation provides. This requirement shall be met by
18 listing these features and providing either a specific reference to the
19 system documentation or providing full syntax and semantics of these
20 features. The conformance document may specify the behavior of the
21 implementation for those features where the referenced standards
22 state that the implementations may vary or where features are identi-
23 fied as undefined or unspecified.

24
25
26 Regardless of whether separate base standard conformance documents are cited,
27 the conformance document for these profile(s) shall contain a statement that indi-
28 cates the full name, number, and date of the standard (i.e. the profile standard)
29 that applies. The conformance document may also list international standards
30 that are available for use by a Conforming POSIX.13 Application. Applicable char-
31 acteristics where documentation is required by one of these standards or by stan-
32 dards of government bodies, may also be included.

33 34 35 36 **5.1.2 Application Conformance**

37
38
39 An application claiming conformance to one or more of these profiles shall use only
40 the facilities described in that profile and included referenced standard elements,
41 and shall fall within one of the categories in 5.1.2.1, 5.1.2.2, or 5.1.2.3.

42 Any application that conforms to one or more of these profiles under the C-Lan-
43 guage option also conforms to POSIX.1. Any application that conforms to one or
44 more of these profiles under the Ada-Language option also conforms to POSIX.5c.

45
46
47
48
49
Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.1.2.1 Strictly Conforming Application

An application is said to be strictly conforming to a given POSIX.13 profile if the application requires only the facilities required in that profile. Such an application shall accept any behavior described in the profile as *unspecified* or *implementation-defined*, and for symbolic constants, shall accept any value in the range permitted by the profile. Such applications are permitted to adapt to the availability of facilities whose availability is indicated by the constants in 6.1.3, 7.1.3, 8.1.3 and 9.1.3.

5.1.2.2 Conformant Application

5.1.2.2.1 ISO/IEC Conformant Application

An application is said to be ISO/IEC Conformant to a given POSIX.13 profile if the application requires only the facilities required in that profile and approved Conformant Language bindings for any ISO or IEC standard. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other ISO or IEC standards used.

5.1.2.2.2 <National Body> Conformant POSIX.13 Application

An application is said to be <National Body> Conformant to a given POSIX.13 profile if the application requires only the facilities required in that profile. Such an application shall include a statement of conformance to document all options and limit dependencies, and all other <National Body> standards used.

5.1.2.3 Conformant Application Using Extensions

An application is said to be conformant using extensions if it only uses nonstandard facilities consistent with this standard. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Conformant Application. A Conformant Application Using Extensions shall be either an ISO/IEC Conformant Application Using Extensions or a <National Body> Conformant Application Using Extensions. (See 5.1.2.2.1 and 5.1.2.2.2)

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 6: Minimal Realtime System Profile (PSE51)

6.1 Introduction

This section specifies those standards required for conformance to the Minimal Realtime System Profile option and, where applicable, the state of any options contained in those standards.

When a referenced standard specifies services beyond those required by the Minimal Realtime System Profile, only those services included in the specified Units of Functionality referenced by this profile shall be required (See Table 1-1 through Table 1-18). All the applicable definitions in POSIX.1 and/or POSIX.5c shall still apply.

6.1.1 Identification

For the C-Language implementation, symbolic names shall be used to specify the presence or absence of each option in this profile. Names reserved for use in this profile begin with the string `_POSIX_AEP_REALTIME_`. For the Ada Language implementation a set of Boolean subtypes contained in package `POSIX_Options` (defined in POSIX.5c, section 2.5) shall be used to specify the presence or absence of each option in this profile.

6.1.2 Conformance

Conformance to the Minimal Realtime System Profile option shall be indicated as follows:

- For the C-Language implementation the symbol `_POSIX_AEP_REALTIME_MINIMAL` being defined in the header `<unistd.h>`.
- For the Ada Language implementation the Boolean subtype `POSIX_Profiles.Realtime_Minimal` subtype having the range `True..True`.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

6.1.3 Options

The presence or absence of optional features shall be indicated as follows:

- For the C-language implementation, if any of the following symbols are defined in the header `<unistd.h>`:

`_POSIX_AEP_REALTIME_LANG_C99`

`_POSIX_AEP_REALTIME_LANG_Ada95`

- For the Ada language implementation, if any of the following `Boolean` subtypes has the range `True..True`, then the corresponding option is supported:

`POSIX_Profiles.Realtime_Lang_C99`

`POSIX_Profiles.Realtime_Lang_Ada95`

6.2 Operating System Interface Requirements

6.2.1 POSIX.1 Requirements (C Language Option)

The Minimal Realtime System Profile implementation shall include interfaces as defined in POSIX.1 for the following Units of Functionality (see Table 1-1):

Table 6-1: POSIX.1 Units of Functionality Requirements

Unit of Functionality
<code>POSIX_C_LANG_JUMP</code>
<code>POSIX_C_LANG_SUPPORT</code>
<code>POSIX_C_LIB_EXT</code>
<code>POSIX_DEVICE_IO</code>
<code>POSIX_FILE_LOCKING</code>
<code>POSIX_PRIORITY_RANGES</code>
<code>POSIX_SIGNALS</code>
<code>POSIX_SINGLE_PROCESS</code>
<code>POSIX_THREADS_BASE</code>
<code>XSI_THREAD_MUTEX_EXT</code>
<code>XSI_THREADS_EXT</code>

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 The Minimal Realtime System Profile implementation shall support the following
 2 options defined in POSIX.1, by defining the associated symbol with a value greater
 3 than zero:
 4

5 **Table 6-2: POSIX.1 Option Requirements**

Option
<code>_POSIX_CLOCK_SELECTION</code>
<code>_POSIX_FSYNC</code>
<code>_POSIX_MEMLOCK</code>
<code>_POSIX_MEMLOCK_RANGE</code>
<code>_POSIX_MONOTONIC_CLOCK</code>
<code>_POSIX_NO_TRUNC</code>
<code>_POSIX_REALTIME_SIGNALS</code>
<code>_POSIX_SEMAPHORES</code>
<code>_POSIX_SHARED_MEMORY_OBJECTS</code>
<code>_POSIX_SYNCHRONIZED_IO</code>
<code>_POSIX_THREAD_ATTR_STACKADDR</code>
<code>_POSIX_THREAD_ATTR_STACKSIZE</code>
<code>_POSIX_THREAD_CPU_TIME</code>
<code>_POSIX_THREAD_PRIO_INHERIT</code>
<code>_POSIX_THREAD_PRIO_PROTECT</code>
<code>_POSIX_THREAD_PRIORITY_SCHEDULING</code>
<code>_POSIX_THREAD_SPORADIC_SERVER</code>
<code>_POSIX_TIMEOUTS</code>
<code>_POSIX_TIMERS</code>

27 The value of `_POSIX_TIMER_MAX` shall be at least 64.

28 The value of `_POSIX_RTSIG_MAX` shall be at least 16.

29 The range of priorities associated with the `SCHED_RR` scheduling policy shall have
 30 at least 31 distinct values that are less than the maximum priority of the
 31 `SCHED_FIFO` policy.
 32

33 An implementation conforming to PSE51 shall provide a mechanism to configure
 34 the system so that the scheduling allocation domain has size one, and so that the
 35 binding of threads to scheduling allocation domains remains static. The mecha-
 36 nism by which this requirement is achieved shall be implementation defined. In
 37 addition, a PSE51 implementation may provide other configurations or facilities to
 38 change the size of the allocation domain and the bindings of threads to allocation
 39 domains. For a description of the scheduling allocation domain see the System In-
 40 terfaces volume of POSIX.1, Section 2.9.2, “Thread Scheduling”.
 41
 42
 43
 44
 45
 46
 47
 48
 49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

6.2.2 POSIX.5c Requirements (Ada Language Option)

The Minimal Realtime System Profile implementation shall include interfaces as defined in POSIX.5c for the following units of functionality (see Table 1-2 through Table 1-18):

Table 6-3: POSIX.5c Units of Functionality Requirements

Unit of Functionality
POSIX_ADA_LANG_SUPPORT
POSIX_DEVICE_IO
POSIX_FILE_LOCKING
POSIX_SIGNALS
POSIX_SINGLE_PROCESS

The Minimal Realtime System Profile implementation shall support the following options defined in POSIX.5c, by defining the associated option subtypes to have the range `True..True`, with the exception of the Filename Truncation option for which the associated subtype shall have the range `False..False`:

Table 6-4: POSIX.5c Option Requirements

Option
File Synchronization
Memory Locking
Memory Range Locking
Filename Truncation
Realtime Signals
Semaphores
Shared Memory Objects
Synchronized I/O
Mutexes Support
Mutex Priority Inheritance
Mutex Priority Ceiling
Timers

`POSIX_Limits.Timers_Maxima'First` shall be at least 64.

`POSIX_Limits.Realtime_Signals_Maxima'First` shall be at least 16.

Regarding task priority scheduling, the implementation shall support the following requirements from POSIX.5c and the Ada95 RM:

- The implementation shall support the priority model defined in the Ada95 RM, clause D.1, and the pragmas and package interfaces defined in the Ada95 RM, clauses D.2-D.5.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

- 1 — The implementation shall meet the requirements of POSIX.5c, section
2 13.3.1.

3
4 The blocking behavior of all reentrant operations defined by POSIX.5c shall be per
5 task, i.e., a blocked task cannot prevent any other task from executing. Therefore,
6 the corresponding `Blocking_Behavior` constants shall have the value `Tasks`.
7 (See POSIX.5c, clause 2.4.1.5.)

8 Implementations of the PSE51 profile shall support the `POSIX_Profiles` pack-
9 age defined in Annex A of this standard.

10
11 The subprogram `POSIX_Signals.Set_Stopped_Child_Signal` shall fail si-
12 lently.

13 The subprogram `POSIX_Signals.Stopped_Child_Signal_Enabled` shall re-
14 turn `False`.

15
16 `POSIX_Limits.Groups_Maxima'First` shall be zero.

17
18 Subprograms not supported by a given profile shall raise `POSIX_Error`, returning
19 an error code of `Operation_Not_Supported`, except as noted otherwise.

20 All `Image` and `Value` functions that appear in the packages supported by a profile
21 must be implemented.

22
23 Where an overloaded subprogram is required by a unit of functionality, all forms
24 of the subprogram appearing in the referenced clause must be supported, except
25 as otherwise noted.

26 27 28 29 30 **6.3 Application Constraints**

31
32 The Minimal Realtime System profile defined in this standard requires only spe-
33 cific `Units of Functionality` of the required standards. The absence of particular el-
34 ements of these standards introduces constraints on the use of some of the features
35 of particular operations. This clause defines the constraints that an application
36 strictly conforming to one of the profiles shall observe when using each of the op-
37 erations required by that profile.

38 39 40 41 **6.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)**

42
43
44 The following table defines a set of functions that shall be either reentrant or non-
45 interruptible by signals and shall be `async-signal-safe`. Therefore applications may
46 invoke them, without restriction, from signal-catching functions. No other func-
47 tion, including those defined in the System Interfaces Volume of POSIX.1, Section
48 2.4.3, “Signal Actions”, is required to be `async-safe` in an implementation of the
49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

PSE51 profile, and thus PSE51 Strictly Conforming Applications shall not use them from inside signal handlers.

Table 6-5: Functions required to be async-signal-safe

<i>alarm()</i>	<i>sigaddset()</i>	<i>sigpending()</i>	<i>timer_getoverrun()</i>
<i>clock_gettime()</i>	<i>sigdelset()</i>	<i>sigprocmask()</i>	<i>timer_gettime()</i>
<i>kill()</i>	<i>sigemptyset()</i>	<i>sigqueue()</i>	<i>timer_settime()</i>
<i>raise()</i>	<i>sigfillset()</i>	<i>sigset()</i>	<i>times()</i>
<i>sem_post()</i>	<i>sigismember()</i>	<i>sysconf()</i>	<i>uname()</i>
<i>sigaction()</i>	<i>signal()</i>	<i>time()</i>	

The *sysconf()* function has the following constraints:

- (1) An application strictly conforming to the PSE51 profile shall not call the *sysconf()* function with the parameter `_POSIX_VERSION` since a meaningful value cannot be returned.¹
- (2) A conforming application must act as if `CHILD_MAX=0`.

An application strictly conforming to PSE51 shall be considered erroneous if any signal results in abnormal termination of the process because this profiles does not support multiple processes.

An application strictly conforming to PSE51 shall not call the *kill()* function with a negative but not -1 argument because this profile does not require process group functionality.

An application strictly conforming to PSE51 shall be guaranteed that the file mode creation mask for any object created by any process is `S-IRWXU`; that is, the object shall be fully accessible to the creator.

An application strictly conforming to PSE51 shall not use the *open()*, *fopen()*, or *freopen()* functions to create new files, since this profile does not require general file system capabilities.

An application strictly conforming to PSE51 shall use the path or file argument for any function using a file pathname (e.g., *open()*) only to specify the name of the object without any file system semantics implied, since this profile does not require general file system semantics.

An application strictly conforming to PSE51 shall not require that any input/output function (e.g., *fclose()*, *fflush()*, *fgetc()*, *fgets()*, *fopen()*, *fprintf()*, *fputc()*, *fputs()*, *fread()*, *fscanf()*, *fwrite()*, *getc()*, *getchar()*, *gets()*, *open()*, *perror()*, *printf()*, *putc()*, *putchar()*, *puts()*, *read()*, *scanf()*, *vfprintf()*, *vfscanf()*, *vprintf()*, *vscanf()*, *write()*) update an access, creation, or modification time for the device read or written, because this profile requires no interfaces that could query such an access time.

1. Conformance to this profile can be checked with the symbols defined in 6.1.3.

6.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)

An application strictly conforming to PSE51 shall not call the functions `POSIX_Configurable_System_Limits.System_POSIX_Version` or `POSIX_Configurable_System_Limits.System_POSIX_Ada_Version`, since a meaningful value cannot be returned.¹

A conforming application must act as if `POSIX_Limits.Child_Processes_Maxima'Last=0`.

An application strictly conforming to PSE51 shall be considered erroneous if any signal results in abnormal termination of the process because this profile does not support multiple processes.

An application strictly conforming to PSE51 shall not call the form of `POSIX_Signals.Send_Signal` that takes a process group ID as an argument because this profile does not require process group functionality.

An application strictly conforming to PSE51 shall not attempt to bind a signal to a task entry.

An application strictly conforming to PSE51 shall not use the `POSIX_IO.Open_Or_Create` function to create new files, since this profile does not require general file system capabilities.

An application strictly conforming to PSE51 shall use a parameter representing a pathname (such as the `Name` parameter of `POSIX_IO.Open` or `POSIX_IO.Open_Or_Create`) only to specify the name of the object without any file system semantics implied, since this profile does not require general file system semantics.

An application strictly conforming to PSE51 shall not require that any input/output function such as `POSIX_IO.Read`, `POSIX_IO.Generic_Read`, `POSIX_IO.Write`, or `POSIX_IO.Generic_Write`, update an access, creation, or modification time for the device read or written, because this profile requires no interfaces that could query such an access time.

Implementations of PSE51 need not support the `Owner`, `Group`, and `Other` fields of the `form` parameter (See POSIX.5c, clause 8.1.1.2), but may instead raise `Use_Error`. The default value used shall be `Read_Write_Execute`.

Implementations of PSE51 need not support the `File_Structure` field of the `form` parameter (See POSIX.5c, clause 8.1.1.2), but may instead raise `Use_Error`. All files shall default to regular files.

In addition, the following constraints apply to the usage of the predefined Ada I/O packages:

1. Conformance to this profile can be checked with the subtypes defined in 6.1.3.

- 1 (1) An application strictly conforming to PSE51 shall not require any of the In-
2 put/Output operations (Read, Write, Get, Put, etc.) contained in the pre-
3 defined Ada I/O packages or their instantiations to update an access,
4 creation, or modification time for the device read or written, because this
5 profile requires no interfaces that could query such an access time.
- 6
7 (2) An application strictly conforming to PSE51 shall use the Name of the Open
8 operations contained in the predefined Ada I/O packages or their instanti-
9 ations only to specify the name of the object without any file system seman-
10 tics implied, since this profile does not require general file system
11 capabilities.
- 12 (3) An application strictly conforming to PSE51 shall not call any of the
13 Create or Delete operations contained in the predefined Ada I/O packag-
14 es or their instantiations, since this profile does not require general file sys-
15 tem capabilities.
16
17
18
19

20 **6.4 Shell and Utility Requirements**

21
22
23 An implementation of the Minimal Realtime System Profile is not required to sup-
24 port any of the services described in the Shell and Utilities Volume of POSIX.1.
25
26
27

28 **6.5 Development Platform Requirements**

29
30
31 One or more of the development options in 6.5.1 and 6.5.2 shall be implemented.
32
33
34

35 **6.5.1 C Language Development Option**

36
37
38 If this option is provided, the implementor shall define a Development Platform
39 and an environment capable of preparing for execution an application conformant
40 with this standard profile. This platform shall include the POSIX2_C_BIND,
41 POSIX2_C_DEV, and POSIX2_SW_DEV options from the Shell and Utilities Volume
42 of POSIX.1.
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

6.5.1.1 Option Indicator

The presence of the C Language Development Option shall be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_C99` being defined in the required header `<unistd.h>`. In addition, the presence of the C Language Development Option may be indicated by the subtype `POSIX_Profiles.Realtime_Lang_C99` having the range `True..True`.

6.5.2 Ada Language Development Option

If this option is provided, the implementor shall define a Development Platform and an environment capable of preparing for execution an application conformant with this profile including applicable portions of the following:

- The Ada95 RM {1}
- POSIX.5c {6}
- The `POSIX2_SW_DEV` option from the Shell and Utilities Volume of POSIX.1.

6.5.2.1 Option Indicator

The presence of the Ada Language Development Option shall be indicated by the subtype `POSIX_Profiles.Realtime_Lang_Ada95` having the range `True..True`. In addition, the presence of the Ada Language Development Option may be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_Ada95` being defined in the header `<unistd.h>`.

6.6 Rationale for Operating System Requirements (informative)

(This subclause is not a normative part of IEEE Std P1003.13)

6.6.1 Operating System Interface Requirements

After reviewing several commercially available small realtime kernels, it was concluded that the POSIX.1 threads model (with all options enabled, but without a file system) best reflected current industry practice in certain embedded realtime ar-

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 eas. Instead of full file system support, basic device I/O (read, write, open, close,
2 control) is considered sufficient for kernels of this size. Systems of this size fre-
3 quently do not include process isolation hardware or software; therefore, multiple
4 processes (as opposed to threads) may not be supportable.

5
6 System options that allow an application to be upwards compatible without modi-
7 fying application source code have been chosen. For example, although the as-
8 sumed hardware model implies fixed address space without a Memory
9 Management Unit (MMU), the symbol `_POSIX_MEMLOCK` is still defined. This in-
10 creases portability of the application code to higher level systems that do not nec-
11 essarily have the same restrictions.

12 13 14 **6.6.1.1 Process Primitives**

15
16
17 Because this profile uses the POSIX.1 Threads model only as the mechanism to
18 achieve concurrency, most POSIX.1 process primitives do not apply. This includes
19 the multi-process, pipes, and signal jump units of functionality, as well as the pro-
20 cess spawn option.

21 The *main()* function is needed to allow application-specific information to be
22 passed from boot code to the single (implicit) process (and its threads).

23 24 25 26 **6.6.1.2 Signals**

27
28
29 Signal services are a basic mechanism within POSIX-based systems and are re-
30 quired for error and event handling. Realtime systems typically have several logi-
31 cally concurrent software elements executing. Each such entity must respond to
32 several cyclic and/or acyclic stimuli, often in a time-critical manner. Although
33 purely synchronous models can supply such functionality via the use of additional
34 processes or threads, the current realtime practice for asynchronous notification
35 for events such as timeout, message arrival, and hardware interrupt can generally
36 be expected to offer higher performance and lower latency. Realtime Signals pro-
37 vide the reliable high-performance mechanism to support such notification.

38
39 The minimum number of realtime signals that the implementation is required to
40 support has been increased from the number specified in the POSIX.1 standard, 8,
41 up to 16. The rationale for this increase is that there are many applications that
42 have more than 8 different kinds of events. Doubling the number of required real-
43 time signals should have a minimum impact on the signal management overhead,
44 while significantly increases the number of event kinds that can be used by a
45 strictly conforming application.

46
47
48
49
Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

6.6.1.3 Process Environment

The functions from the POSIX.1 Process Environment group are deemed necessary to allow an application to determine and configure its system environment. This allows a single version of an application to be run on similar but differing platforms; however, conforming applications must act as if `CHILD_MAX=0`.

6.6.1.4 Files and Directories

The `open()` function is needed to do basic device I/O and also to provide device initialization. Although this requires some form of name resolution, a full pathname space is specifically not required. Directories also are not required. Units of functionality or options associated with files, such as `POSIX_FD_MGMT`, `POSIX_FIFO`, `POSIX_FILE_ATTRIBUTES`, `POSIX_FILE_SYSTEM`, `POSIX_FILE_SYSTEM_EXT`, `_POSIX_ADVISORY_INFO`, and `_POSIX_MAPPED_FILES`, are not required.

Since a file system is not a part of this realtime profile, the `_POSIX_NO_TRUNC` option is applied to the names of devices and shared memory objects.

The File Locking option is required in the C-language option to maintain a consistent and safe way of accessing stdio (*FILE* *) objects from threads, across the four realtime profiles.

6.6.1.5 Input and Output Primitives

The functions contained in the Device I/O unit of functionality are required to do basic I/O and device cleanup.

Asynchronous I/O is not required because it can be easily implemented using threads dedicated to I/O.

6.6.1.6 Synchronized Input and Output

The Synchronized (unbuffered) I/O interface (including the File Synchronization option) is typical for basic device I/O and is required for upward portability.

6.6.1.7 Device- and Class-Specific Functions

POSIX.1 Device- or Class-Specific functions are not required, because small embedded systems usually don't require general-purpose terminal interfaces.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

6.6.1.8 System Databases, Users and Groups

Implementations are not required to support more than one user and group id since there are not multiple users and groups. No POSIX.1 System Database functions are required.

6.6.1.9 Synchronization

Mutexes and Condition Variables are required as part of the threads model of concurrency.

The Process Shared option is not required because there is only a single process.

Semaphores are required in the PSE51 profile for synchronization between threads to maintain compatibility with past industry practice. However, mutexes and condition variables are preferred in most current applications. It must be noted that POSIX semaphores do not have the mechanisms built in to avoid unbounded priority inversion when using them for mutually exclusive access to shared resources. Mutexes with the appropriate priority inheritance or priority ceiling (also called priority protection) protocols can be used to avoid this unbounded priority inversion.

Barriers are not required because they can easily be implemented using mutexes and condition variables. Although a direct implementation of barriers can have a significant efficiency benefit in some multiprocessor architectures, a mutex-and-condition-variable implementation will not be significantly slower in most architectures, and thus requiring barriers for all implementations is not justified.

Spin locks are not required because, although they are an efficient synchronization mechanism, they cannot be portably used with the current POSIX.1 interfaces in realtime applications. If a realtime scheduling policy such as SCHED_FIFO or SCHED_RR is used, spin locks may cause deadlock on a single processor. On multiprocessors, to avoid deadlock, it would be necessary for threads using a given lock to be allocated to different processors. There are no standard APIs in the current POSIX.1 to allocate threads to specific processors.

Reader/Writer Locks are not required because they are not designed to avoid unbounded priority inversion, and thus very long delays could occur in realtime applications, with a low but nevertheless non-zero probability. It is expected that a future revision of the POSIX.1 standard will add the priority inheritance and/or priority ceiling options to reader/writer locks, which would eliminate the unbounded priority inversion.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

6.6.1.10 Priority Scheduling

Thread priority scheduling is required for realtime applications. The Sporadic Server Scheduling option is also required to enhance support of applications with aperiodic timing requirements. The POSIX_PRIORITY_RANGES unit of functionality is required because threads need to obtain the values of the priority ranges associated with realtime scheduling policies to use those policies.

A common requirement of realtime systems is that they be able to run threads with real-time requirements together with threads with no real-time requirements. One common way of doing this is by having the real-time threads run under the SCHED_FIFO scheduling policy, while the non real-time threads run at a lower priority under the round-robin policy (SCHED_RR) to fairly share the available portion of the processor among them. POSIX requires each policy to have a range of priorities of at least 32 distinct values, but does not impose any requirements on how these priority ranges relate to each other. It could happen that most or all of the SCHED_RR priorities were larger than the SCHED_FIFO priorities, thus making it impossible to mix realtime and non-realtime threads as required above. To solve this problem in a portable way, this profile requires that there are at least 31 SCHED_RR priority levels below the maximum priority of SCHED_FIFO. In this way, a strictly conforming application can use the inclusive priority range [*max_FIFO_prio*, *max_FIFO_prio*-30] with SCHED_FIFO for real-time threads (with a total of 31 priority levels), and then use the priority value $\min(\text{max_FIFO_prio}-31, \text{max_RR_prio})$ with the SCHED_RR policy, for the non real-time threads, with guarantee that the latter priority value is valid for the round-robin policy.

Support for a scheduling allocation domain of size one and static binding of threads to allocation domains is required in all the realtime profiles to achieve predictable scheduling behavior. The allocation domain of a thread is the set of processors on which that thread can be scheduled at any given time. The POSIX.1 standard specifies that the scheduling rules have predictable effects only if the allocation domain is of size one; hence the need for this requirement. For single-processor systems the allocation domain is generally of size one and thus the application can meet the requirement just by specifying in the conformance document that the scheduling allocation domain is of size one and that static binding of threads to allocation domains is the default behavior.

6.6.1.11 Process Memory Locking

Process memory locking is inherent in systems following this profile because most PSE51 targets have no MMU and thus swapping is not supported; code and data stay in physical memory until explicitly removed. Nevertheless, memory locking APIs are required for upward portability to allow an application developer to take code intended for a bare PSE51 target and unit test that code on a much larger and more capable platform, perhaps a PSE54, with minimal modification. In those tar-

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 gets not using a MMU for virtual memory, the locking functions do nothing and al-
2 ways report success, while in the larger profiles there really is memory to be
3 locked. In summary, by requiring this service in the PSE51 profile, it is possible to
4 write portable application code that runs correctly in all the profiles.
5
6

7 **6.6.1.12 Shared Memory**

8
9

10 Memory Mapped I/O may be implemented using the Shared Memory facility. An
11 implementation is required to provide facilities for creating (shared) memory ob-
12 jects that represent ranges of physical memory that contain device control and sta-
13 tus registers or buffers. These facilities encourage the development of portable
14 applications.
15

16 Typed Memory objects are not required because they are useful only to systems
17 with special hardware architectures that have various often specialized kinds of
18 memory. Implementors providing support for such special architectures always
19 have the option to provide typed memory objects as an extension.
20
21

22 **6.6.1.13 Clocks and Timers**

23
24

25 High-resolution timer functions are required in most realtime systems for imple-
26 menting time management operations such as periodic activations, short duration
27 time-outs, etc. The normal POSIX.1 time management functions *sleep()* and
28 *alarm()* only provide a time resolution of one second, but many realtime systems
29 require finer resolution for specifying time.
30

31 The Monotonic Clock is required for realtime applications to ensure that deadlines
32 and timing requirements are not affected by clock jumps.

33 The Clock Selection option is required to enable choosing the clock on which sleep
34 operations are performed, and to have access to an absolute sleep operation, which
35 is a common requirement in realtime applications with periodic timing require-
36 ments.
37

38 CPU-Time clocks and timers are required as a means to detect and handle situa-
39 tions in which a thread overruns its assigned maximum execution time. Bounding
40 the execution times of the different threads in the application increases predict-
41 ability and reliability.

42 The Timeouts option is a general requirement for realtime applications and thus
43 is required in this profile.
44

45 The minimum number of timers that the implementation is required to support
46 has been increased from the number specified in the POSIX.1 standard, 32, up to
47 64, which is the required minimum number of threads. The reason for this increase
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 is that there are many applications that require one timer per thread (either real-
2 time or CPU-time based).
3
4

5 **6.6.1.14 Message Passing**

6
7

8 In the PSE51 profile of IEEE Std 1003.13-1998, message queues were required be-
9 cause commercial realtime kernels available at that time with similar function-
10 ality to the Minimal Realtime System Profile typically included some form of
11 message queueing mechanism for communication between threads.
12

13 However, many embedded realtime applications for small systems do not require
14 message queues and this feature makes the implementation larger. Because mes-
15 sage queues can be easily implemented by the application using mutexes and con-
16 dition variables, this version of the standard has dropped the requirement to
17 support message queues.
18

19 **6.6.1.15 Threads**

20
21
22

23 The basic assumption in this profile is that the system will consist of a single (im-
24 plicit) process, with multiple threads. Therefore, all basic thread services are re-
25 quired, except for those related to multiple processes. The
26 POSIX_THREADS_BASE unit of functionality was specified in this document in-
27 stead of the POSIX_THREADS option, because this option requires reader/writer
28 locks, but this profile does not.
29
30

31 **6.6.1.16 Tracing**

32
33

34 Tracing is not required for the PSE51 environment to keep the implementation of
35 this profile small.
36
37

38 **6.6.1.17 Networking**

39
40

41 Although some small embedded systems require networking services, most don't,
42 so to keep the implementation small, this unit of functionality is not required.
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

6.6.1.18 Event Management

The *select()* function is usually associated with networking facilities, which are not required for PSE51. Although the function could be used for regular device I/O operations, most kernels that do not have networking services do not support *select()*. Therefore, to keep the implementation small, the event management unit of functionality is not required.

6.6.1.19 Interfaces Related to the Shell and Utilities

Interfaces defined in the `POSIX_REGEX` and `POSIX_SHELL_FUNC` units of functionality are related to shells and utilities, which are not required in this profile; therefore, these units of functionality are not required either.

6.6.1.20 X/Open Units of Functionality and Options

Some XSI Units of Functionality (`XSI_C_LANG_SUPPORT`, `XSI_DEVICE_IO`, `XSI_DEVICE_SPECIFIC`, `XSI_FD_MGMT`, `XSI_FILE_SYSTEM`, `XSI_IPC`, `XSI_JOB_CONTROL`, `XSI_JUMP`, `XSI_MATH`, `XSI_MULTI_PROCESS`, `XSI_SIGNALS`, `XSI_SINGLE_PROCESS`, `XSI_SYSTEM_DATABASE`, `XSI_TIMERS`, `XSI_USER_GROUPS`, `XSI_WIDE_CHAR`) have interfaces that represent extensions or alternatives to interfaces in other Units of Functionality or POSIX.1 options, and therefore are not necessary for PSE51 environments.

The `XSI_DBM` unit of functionality includes interfaces for database management that are not required in the PSE51 application environment.

The `XSI_DYNAMIC_LINKING` unit of functionality is not required for small embedded systems, which usually operate in a static context.

The `XSI_I18N` unit of functionality provides facilities for natural language messages to the user, which are not required in small embedded systems, which typically do not have general-purpose human interfaces.

The `XSI_SYSTEM_LOGGING` unit of functionality provides facilities for logging system activities, which are not required in PSE51 environments.

The `XSI_THREAD_MUTEX_EXT` unit of functionality is required because it has options for controlling the behavior of mutexes under erroneous application use. This capability is interesting for any realtime application, including those targeted at small embedded systems.

The `XSI_THREADS_EXT` unit of functionality is required because it provides functions to better control a thread's stack. This is considered useful for any realtime application.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 The `_XOPEN_CRYPT` option provides cryptography facilities that are not required
2 in PSE51 environments.

3
4 The `_XOPEN_LEGACY` option provides facilities for backwards compatibility that
5 are not required in PSE51 environments.

6 The `_XOPEN_STREAMS` option provides facilities that are mainly related to net-
7 working, and thus are not required for PSE51 environments, as discussed above.
8
9

10 **6.6.1.21 Language-Specific Services for the C Programming Language**

11
12
13 Support for the C Language is required in the C Language option, with the excep-
14 tions of the `POSIX_C_LANG_MATH` and `POSIX_C_LANG_WIDE_CHAR` units of func-
15 tionality. The reasons for these exceptions are that these are very large libraries
16 that are not necessary for many of the PSE51 applications.
17
18

19 **6.6.1.22 Language-Specific Services for the Ada Programming Language**

20
21
22 Support for the Ada language-specific services defined in POSIX.5c is required in
23 the Ada Language option.
24
25

26 **6.6.2 Shell and Utility Requirements**

27
28
29
30 Because the Minimal Realtime System Profile is intended for small embedded sys-
31 tems which usually have no terminal or graphical user interface, such a platform
32 would be incapable of executing a shell. In such an environment the utilities de-
33 scribed in the Shell and Utilities Volume of POSIX.1 are not usually required.
34
35

36 **6.6.3 Development Platform Requirements**

37
38
39
40 The embedded nature of the PSE51 execution platform makes it difficult to use as
41 a development platform. Therefore, the implementation is required to define a de-
42 velopment environment in which a PSE51 application can be prepared for execu-
43 tion on the target platform. The development platform depends on the language
44 option chosen by the implementation.
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 7: Realtime Controller System Profile (PSE52)

7.1 Introduction

This section specifies those standards required for conformance to the Realtime Controller System Profile option and, where applicable, the state of any options contained in those standards.

When a referenced standard specifies services beyond those required by the Realtime Controller System Profile, only those services included in the specified Units of Functionality referenced by this profile shall be required (See Table 1-1 through Table 1-18). All the applicable definitions in POSIX.1 and/or POSIX.5c still apply.

7.1.1 Identification

For the C Language implementation, symbolic names shall be used to specify the presence or absence of each option in this profile. Names reserved for use in this profile begin with the string `_POSIX_AEP_REALTIME_`. For the Ada language implementation a set of Boolean subtypes contained in package `POSIX_Options` (defined in POSIX.5c, clause 2.5) shall be used to specify the presence or absence of each option in this profile.

7.1.2 Conformance

Conformance to the Realtime Controller System Profile option shall be indicated as follows:

- For the C language implementation the symbol `_POSIX_AEP_REALTIME_CONTROLLER` being defined in the header `<unistd.h>`.
- For the Ada language implementation the Boolean subtype `POSIX_Profiles.Realtime_Controller` having the range `True..True`.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7.1.3 Options

The presence or absence of optional features shall be indicated as follows:

- For the C language implementation, if any of the following symbols are defined in the header `<unistd.h>`, then the corresponding option is supported:

`_POSIX_AEP_REALTIME_LANG_C99`

`_POSIX_AEP_REALTIME_LANG_Ada95`

- For the Ada language implementation, if any of the following Boolean subtypes has the range `True..True`, then the corresponding option is supported:

`POSIX_Profiles.Realtime_Lang_C99`

`POSIX_Profiles.Realtime_Lang_Ada95`

7.2 Operating System Interface Requirements

7.2.1 POSIX.1 Requirements (C language Option)

The Realtime Controller System Profile implementation shall include interfaces as defined in POSIX.1 for the following Units of Functionality (see Table 1-1):

Table 7-1: POSIX.1 Units of Functionality Requirements

Unit of Functionality
<code>POSIX_C_LANG_JUMP</code>
<code>POSIX_C_LANG_MATH</code>
<code>POSIX_C_LANG_SUPPORT</code>
<code>POSIX_DEVICE_IO</code>
<code>POSIX_FD_MGMT</code>
<code>POSIX_FILE_LOCKING</code>
<code>POSIX_FILE_SYSTEM</code>
<code>POSIX_PRIORITY_RANGES</code>
<code>POSIX_SIGNALS</code>
<code>POSIX_SINGLE_PROCESS</code>
<code>POSIX_THREADS_BASE</code>
<code>XSI_THREAD_MUTEX_EXT</code>
<code>XSI_THREADS_EXT</code>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 The Realtime Controller System Profile implementation shall support the follow-
 2 ing options defined in POSIX.1, by defining the associated symbol with a value
 3 greater than zero:
 4

5 **Table 7-2: POSIX.1 Option Requirements**

Option
<code>_POSIX_CLOCK_SELECTION</code>
<code>_POSIX_FSYNC</code>
<code>_POSIX_MAPPED_FILES</code>
<code>_POSIX_MEMLOCK</code>
<code>_POSIX_MEMLOCK_RANGE</code>
<code>_POSIX_MESSAGE_PASSING</code>
<code>_POSIX_MONOTONIC_CLOCK</code>
<code>_POSIX_NO_TRUNC</code>
<code>_POSIX_REALTIME_SIGNALS</code>
<code>_POSIX_SEMAPHORES</code>
<code>_POSIX_SHARED_MEMORY_OBJECTS</code>
<code>_POSIX_SYNCHRONIZED_IO</code>
<code>_POSIX_THREAD_ATTR_STACKADDR</code>
<code>_POSIX_THREAD_ATTR_STACKSIZE</code>
<code>_POSIX_THREAD_CPU_TIME</code>
<code>_POSIX_THREAD_PRIO_INHERIT</code>
<code>_POSIX_THREAD_PRIO_PROTECT</code>
<code>_POSIX_THREAD_PRIORITY_SCHEDULING</code>
<code>_POSIX_THREAD_SPORADIC_SERVER</code>
<code>_POSIX_TIMEOUTS</code>
<code>_POSIX_TIMERS</code>
<code>_POSIX_TRACE</code>
<code>_POSIX_TRACE_EVENT_FILTER</code>
<code>_POSIX_TRACE_LOG</code>

32 The value of `_POSIX_TIMER_MAX` shall be at least 64.

34 The value of `_POSIX_RTSIG_MAX` shall be at least 16.

36 The range of priorities associated with the `SCHED_RR` scheduling policy shall have
 37 at least 31 distinct values that are less than the maximum priority of the
 38 `SCHED_FIFO` policy.

39 An implementation conforming to PSE52 shall provide a mechanism to configure
 40 the system so that the scheduling allocation domain has size one, and so that the
 41 binding of threads to scheduling allocation domains remains static. The mecha-
 42 nism by which this requirement is achieved shall be implementation defined. In
 43 addition, a PSE52 implementation may provide other configurations or facilities to
 44 change the size of the allocation domain and the bindings of threads to allocation
 45 domains. For a description of the scheduling allocation domain see the System In-
 46 terfaces volume of POSIX.1, Section 2.9.2, “Thread Scheduling”.
 47
 48
 49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

7.2.2 POSIX.5c Requirements (Ada Language Option)

The Realtime Controller System Profile implementation shall include interfaces as defined in POSIX.5c for the following Units of Functionality (see Table 1-1):

Table 7-3: POSIX.5c Units of Functionality Requirements

Unit of Functionality
POSIX_ADA_LANG_SUPPORT
POSIX_DEVICE_IO
POSIX_FD_MGMT
POSIX_FILE_SYSTEM
POSIX_SIGNALS
POSIX_SINGLE_PROCESS

The Realtime Controller System Profile implementation shall support the following options defined in POSIX.5c, by defining the associated option subtypes to have the range `True..True`, with the exception of the Filename Truncation option for which the associated subtype shall have the range `False..False`:

Table 7-4: POSIX.5c Option Requirements

Option
File Synchronization
Memory Mapped Files
Memory Locking
Memory Range Locking
Message Queues
Filename Truncation
Realtime Signals
Semaphores
Shared Memory Objects
Synchronized I/O
Mutexes Support
Mutex Priority Inheritance
Mutex Priority Ceiling
Timers

`POSIX_Limits.Timers_Maxima'First` shall be at least 64.

`POSIX_Limits.Realtime_Signals_Maxima'First` shall be at least 16.

Regarding task priority scheduling, the implementation shall support the following requirements from POSIX.5c and the Ada95 RM:

- The implementation shall support the priority model defined in the Ada95 RM, clause D.1, and the pragmas and package interfaces defined in the Ada95 RM, clauses D.2-D.5.

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 1 — The implementation shall meet the requirements of POSIX.5c, section
2 13.3.1.

3
4 The blocking behavior of all reentrant operations defined by POSIX.5c shall be per
5 task, i.e., a blocked task cannot prevent any other task from executing. Therefore,
6 the corresponding `Blocking_Behavior` constants shall have the value `Tasks`.
7 (See POSIX.5c, clause 2.4.1.5.)

8 Implementations of the PSE52 profile shall support the `POSIX_Profiles` pack-
9 age defined in Annex A of this standard.

10
11 The subprogram `POSIX_Signals.Set_Stopped_Child_Signal` shall fail si-
12 lently.

13 The subprogram `POSIX_Signals.Stopped_Child_Signal_Enabled` shall re-
14 turn `False`.

15
16 `POSIX_Limits.Groups_Maxima'First` shall be zero.

17
18 Subprograms not supported by a given profile shall raise `POSIX_Error`, returning
19 an error code of `Operation_Not_Supported`, except as noted otherwise.

20 All `Image` and `Value` functions that appear in the packages supported by a profile
21 must be implemented.

22
23 Where an overloaded subprogram is required by a unit of functionality, all forms
24 of the subprogram appearing in the referenced clause must be supported, except
25 as otherwise noted.

26 27 28 29 30 **7.3 Application Constraints**

31
32 The Realtime Controller System profile defined in this standard requires only spe-
33 cific `Units of Functionality` of the required standards. The absence of particular el-
34 ements of these standards introduces constraints on the use of some of the features
35 of particular operations. This clause defines the constraints that an application
36 strictly conforming to one of the profiles shall observe when using each of the op-
37 erations required by that profile.

38 39 40 41 **7.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)**

42
43
44 The following table defines a set of functions that shall be either reentrant or non-
45 interruptible by signals and shall be `async-signal-safe`. Therefore applications may
46 invoke them, without restriction, from signal-catching functions. No other func-
47 tion, including those defined in the System Interfaces Volume of POSIX.1, Section
48 2.4.3, “Signal Actions”, is required to be `async-safe` in an implementation of the
49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

PSE52 profile, and thus PSE52 Strictly Conforming Applications shall not use them from inside signal handlers.

Table 7-5: Functions required to be async-signal-safe

<i>alarm()</i>	<i>sigaddset()</i>	<i>sigpending()</i>	<i>timer_getoverrun()</i>
<i>clock_gettime()</i>	<i>sigdelset()</i>	<i>sigprocmask()</i>	<i>timer_gettime()</i>
<i>kill()</i>	<i>sigemptyset()</i>	<i>sigqueue()</i>	<i>timer_settime()</i>
<i>raise()</i>	<i>sigfillset()</i>	<i>sigset()</i>	<i>times()</i>
<i>sem_post()</i>	<i>sigismember()</i>	<i>sysconf()</i>	<i>uname()</i>
<i>sigaction()</i>	<i>signal()</i>	<i>time()</i>	

The *sysconf()* function has the following constraints:

- (1) An application strictly conforming to the PSE52 profile shall not call the *sysconf()* function with the parameter `_POSIX_VERSION` since a meaningful value cannot be returned.¹
- (2) A conforming application must act as if `CHILD_MAX=0`.

An application strictly conforming to PSE52 shall be considered erroneous if any signal results in abnormal termination of the process because this profiles does not support multiple processes.

An application strictly conforming to PSE52 shall not call the *kill()* function with a negative but not -1 argument because this profile does not require process group functionality.

An application strictly conforming to PSE52 shall be guaranteed that the file mode creation mask for any object created by any process is `S-IRWXU`; that is, the object shall be fully accessible to the creator.

7.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)

An application strictly conforming to PSE52 shall not call the functions `POSIX_Configurable_System_Limits.System_POSIX_Version` or `POSIX_Configurable_System_Limits.System_POSIX_Ada_Version`, since a meaningful value cannot be returned.²

A conforming application must act as if `POSIX_Limits.Child_Processes_Maxima'Last=0`.

An application strictly conforming to PSE52 shall be considered erroneous if any signal results in abnormal termination of the process because this profile does not support multiple processes.

-
1. Conformance to this profile can be checked with the symbols defined in 7.1.3.
 2. Conformance to this profile can be checked with the subtypes defined in 7.1.3.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 An application strictly conforming to PSE52 shall not call the form of
2 `POSIX_Signals.Send_Signal` that takes a process group ID as an argument be-
3 cause this profile does not require process group functionality.

4
5 An application strictly conforming to PSE52 shall not attempt to bind a signal to a
6 task entry.

7 Implementations of PSE52 need not support the `File_Structure` field of the
8 form parameter (See POSIX.5c, clause 8.1.1.2), but may instead raise `Use_Error`.
9 All files shall default to regular files.

14 7.4 Shell and Utility Requirements

17 An implementation of the Realtime Controller System Profile is not required to
18 support any of the services described in the Shell and Utilities Volume of POSIX.1.

22 7.5 Development Platform Requirements

25 One or more of the development options in 7.5.1 and 7.5.2 shall be implemented.

29 7.5.1 C Language Development Option

32 If this option is provided, the implementor shall define a Development Platform
33 and an environment capable of preparing for execution an application conformant
34 with this standard profile. This platform shall include the `POSIX2_C_BIND`,
35 `POSIX2_C_DEV`, and `POSIX2_SW_DEV` options from the Shell and Utilities Volume
36 of POSIX.1.

39 7.5.1.1 Option Indicator

42 The presence of the C Language Development Option shall be indicated by the
43 symbol `_POSIX_AEP_REALTIME_LANG_C99` being defined in the required header
44 `<unistd.h>`. In addition, the presence of the C Language Development Option
45 may be indicated by the subtype `POSIX_Profiles.Realtime_Lang_C99` having
46 the range `True..True`.

47
48
49
Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7.5.2 Ada Language Development Option

If this option is provided, the implementor shall define a Development Platform and an environment capable of preparing for execution an application conformant with this profile including applicable portions of the following:

- The Ada95 RM {1}
- POSIX.5c {6}
- The POSIX2_SW_DEV option from the Shell and Utilities Volume of POSIX.1.

7.5.2.1 Option Indicator

The presence of the Ada Language Development Option shall be indicated by the subtype `POSIX_Profiles.Realtime_Lang_Ada95` having the range `True..True`. In addition, the presence of the Ada Language Development Option may be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_Ada95` being defined in the header `<unistd.h>`.

7.6 Rationale for Operating System Requirements (informative)

(This subclause is not a normative part of IEEE Std P1003.13)

7.6.1 Operating System Interface Requirements

This model introduces system functionality that is more sophisticated than in the Minimal Realtime System Profile, specifically in the area of I/O. Two general categories of services are added.

The first extension is support for a simplified file and directory system. These features are used in applications that require an alterable file name space, typically in systems that support secondary storage and require the ability to create, change, and delete named regular files located on a storage device. The included functions allow the creation, deletion, and changing of file attributes of regular files.

This profile assumes the following hardware model: one or more processors with local memory and one or more serial interfaces. (It is anticipated that the serial interface(s) may be removed in final production systems.) Driver-level I/O to stan-

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 dard and non-standard devices are supported. In addition, a file system device is
2 supported. The hardware is not required to provide memory management.
3
4

5 **7.6.1.1 Process Primitives**

6
7

8 Because this profile uses the POSIX.1 Threads model only as the mechanism to
9 achieve concurrency, most POSIX.1 process primitives do not apply. This includes
10 the multi-process, pipes, and signal jump units of functionality, as well as the pro-
11 cess spawn option. Although PSE52 has only a single (implicit) process, some in-
12 terprocess APIs are required to support communication between applications.
13

14 The *main()* function is needed to allow application-specific information to be
15 passed from boot code to the single process (and its threads).
16
17

18 **7.6.1.2 Signals**

19
20

21 Signal services are a basic mechanism within POSIX-based systems and are re-
22 quired for error and event handling. Realtime systems typically have several logi-
23 cally concurrent software elements executing. Each such entity must respond to
24 several cyclic and/or acyclic stimuli, often in a time-critical manner. Although
25 purely synchronous models can supply such functionality via the use of additional
26 processes or threads, the current realtime practice for asynchronous notification
27 for events such as timeout, message arrival, and hardware interrupt can generally
28 be expected to offer higher performance and lower latency. Realtime Signals pro-
29 vide the reliable high-performance mechanism to support such notification.
30

31 The minimum number of realtime signals that the implementation is required to
32 support has been increased from the number specified in the POSIX.1 standard, 8,
33 up to 16. The rationale for this increase is that there are many applications that
34 have more than 8 different kinds of events. Doubling the number of required real-
35 time signals should have a minimum impact on the signal management overhead,
36 while significantly increases the number of event kinds that can be used by a
37 strictly conforming application.
38
39

40 **7.6.1.3 Process Environment**

41
42

43 The functions from the POSIX.1 Process Environment group are deemed necessary
44 to allow an application to determine and configure its system environment. This
45 allows a single version of an application to be run on similar but differing plat-
46 forms; however, conforming applications must act as if `CHILD_MAX=0`.
47
48
49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

7.6.1.4 Files and Directories

Since this profile has a file system, all POSIX.1 functions that manage a basic file systems are required. However, the file system in a PSE52 platform is a simplified version of a full POSIX.1 file system, and for this reason the POSIX_FIFO, POSIX_FILE_ATTRIBUTES, and POSIX_FILE_SYSTEM_EXT, units of functionality and the _POSIX_ADVISORY_INFO option are not required.

The File Locking option is required in the C-language option to maintain a consistent and safe way of accessing stdio (*FILE* *) objects from threads, across the four realtime profiles.

7.6.1.5 Input and Output Primitives

The functions contained in the Device I/O and File Descriptor Management units of functionality are required to do basic I/O and device cleanup.

Asynchronous I/O is not required because it can be easily implemented using threads dedicated to I/O.

7.6.1.6 Synchronized Input and Output

The Synchronized (unbuffered) I/O interface (including the File Synchronization option) is typical for basic device I/O and is required for upward portability.

Those realtime systems that use file management systems will frequently require synchronized I/O to provide data integrity and/or relinquish resources to other users. Synchronized I/O as defined in POSIX.1 provides these mechanisms.

7.6.1.7 Device- and Class-Specific Functions

POSIX.1 Device- or Class-Specific functions are not required, because PSE52 systems usually don't require general-purpose terminal interfaces.

7.6.1.8 System Databases, Users and Groups

Implementations are not required to support more than one user and group id since there are not multiple users and groups. No POSIX.1 System Database functions are required.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7.6.1.9 Synchronization

Mutexes and Condition Variables are required as part of threads model of concurrency.

The Process Shared option is not required because there is only a single process.

Semaphores are required in the PSE52 profile for synchronization between threads to maintain compatibility with past industry practice. However, mutexes and conditional variables are preferred in most current applications. It must be noted that POSIX semaphores do not have the mechanisms built in to avoid unbounded priority inversion when using them for mutually exclusive access to shared resources. Mutexes with the appropriate priority inheritance or priority ceiling (also called priority protection) protocols can be used to avoid this unbounded priority inversion.

Barriers are not required because they can easily be implemented using mutexes and condition variables. Although a direct implementation of barriers can have a significant efficiency benefit in some multiprocessor architectures, a mutex-and-condition-variable implementation will not be significantly slower in most architectures, and thus requiring barriers for all implementations is not justified.

Spin locks are not required because, although they are an efficient synchronization mechanism, they cannot be portably used with the current POSIX.1 interfaces in realtime applications. If a realtime scheduling policy such as SCHED_FIFO or SCHED_RR is used, spin locks may cause deadlock on a single processor. On multiprocessors, to avoid deadlock, it would be necessary for threads using a given lock to be allocated to different processors. There are no standard APIs in the current POSIX.1 to allocate threads to specific processors.

Reader/Writer Locks are not required because they are not designed to avoid unbounded priority inversion, and thus very long delays could occur in realtime applications, with a low but nevertheless non-zero probability. It is expected that a future revision of the POSIX.1 standard will add the priority inheritance and/or priority ceiling options to reader/writer locks, which would eliminate the unbounded priority inversion.

7.6.1.10 Priority Scheduling

Thread priority scheduling is required for realtime applications. The Sporadic Server Scheduling option is also required to enhance support of applications with aperiodic timing requirements. The POSIX_PRIORITY_RANGES unit of functionality is required because threads need to obtain the values of the priority ranges associated with realtime scheduling policies to use those policies.

A common requirement of realtime systems is that they be able to run threads with real-time requirements together with threads with no real-time requirements. One

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 common way of doing this is by having the real-time threads run under the
2 SCHED_FIFO scheduling policy, while the non real-time threads run at a lower pri-
3 ority under the round-robin policy (SCHED_RR) to fairly share the available por-
4 tion of the processor among them. POSIX requires each policy to have a range of
5 priorities of at least 32 distinct values, but does not impose any requirements on
6 how these priority ranges relate to each other. It could happen that most or all of
7 the SCHED_RR priorities were larger than the SCHED_FIFO priorities, thus mak-
8 ing it impossible to mix realtime and non-realtime threads as required above. To
9 solve this problem in a portable way, this profile requires that there are at least 31
10 SCHED_RR priority levels below the maximum priority of SCHED_FIFO. In this
11 way, a strictly conforming application can use the inclusive priority range
12 [*max_FIFO_prio*, *max_FIFO_prio*-30] with SCHED_FIFO for real-time threads
13 (with a total of 31 priority levels), and then use the priority value
14 $\min(\text{max_FIFO_prio}-31, \text{max_RR_prio})$ with the SCHED_RR policy, for the non
15 real-time threads, with guarantee that the latter priority value is valid for the
16 round-robin policy.

17
18 Support for a scheduling allocation domain of size one and static binding of threads
19 to allocation domains is required in all the realtime profiles to achieve predictable
20 scheduling behavior. The allocation domain of a thread is the set of processors on
21 which that thread can be scheduled at any given time. The POSIX.1 standard spec-
22 ifies that the scheduling rules have predictable effects only if the allocation domain
23 is of size one; hence the need for this requirement. For single-processor systems the
24 allocation domain is generally of size one and thus the application can meet the re-
25 quirement just by specifying in the conformance document that the scheduling al-
26 location domain is of size one and that static binding of threads to allocation
27 domains is the default behavior.

30 **7.6.1.11 Process Memory Locking**

31
32
33 Process memory locking is inherent in systems following this profile because most
34 PSE52 targets have no MMU and thus swapping is not supported; code and data
35 stays in physical memory until explicitly removed. Nevertheless, memory locking
36 APIs are required for upward portability to allow an application developer to take
37 code intended for a bare PSE52 target and unit test that code on a much larger and
38 more capable platform, perhaps a PSE54, with minimal modification. In those tar-
39 gets not using an MMU for virtual memory, the locking functions do nothing and
40 always report success, while in the larger profiles there really is memory to be
41 locked. In summary, by requiring this service in the PSE52 profile, it is possible to
42 write portable application code that runs correctly in all the profiles.

43
44
45
46
47
48
49
Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

7.6.1.12 Shared Memory

Memory Mapped I/O may be implemented using the Shared Memory facility. An implementation is required to provide facilities for creating (shared) memory objects that represent ranges of physical memory that contain device control and status registers or buffers. These facilities encourage the development of portable applications.

The Memory-Mapped Files option is included because the implementation has file-system capabilities, and memory-mapped files are a convenient paradigm for reading and writing information in applications following this profile. In memory-mapped files, I/O is not managed by the programmer because data can be manipulated as memory. The implementation of memory-mapped files does not require a significant amount of additional memory or execution overhead to achieve the additional capability.

System vendors are expected to implement the chosen interface in a manner that meets the needs of the applications. In particular, a rotating media-based implementation is allowed but not required by the interface definition.

Typed Memory objects are not required because they are useful only to systems with special hardware architectures that have various often specialized kinds of memory. Implementors providing support for such special architectures always have the option to provide typed memory objects as an extension.

7.6.1.13 Clocks and Timers

High-resolution timer functions are required in most realtime systems for implementing time management operations such as periodic activations, short duration time-outs, etc. The normal POSIX.1 time management functions *sleep()* and *alarm()* only provide a time resolution of one second, but many realtime systems require finer resolution for specifying time.

The Monotonic Clock is required for realtime applications to ensure that deadlines and timing requirements are not affected by clock jumps.

The Clock Selection option is required to enable choosing the clock on which sleep operations are performed, and to have access to an absolute sleep operation, which is a common requirement in realtime applications with periodic timing requirements.

CPU-Time clocks and timers are required as a means to detect and handle situations in which a thread overruns its assigned maximum execution time. Delimiting the execution times of the different threads in the application provides temporal partitioning in realtime applications, and thus increases predictability and reliability.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 The Timeouts option is a general requirement for realtime applications and thus
2 is required in this profile.

3
4 The minimum number of timers that the implementation is required to support
5 has been increased from the number specified in the POSIX.1 standard, 32, up to
6 64, which is the required minimum number of threads. The reason for this increase
7 is that there are many applications that require one timer per thread (either real-
8 time or CPU-time based).

11 **7.6.1.14 Message Passing**

12
13
14 Currently available commercial realtime kernels with similar functionality to the
15 Realtime Controller System Profile typically include some form of message queue-
16 ing mechanism for communication between threads. The POSIX.1 Message Pass-
17 ing offers an appropriate level of performance to provide this functionality.

20 **7.6.1.15 Threads**

21
22
23 The basic assumption in this profile is that the system will consist of a single (im-
24 plicit) process, with multiple threads. Therefore, all basic thread services are re-
25 quired, except for those related to multiple processes. The
26 POSIX_THREADS_BASE unit of functionality was specified in this document in-
27 stead of the POSIX_THREADS option, because this option requires reader/writer
28 locks, but this profile does not.

31 **7.6.1.16 Tracing**

32
33
34 Tracing is required for the PSE52 environment because most of these systems
35 work in an unattended mode for long periods of time, and tracing provides an ex-
36 cellent mechanism to support post-failure analysis, particularly for failures having
37 a low probability of occurrence.

38
39 The Trace Event Filtering option is required for the system to be able to filter out
40 those trace events that are not meaningful for the application, thus making better
41 use of system resources by capturing only the interesting events.

42
43 The presence of a file system in the PSE52 profile facilitates the recording of the
44 trace events, through the Trace Log option, which is required for this profile.

45
46
47
48
49
Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7.6.1.17 Networking

Although some small controller systems require networking services, most don't, so to keep the implementation small, this unit of functionality is not required.

7.6.1.18 Event Management

The *select()* function is usually associated with networking facilities, which are not required for PSE52. Although the function could be used for regular device I/O operations, most kernels that do not have networking services do not support *select()*. Therefore, to keep the implementation small, the event management unit of functionality is not required.

7.6.1.19 Interfaces Related to the Shell and Utilities

Interfaces defined in the POSIX_REGEX and POSIX_SHELL_FUNC units of functionality are related to shells and utilities, which are not required in this profile; therefore, these units of functionality are not required either.

7.6.1.20 X/Open Units of Functionality and Options

Some XSI Units of Functionality (XSI_C_LANG_SUPPORT, XSI_DEVICE_IO, XSI_DEVICE_SPECIFIC, XSI_FD_MGMT, XSI_FILE_SYSTEM, XSI_IPC, XSI_JOB_CONTROL, XSI_JUMP, XSI_MATH, XSI_MULTI_PROCESS, XSI_SIGNALS, XSI_SINGLE_PROCESS, XSI_SYSTEM_DATABASE, XSI_TIMERS, XSI_USER_GROUPS, XSI_WIDE_CHAR) have interfaces that represent extensions or alternatives to interfaces in other Units of Functionality or POSIX.1 options, and therefore are not necessary for PSE52 environments.

The XSI_DBM unit of functionality includes interfaces for database management that are not required in the PSE52 application environment.

The XSI_DYNAMIC_LINKING unit of functionality is not required for small embedded systems, which usually operate in a static context.

The XSI_I18N unit of functionality provides facilities for natural language messages to the user, which are not required in realtime controller systems, which typically do not have general-purpose human interfaces.

The XSI_SYSTEM_LOGGING unit of functionality provides facilities for logging system activities, which are not required in PSE52 environments.

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 The `XSI_THREAD_MUTEX_EXT` unit of functionality is required because it has op-
2 tions for controlling the behavior of mutexes under erroneous application use. This
3 capability is interesting for any realtime application, including those targeted at
4 control systems.

5 The `XSI_THREADS_EXT` unit of functionality is required because it provides func-
6 tions to better control a thread's stack. This is considered useful for any realtime
7 application.
8

9 The `_XOPEN_CRYPT` option provides cryptography facilities that are not required
10 in PSE52 environments.
11

12 The `_XOPEN_LEGACY` option provides facilities for backwards compatibility that
13 are not required in PSE52 environments.
14

15 The `_XOPEN_STREAMS` option provides facilities that are mainly related to net-
16 working, and thus are not required for PSE52 environments, as discussed above.
17

18 **7.6.1.21 Language-Specific Services for the C Programming Language**

19
20
21 Support for the C Language is required in the C Language option, with the excep-
22 tion of the `POSIX_C_LANG_WIDE_CHAR` unit of functionality. The reason for this
23 exception is that this is a very large library that is not necessary for many of the
24 PSE52 applications.
25
26
27

28 **7.6.1.22 Language-Specific Services for the Ada Programming Language**

29
30
31 Support for the Ada language-specific services defined in POSIX.5c is required in
32 the Ada Language option.
33
34
35

36 **7.6.2 Shell and Utility Requirements**

37
38 Because the Realtime Controller System Profile is intended for control systems
39 which usually have no terminal or graphical user interface, such a platform would
40 be incapable of executing a shell. In such an environment the utilities described in
41 the Shell and Utilities Volume of POSIX.1 are not usually required.
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

7.6.3 Development Platform Requirements

The special-purpose nature of the PSE52 execution platform makes it difficult to use as a development platform. Therefore, the implementation is required to define a development environment in which a PSE52 application can be prepared for execution on the target platform. The development platform depends on the language option chosen by the implementation.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 8: Dedicated Realtime System Profile (PSE53)

8.1 Introduction

This section specifies those standards required for conformance to the Dedicated Realtime System Profile option and, where applicable, the state of any options contained in those standards.

When a referenced standard specifies services beyond those required by the Dedicated Realtime System Profile, only those services included in the specified Units of Functionality referenced by this profile shall be required (See Table 1-1 through Table 1-18). All the applicable definitions in POSIX.1 and/or POSIX.5c still apply.

8.1.1 Identification

For the C-Language implementation, symbolic names shall be used to specify the presence or absence of each option in this profile. Names reserved for use in this profile begin with the string `_POSIX_AEP_REALTIME_`. For the Ada Language implementation a set of Boolean subtypes contained in package `POSIX_Options` (defined in POSIX.5c, section 2.5) shall be used to specify the presence or absence of each option in this profile.

8.1.2 Conformance

Conformance to the Dedicated Realtime System Profile option shall be indicated as follows:

- For the C-Language implementation the symbol `_POSIX_AEP_REALTIME_DEDICATED` being defined in the header `<unistd.h>`.
- For the Ada Language implementation the Boolean subtype `POSIX_Profiles.Realtime_Dedicated` subtype having the range `True..True`.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

8.1.3 Options

The presence or absence of optional features shall be indicated as follows:

- For the C-language implementation, if any of the following symbols are defined in the header `<unistd.h>`:

`_POSIX_AEP_REALTIME_LANG_C99`

`_POSIX_AEP_REALTIME_LANG_Ada95`

- For the Ada language implementation, if any of the following `Boolean` subtypes has the range `True..True`, then the corresponding option is supported:

`POSIX_Profiles.Realtime_Lang_C99`

`POSIX_Profiles.Realtime_Lang_Ada95`

8.2 Operating System Interface Requirements

8.2.1 POSIX.1 Requirements (C Language Option)

The Dedicated Realtime System Profile implementation shall include interfaces as defined in POSIX.1 for the following Units of Functionality (see Table 1-1)

Table 8-1: POSIX.1 Units of Functionality Requirements

Unit of Functionality
<code>POSIX_C_LANG_JUMP</code>
<code>POSIX_C_LANG_MATH</code>
<code>POSIX_C_LANG_SUPPORT</code>
<code>POSIX_DEVICE_IO</code>
<code>POSIX_EVENT_MGMT</code>
<code>POSIX_FD_MGMT</code>
<code>POSIX_FILE_LOCKING</code>
<code>POSIX_FILE_SYSTEM</code>
<code>POSIX_MULTI_PROCESS</code>
<code>POSIX_NETWORKING</code>
<code>POSIX_PIPE</code>
<code>POSIX_SIGNALS</code>
<code>POSIX_SIGNAL_JUMP</code>
<code>POSIX_SINGLE_PROCESS</code>
<code>POSIX_THREADS_BASE</code>
<code>XSI_THREAD_MUTEX_EXT</code>

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

Table 8-1: POSIX.1 Units of Functionality Requirements (Continued)

Unit of Functionality
XSI_THREADS_EXT

The Dedicated Realtime System Profile implementation shall support the following options defined in POSIX.1, by defining the associated symbol with a value greater than zero:

Table 8-2: POSIX.1 Option Requirements

Option
_POSIX_ASYNCHRONOUS_IO
_POSIX_CLOCK_SELECTION
_POSIX_CPUTIME
_POSIX_FSYNC
_POSIX_MAPPED_FILES
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MEMORY_PROTECTION
_POSIX_MESSAGE_PASSING
_POSIX_MONOTONIC_CLOCK
_POSIX_NO_TRUNC
_POSIX_PRIORITIZED_IO
_POSIX_PRIORITY_SCHEDULING
_POSIX_RAW_SOCKETS
_POSIX_REALTIME_SIGNALS
_POSIX_SEMAPHORES
_POSIX_SHARED_MEMORY_OBJECTS
_POSIX_SPAWN
_POSIX_SPORADIC_SERVER
_POSIX_SYNCHRONIZED_IO
_POSIX_THREAD_ATTR_STACKADDR
_POSIX_THREAD_ATTR_STACKSIZE
_POSIX_THREAD_CPUTIME
_POSIX_THREAD_PRIO_INHERIT
_POSIX_THREAD_PRIO_PROTECT
_POSIX_THREAD_PRIORITY_SCHEDULING
_POSIX_THREAD_PROCESS_SHARED
_POSIX_THREAD_SPARADIC_SERVER
_POSIX_TIMEOUTS
_POSIX_TIMERS
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_LOG

The value of _POSIX_TIMER_MAX shall be at least 64.

The value of _POSIX_RTSIG_MAX shall be at least 16.

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1 The range of priorities associated with the SCHED_RR scheduling policy shall have
 2 at least 31 distinct values that are less than the maximum priority of the
 3 SCHED_FIFO policy.

4 An implementation conforming to PSE53 shall support the
 5 PTHREAD_SCOPE_SYSTEM scheduling contention scope. In addition, it may sup-
 6 port PTHREAD_SCOPE_PROCESS. For a description of the scheduling contention
 7 scope see the System Interfaces volume of POSIX.1, Section 2.9.2, “Thread Sched-
 8 uling”.

9
 10 An implementation conforming to PSE53 shall provide a mechanism to configure
 11 the system so that the scheduling allocation domain has size one, and so that the
 12 binding of threads to scheduling allocation domains remains static. The mecha-
 13 nism by which this requirement is achieved shall be implementation defined. In
 14 addition, a PSE53 implementation may provide other configurations or facilities to
 15 change the size of the allocation domain and the bindings of threads to allocation
 16 domains. For a description of the scheduling allocation domain see the System In-
 17 terfaces volume of POSIX.1, Section 2.9.2, “Thread Scheduling”.

20 8.2.2 POSIX.5c Requirements (Ada Language Option)

21
 22 The Dedicated Realtime System Profile implementation shall include interfaces as
 23 defined in POSIX.5c for the following units of functionality (see Table 1-2 through
 24 Table 1-18):

25
 26
 27 **Table 8-3: POSIX.5c Units of Functionality Requirements**

Unit of Functionality
POSIX_ADA_LANG_SUPPORT
POSIX_DEVICE_IO
POSIX_EVENT_MGMT
POSIX_FD_MGMT
POSIX_FILE_SYSTEM
POSIX_MULTI_PROCESS ^a
POSIX_NETWORKING
POSIX_PIPE
POSIX_SIGNALS
POSIX_SINGLE_PROCESS

- 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41 a. The POSIX_MULTI_PROCESS unit of functionality shall be supported, with
 42 the provision that the package POSIX_Unsafe_Process_Primitives is
 43 not required

44
 45 The Dedicated Realtime System Profile implementation shall support the follow-
 46 ing options defined in POSIX.5c, by defining the associated option subtypes to have
 47
 48
 49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

the range True..True, with the exception of the Filename Truncation option for which the associated subtype shall have the range False..False:

Table 8-4: POSIX.5c Option Requirements

Option
Asynchronous I/O
File Synchronization
Memory Mapped Files
Memory Locking
Memory Range Locking
Memory Protection
Message Queues
Filename Truncation
Prioritized I/O
Priority Process Scheduling
Realtime Signals
Semaphores
Shared Memory Objects
Synchronized I/O
Mutexes Support
Mutex Priority Inheritance
Mutex Priority Ceiling
Process Shared
Timers

POSIX_Limits.Timers_Maxima'First shall be at least 64.

POSIX_Limits.Realtime_Signals_Maxima'First shall be at least 16.

Regarding task priority scheduling, the implementation shall support the following requirements from POSIX.5c and the Ada95 RM:

- The implementation shall support the priority model defined in the Ada95 RM, clause D.1, and the pragmas and package interfaces defined in the Ada95 RM, clauses D.2-D.5.
- The implementation shall meet the requirements of POSIX.5c, section 13.3.1.

Implementations of the PSE53 profile shall support the POSIX_Profiles package defined in Annex A of this standard.

The subprogram POSIX_Signals.Set_Stopped_Child_Signal shall fail silently.

The subprogram POSIX_Signals.Stopped_Child_Signal_Enabled shall return False.

POSIX_Limits.Groups_Maxima'First shall be zero.

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1 Subprograms not supported by a given profile shall raise `POSIX_Error`, returning
 2 an error code of `Operation_Not_Supported`, except as noted otherwise.

3
 4 All `Image` and `Value` functions that appear in the packages supported by a profile
 5 must be implemented.

6 Where an overloaded subprogram is required by a unit of functionality, all forms
 7 of the subprogram appearing in the referenced clause must be supported, except
 8 as otherwise noted.
 9

10 11 12 13 **8.3 Application Constraints**

14
15
16 The Dedicated Realtime System profile defined in this standard requires only spe-
 17 cific units of functionality of the required standards. The absence of particular el-
 18 ements of these standards introduces constraints on the use of some of the features
 19 of particular operations. This clause defines the constraints that an application
 20 strictly conforming to one of the profiles shall observe when using each of the op-
 21 erations required by that profile.
 22
 23

24 25 **8.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)**

26
27 The `sysconf()` function has the following constraints:

- 28
29 (1) An application strictly conforming to the PSE53 profile shall not call the
 30 `sysconf()` function with the parameter `_POSIX_VERSION` since a meaningful
 31 value cannot be returned.¹
 32

33 An application strictly conforming to PSE53 shall not call the `kill()` function with
 34 a negative argument because this profile does not require process group function-
 35 ality.

36 An application strictly conforming to PSE53, shall be guaranteed that the file
 37 mode creation mask for any object created by any process is `S-IRWXU`; that is, the
 38 object shall be fully accessible to the creator.
 39
 40
 41

42 43 **8.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)**

44
45 An application strictly conforming to PSE53 shall not call the functions
 46 `POSIX_Configurable_System_Limits.System_POSIX_Version` or
 47 _____

- 48 1. Conformance to this profile can be checked with the symbols defined in 8.1.3.
 49

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1 POSIX_Configurable_System_Limits.System_POSIX_Ada_Version, since
 2 a meaningful value cannot be returned.¹

3
 4 An application strictly conforming to PSE53 shall not call the subprograms con-
 5 tained in the package Posix_Unsafe_Process_Primitives, but shall instead
 6 rely upon either Posix_Process_Primitives.Start_Process or
 7 Posix_Process_Primitives.Start_Process_Search to create new process-
 8 es.

9 An application strictly conforming to PSE53 shall not call the form of
 10 POSIX_Signals.Send_Signal that takes a process group ID as an argument be-
 11 cause this profile does not require process group functionality.

12
 13 An application strictly conforming to PSE53 shall not attempt to bind a signal to a
 14 task entry.

15 Implementations of PSE53 need not support the File_Structure field of the
 16 form parameter (See POSIX.5c, clause 8.1.1.2), but may instead raise Use_Error.
 17 All files shall default to regular files.
 18

19
 20
 21

22 **8.4 Shell and Utility Requirements**

23
 24
 25
 26

27 An implementation of the Dedicated Realtime System Profile is not required to
 28 support any of the services described in the Shell and Utilities Volume of POSIX.1.

29
 30

31 **8.5 Development Platform Requirements**

32
 33
 34

35 One or more of the development options in 8.5.1 and 8.5.2 shall be implemented.

36
 37

38 **8.5.1 C Language Development Option**

39
 40
 41
 42
 43
 44
 45

46 If this option is provided, the implementor shall define a Development Platform
 47 and an environment capable of preparing for execution an application conformant
 48 with this standard profile. This platform shall include the POSIX2_C_BIND,
 49 POSIX2_C_DEV, and POSIX2_SW_DEV options from the Shell and Utilities Volume
 of POSIX.1.

46
 47
 48
 49

1. Conformance to this profile can be checked with the subtypes defined in 8.1.3.

8.5.1.1 Option Indicator

The presence of the C Language Development Option shall be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_C99` being defined in the required header `<unistd.h>`. In addition, the presence of the C Language Development Option may be indicated by the subtype `POSIX_Profiles.Realtime_Lang_C99` having the range `True..True`.

8.5.2 Ada Language Development Option

If this option is provided, the implementor shall define a Development Platform and an environment capable of preparing for execution an application conformant with this profile including applicable portions of the following:

- The Ada95 RM {1}
- POSIX.5c {6}
- The `POSIX2_SW_DEV` option from the Shell and Utilities Volume of POSIX.1.

8.5.2.1 Option Indicator

The presence of the Ada Language Development Option shall be indicated by the subtype `POSIX_Profiles.Realtime_Lang_Ada95` having the range `True..True`. In addition, the presence of the Ada Language Development Option may be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_Ada95` being defined in the header `<unistd.h>`.

8.6 Rationale for Operating System Requirements (informative)

(This subclause is not a normative part of IEEE Std P1003.13)

8.6.1 Operating System Interface Requirements

This profile is based on existing practice in large embedded systems (a single user is assumed). Traditionally, these applications are designed to run with either a home-grown or standard operating system providing process, I/O, time, memory,

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 and event management services. These applications require support for a simpli-
2 fied file system.

3
4 Where convenient, the AEP profile working group has chosen system options that
5 allow an application to be upwardly portable without modifying application source
6 code.

9 **8.6.1.1 Process Primitives**

10
11 Applications that correspond to the Dedicated Realtime System Environment are
12 usually large embedded systems that require multiple processes for handling mul-
13 tiple, concurrent activities with independent address spaces. The process control
14 functions (which include process creation and execution) are the basic operating
15 system services required to support multiple processes, and are therefore required
16 in these systems.
17

19 **8.6.1.2 Signals**

20
21 Signal services are a basic mechanism within POSIX-based systems and are re-
22 quired for error and event handling. Realtime systems typically have several logi-
23 cally concurrent software elements executing. Each such entity must respond to
24 several cyclic and/or acyclic stimuli, often in a time-critical manner. Although
25 purely synchronous models can supply such functionality via the use of additional
26 processes or threads, the current realtime practice for asynchronous notification
27 for events such as timeout, message arrival, and hardware interrupt can generally
28 be expected to offer higher performance and lower latency. Realtime Signals pro-
29 vide the reliable high-performance mechanism to support such notification.
30

31
32 The minimum number of realtime signals that the implementation is required to
33 support has been increased from the number specified in the POSIX.1 standard, 8,
34 up to 16. The rationale for this increase is that there are many applications that
35 have more than 8 different kinds of events. Doubling the number of required real-
36 time signals should have a minimum impact on the signal management overhead,
37 while significantly increases the number of event kinds that can be used by a
38 strictly conforming application.
39

41 **8.6.1.3 Process Environment**

42
43 The functions from the POSIX.1 Process Environment group are deemed necessary
44 to allow an application to determine and configure its system environment. This
45 allows a single version of an application to be run on similar but differing
46 platforms.
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 Since these systems require multiple processes, but not users or groups, the func-
2 tions defined by the POSIX_MULTI_PROCESS unit of functionality are required.
3
4

5 **8.6.1.4 Files and Directories**

8 Since this profile has a file system, all POSIX.1 functions that manage a basic file
9 systems are required. However, the file system in a PSE53 platform is a simplified
10 version of a full POSIX.1 file system, and for this reason the POSIX_FIFO,
11 POSIX_FILE_ATTRIBUTES, and POSIX_FILE_SYSTEM_EXT, units of functionality
12 and the _POSIX_ADVISORY_INFO option are not required.
13

14 The File Locking option is required in the C-language option to maintain a consis-
15 tent and safe way of accessing stdio (*FILE* *) objects from threads, across the four
16 realtime profiles.

17 The File Descriptor Management unit of functionality is included to aid the han-
18 dling of file descriptors across the process creation and program execution opera-
19 tions.
20
21

22 **8.6.1.5 Input and Output Primitives**

25 The functions contained in the Device I/O unit of functionality are required to do
26 basic I/O and device cleanup.
27

28 Although asynchronous I/O can be easily implemented using threads dedicated to
29 I/O, it is required in the PSE53 profile to support portability of applications that
30 may have been developed before POSIX threads implementations were widely
31 available.
32
33

34 **8.6.1.6 Synchronized Input and Output**

37 The Synchronized (unbuffered) I/O interface (including the File Synchronization
38 option) is typical for basic device I/O and is required for upward portability.
39

40 Those realtime systems that use file management systems will frequently require
41 synchronized I/O to provide data integrity and/or relinquish resources to other us-
42 ers. Synchronized I/O as defined in POSIX.1 provides these mechanisms.
43
44
45
46
47
48
49

8.6.1.7 Device- and Class-Specific Functions

POSIX.1 Device- or Class-Specific functions are not required, because embedded systems usually don't require general-purpose terminal interfaces.

8.6.1.8 System Databases, Users and Groups

Implementations are not required to support more than one user and group id since there are not multiple users and groups. No POSIX.1 System Database functions are required.

8.6.1.9 Synchronization

Mutexes and Condition Variables are required as part of threads model of concurrency.

Semaphores are required to support portability of applications that might be using this mechanism instead of the preferred mutexes and condition variables. It must be noted, however, that POSIX semaphores do not have the mechanisms built in to avoid unbounded priority inversion when using them for mutually exclusive access to shared resources. Mutexes with the appropriate priority inheritance or priority ceiling (also called priority protection) protocols can be used to avoid this unbounded priority inversion. The Process Shared option is required to support applications requiring this mechanism for synchronization across different processes.

Barriers are not required because they can easily be implemented using mutexes and condition variables. Although a direct implementation of barriers can have a significant efficiency benefit in some multiprocessor architectures, a mutex-and-condition-variable implementation will not be significantly slower in most architectures, and thus requiring barriers for all implementations is not justified.

Spin locks are not required because, although they are an efficient synchronization mechanism, they cannot be portably used with the current POSIX.1 interfaces in realtime applications. If a realtime scheduling policy such as SCHED_FIFO or SCHED_RR is used, spin locks may cause deadlock on a single processor. On multiprocessors, to avoid deadlock, it would be necessary for threads using a given lock to be allocated to different processors. There are no standard APIs in the current POSIX.1 to allocate threads to specific processors.

Reader/Writer Locks are not required because they are not designed to avoid unbounded priority inversion, and thus very long delays could occur in realtime applications, with a low but nevertheless non-zero probability. It is expected that a future revision of the POSIX.1 standard will add the priority inheritance and/or

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 priority ceiling options to reader/writer locks, which would eliminate the unbound-
2 ed priority inversion.
3
4

5 **8.6.1.10 Priority Scheduling**

6

7
8 Thread and process priority scheduling are required for realtime applications. The
9 Sporadic Server Scheduling option is also required for processes and threads, to
10 enhance support of applications with aperiodic timing requirements. The
11 POSIX_PRIORITY_RANGES unit of functionality is not required because its func-
12 tions are already included in the required POSIX_PRIORITY_SCHEDULING op-
13 tion.
14

15 A common requirement of realtime systems is that they be able to run threads or
16 processes with real-time requirements together with threads with no real-time re-
17 quirements. One common way of doing this is by having the real-time threads run
18 under the SCHED_FIFO scheduling policy, while the non real-time threads run at
19 a lower priority under the round-robin policy (SCHED_RR) to fairly share the avail-
20 able portion of the processor among them. POSIX requires each policy to have a
21 range of priorities of at least 32 distinct values, but does not impose any require-
22 ments on how these priority ranges relate to each other. It could happen that most
23 or all of the SCHED_RR priorities were larger than the SCHED_FIFO priorities,
24 thus making it impossible to mix realtime and non-realtime threads as required
25 above. To solve this problem in a portable way, this profile requires that there are
26 at least 31 SCHED_RR priority levels below the maximum priority of SCHED_FIFO.
27 In this way, a strictly conforming application can use the inclusive priority range
28 [max_FIFO_prio , $max_FIFO_prio-30$] with SCHED_FIFO for real-time threads
29 (with a total of 31 priority levels), and then use the priority value
30 $min(max_FIFO_prio-31, max_RR_prio)$ with the SCHED_RR policy, for the non
31 real-time threads, with guarantee that the latter priority value is valid for the
32 round-robin policy.
33

34 The implementation is required to support the PTHREAD_SYSTEM_SCOPE thread-
35 scheduling contention scope. The contention scope of a thread defines the set of
36 threads with which the thread competes for use of the processing resources. A
37 thread created with PTHREAD_SCOPE_SYSTEM scheduling contention scope con-
38 tends for resources with all other threads in the system that have the same sched-
39 uling allocation domain. This allows a consistent scheduling of threads across the
40 system and therefore a predictable timing behavior. As a consequence, this is the
41 preferred method for realtime systems.

42 The current POSIX.1 specification allows implementations to support either sys-
43 tem-wide or process-wide contention scope, or both. This represents a compromise
44 that tries to address the requirements of both realtime and non-realtime applica-
45 tions, but introduces a potential source for non portability. Because the realtime
46 profiles are specifically targeted at realtime systems, the system-wide contention
47 scope option is required in the profiles that support multiple processes. Process-
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 wide contention scope may also be provided, perhaps for the non realtime threads
2 of the application.

3
4 Support for a scheduling allocation domain of size one and static binding of threads
5 to allocation domains is required in all the realtime profiles to achieve predictable
6 scheduling behavior. The allocation domain of a thread is the set of processors on
7 which that thread can be scheduled at any given time. The POSIX.1 standard spec-
8 ifies that the scheduling rules have predictable effects only if the allocation domain
9 is of size one; hence the need for this requirement. For single-processor systems the
10 allocation domain is generally of size one and thus the application can meet the re-
11 quirement just by specifying in the conformance document that the scheduling al-
12 location domain is of size one and that static binding of threads to allocation
13 domains is the default behavior.

14 15 16 **8.6.1.11 Process Memory Locking**

17
18
19 Realtime processes must be able to guarantee memory residency to reduce the la-
20 tency for instruction fetches, data access, I/O operations, etc. The mechanism de-
21 scribed in the POSIX.1 Process Memory Locking extension will satisfy this
22 requirement.

23 24 25 **8.6.1.12 Shared Memory**

26
27
28 The Shared Memory Objects option provides the capability for more than one exe-
29 cution entity to share memory, without incurring the overhead of the shared mem-
30 ory object on permanent media. Memory Mapped I/O may be implemented using
31 the Shared Memory facility. An implementation must provide facilities for creat-
32 ing a block of physical memory in which the application may place devices and fa-
33 cilities for binding to a user-provided pathname through which a device may
34 subsequently be opened as a Shared Memory special file, and mapped into the pro-
35 cess address space for the purpose of performing I/O or other functions from appli-
36 cations programs.

37
38 Typed Memory objects are not required because they are useful only to systems
39 with special hardware architectures that have various often specialized kinds of
40 memory. Implementors providing support for such special architectures always
41 have the option to provide typed memory objects as an extension.

42 43 44 **8.6.1.13 Clocks and Timers**

45
46
47 High-resolution timer functions are required in most realtime systems for imple-
48 menting time management operations such as periodic activations, short duration
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 time-outs, etc. The normal POSIX.1 time management functions *sleep()* and
2 *alarm()* only provide a time resolution of one second, but many realtime systems
3 require finer resolution for specifying time.

4 The Monotonic Clock is required for realtime applications to ensure that deadlines
5 and timing requirements are not affected by clock jumps.

7 The Clock Selection option is required to enable choosing the clock on which sleep
8 operations are performed, and to have access to an absolute sleep operation, which
9 is a common requirement in realtime applications with periodic timing require-
10 ments.

12 CPU-Time clocks and timers are required as a means to detect and handle situa-
13 tions in which a thread overruns its assigned maximum execution time. Delimiting
14 the execution times of the different threads in the application provides temporal
15 partitioning in realtime applications, and thus increases predictability and reli-
16 ability.

17 The Timeouts option is a general requirement for realtime applications and thus
18 is required in this profile.

20 The minimum number of per-process timers that the implementation is required
21 to support has been increased from the number specified in the POSIX.1 standard,
22 32, up to 64, which is the required minimum number of threads per process. The
23 reason for this increase is that there are many applications that require one timer
24 per thread (either realtime or CPU-time based).

27 **8.6.1.14 Message Passing**

30 These realtime systems typically include some form of message queuing mecha-
31 nism for communication among processes or threads. The POSIX.1 message pass-
32 ing offers an appropriate level of performance to provide this functionality.

35 **8.6.1.15 Threads**

38 The basic assumption in this profile is that the system will consist of one or more
39 processes with multiple threads. Therefore, all thread services are required. The
40 POSIX_THREADS_BASE unit of functionality was specified in this document in-
41 stead of the POSIX_THREADS option, because this option requires reader/writer
42 locks, but this profile does not.

44
45
46
47
48
49
Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

8.6.1.16 Tracing

Tracing is required for the PSE53 environment because most of these systems work in an unattended mode for long periods of time, and tracing provides an excellent mechanism to support post-failure analysis, particularly for failures having a low probability of occurrence.

The Trace Event Filtering option is required for the system to be able to filter out those trace events that are not meaningful for the application, thus making better use of system resources by capturing only the interesting events.

Because the PSE53 profile does not require general file system capabilities, the Trace Log option is not required for this profile.

8.6.1.17 Networking

Today, most of the platforms and applications belonging to the PSE53 environment require network communications, and thus the networking unit of functionality is required in this profile. The Raw Sockets option is required to aid reconfiguration of networked applications, and to implement special protocols directly, without the weight of a full protocol stack. The Internet Protocol Version 6 option is not required because most applications are not using this version of the protocol yet.

8.6.1.18 Event Management

The *select()* function is usually associated with networking facilities, which are required for PSE53, and thus the Event Management unit of functionality is required in the PSE53 environment.

8.6.1.19 Interfaces Related to the Shell and Utilities

Interfaces defined in the POSIX_REGEX and POSIX_SHELL_FUNC units of functionality are related to shells and utilities, which are not required in this profile; therefore, these units of functionality are not required either.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

8.6.1.20 X/Open Units of Functionality and Options

Some XSI Units of Functionality (XSI_C_LANG_SUPPORT, XSI_DEVICE_IO, XSI_DEVICE_SPECIFIC, XSI_FD_MGMT, XSI_FILE_SYSTEM, XSI_IPC, XSI_JOB_CONTROL, XSI_JUMP, XSI_MATH, XSI_MULTI_PROCESS, XSI_SIGNALS, XSI_SINGLE_PROCESS, XSI_SYSTEM_DATABASE, XSI_TIMERS, XSI_USER_GROUPS, XSI_WIDE_CHAR) have interfaces that represent extensions or alternatives to interfaces in other Units of Functionality or POSIX.1 options, and therefore are not necessary for PSE53 environments.

The XSI_DBM unit of functionality includes interfaces for database management that are not required in the PSE53 application environment.

The XSI_DYNAMIC_LINKING unit of functionality is not required for embedded systems, which usually operate in a static context.

The XSI_I18N unit of functionality provides facilities for natural language messages to the user, which are not required in embedded systems, which typically do not have general-purpose human interfaces.

The XSI_SYSTEM_LOGGING unit of functionality provides facilities for logging system activities, which are not required in PSE53 environments.

The XSI_THREAD_MUTEX_EXT unit of functionality is required because it has options for controlling the behavior of mutexes under erroneous application use. This capability is interesting for any realtime application, including those targeted at small embedded systems.

The XSI_THREADS_EXT unit of functionality is required because it provides functions to better control a thread's stack. This is considered useful for any realtime application.

The _XOPEN_CRYPT option provides cryptography facilities that are not required in most PSE53 environments.

The _XOPEN_LEGACY option provides facilities for backwards compatibility that are not required in PSE53 environments.

The _XOPEN_STREAMS option provides facilities that are not required in most PSE53 environments.

8.6.1.21 Language-Specific Services for the C Programming Language

Support for the C Language is required in the C language option, with the exception of the POSIX_C_LANG_WIDE_CHAR unit of functionality. The reason for this exception is that this is a very large library that is not necessary for many of the PSE53 applications.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

8.6.1.22 Language-Specific Services for the Ada Programming Language

Support for the Ada language-specific services defined in POSIX.5c is required in the Ada language option.

8.6.2 Shell and Utility Requirements

Because the Dedicated Realtime System Profile is intended for embedded systems which usually have no terminal or general-purpose graphical user interface, such a platform would be incapable of executing a shell. In such an environment the utilities described in the Shell and Utilities Volume of POSIX.1 are not usually required.

8.6.3 Development Platform Requirements

The embedded nature of the PSE53 execution platform makes it difficult to use as a development platform. Therefore, the implementation is required to define a development environment in which a PSE53 application can be prepared for execution on the target platform. The development platform depends on the language option chosen by the implementation.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Section 9: Multi-Purpose Realtime System Profile (PSE54)

9.1 Introduction

This section specifies those standards required for conformance to the Multi-Purpose Realtime System Profile option and, where applicable, the state of any options contained in those standards.

When a referenced standard specifies services beyond those required by the Multi-Purpose Realtime System Profile, only those services included in the specified Units of Functionality referenced by this profile shall be required (See Table 1-1 through Table 1-18). All the applicable definitions in POSIX.1 and/or POSIX.5c still apply.

9.1.1 Identification

For the C-Language implementation, symbolic names shall be used to specify the presence or absence of each option in this profile. Names reserved for use in this profile begin with the string `_POSIX_AEP_REALTIME_`. For the Ada Language implementation a set of `Boolean` subtypes contained in package `POSIX_Options` (defined in POSIX.5c, section 2.5) shall be used to specify the presence or absence of each option in this profile.

9.1.2 Conformance

Conformance to the Multi-Purpose Realtime System Profile option shall be indicated as follows:

- For the C-Language implementation the symbol `_POSIX_AEP_REALTIME_MULTI` being defined in the header `<unistd.h>`.
- For the Ada Language implementation the `Boolean` subtype `POSIX_Profiles.Realtime_Multi` subtype having the range `True..True`.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

9.1.3 Options

The presence or absence of optional features shall be indicated as follows:

- For the C-language implementation, if any of the following symbols are defined in the header `<unistd.h>`:

`_POSIX_AEP_REALTIME_LANG_C99`

`_POSIX_AEP_REALTIME_LANG_Ada95`

- For the Ada language implementation, if any of the following `Boolean` subtypes has the range `True..True`, then the corresponding option is supported:

`POSIX_Profiles.Realtime_Lang_C99`

`POSIX_Profiles.Realtime_Lang_Ada95`

9.2 Operating System Interface Requirements

9.2.1 POSIX.1 Requirements (C Language Option)

The Multi-Purpose Realtime System Profile implementation shall include interfaces as defined in POSIX.1 for the following Units of Functionality (see Table 1-1)

Table 9-1: POSIX.1 Units of Functionality Requirements

Unit of Functionality
<code>POSIX_C_LANG_JUMP</code>
<code>POSIX_C_LANG_MATH</code>
<code>POSIX_C_LANG_SUPPORT</code>
<code>POSIX_C_LANG_WIDE_CHAR</code>
<code>POSIX_DEVICE_IO</code>
<code>POSIX_DEVICE_SPECIFIC</code>
<code>POSIX_EVENT_MGMT</code>
<code>POSIX_FD_MGMT</code>
<code>POSIX_FIFO</code>
<code>POSIX_FILE_ATTRIBUTES</code>
<code>POSIX_FILE_LOCKING</code>
<code>POSIX_FILE_SYSTEM</code>
<code>POSIX_FILE_SYSTEM_EXT</code>
<code>POSIX_JOB_CONTROL</code>
<code>POSIX_MULTI_PROCESS</code>
<code>POSIX_NETWORKING</code>

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

Table 9-1: POSIX.1 Units of Functionality Requirements (Continued)

Unit of Functionality
POSIX_PIPE
POSIX_REGEX
POSIX_SHELL_FUNC
POSIX_SIGNALS
POSIX_SIGNAL_JUMP
POSIX_SINGLE_PROCESS
POSIX_STRING_MATCHING
POSIX_SYMBOLIC_LINKS
POSIX_SYSTEM_DATABASE
POSIX_THREADS_BASE
POSIX_USER_GROUPS
POSIX_WIDE_CHAR_IO
XSI_DYNAMIC_LINKING
XSI_SYSTEM_LOGGING
XSI_THREAD_MUTEX_EXT
XSI_THREADS_EXT

The Multi-Purpose Realtime System Profile implementation shall support the following options defined in POSIX.1, by defining the associated symbol with a value greater than zero:

Table 9-2: POSIX.1 Option Requirements

Option
_POSIX_ADVISORY_INFO
_POSIX_ASYNCHRONOUS_IO
_POSIX_CHOWN_RESTRICTED
_POSIX_CLOCK_SELECTION
_POSIX_CPUTIME
_POSIX_FSYNC
_POSIX_JOB_CONTROL
_POSIX_MAPPED_FILES
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MEMORY_PROTECTION
_POSIX_MESSAGE_PASSING
_POSIX_MONOTONIC_CLOCK
_POSIX_NO_TRUNC
_POSIX_PRIORITIZED_IO
_POSIX_PRIORITY_SCHEDULING
_POSIX_RAW_SOCKETS
_POSIX_REALTIME_SIGNALS
_POSIX_REGEX
_POSIX_SAVED_IDS
_POSIX_SEMAPHORES
_POSIX_SHARED_MEMORY_OBJECTS

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

Table 9-2: POSIX.1 Option Requirements (Continued)

Option
<code>_POSIX_SHELL</code>
<code>_POSIX_SPAWN</code>
<code>_POSIX_SPORADIC_SERVER</code>
<code>_POSIX_SYNCHRONIZED_IO</code>
<code>_POSIX_THREAD_ATTR_STACKADDR</code>
<code>_POSIX_THREAD_ATTR_STACKSIZE</code>
<code>_POSIX_THREAD_CPU_TIME</code>
<code>_POSIX_THREAD_PRIO_INHERIT</code>
<code>_POSIX_THREAD_PRIO_PROTECT</code>
<code>_POSIX_THREAD_PRIORITY_SCHEDULING</code>
<code>_POSIX_THREAD_PROCESS_SHARED</code>
<code>_POSIX_THREAD_SAFE_FUNCTIONS</code>
<code>_POSIX_THREAD_SPORADIC_SERVER</code>
<code>_POSIX_TIMEOUTS</code>
<code>_POSIX_TIMERS</code>
<code>_POSIX_TRACE</code>
<code>_POSIX_TRACE_EVENT_FILTER</code>
<code>_POSIX_TRACE_LOG</code>
<code>_POSIX_VDISABLE</code>

The type *off_t* shall be capable of storing any value contained in type *long*.

The minimum value of `_POSIX_NGROUPS_MAX` shall be at least 8.

The minimum value of `CHILD_MAX` shall be at least 25.

The value of `_POSIX_TIMER_MAX` shall be at least 64.

The value of `_POSIX_RTSIG_MAX` shall be at least 16.

The range of priorities associated with the `SCHED_RR` scheduling policy shall have at least 31 distinct values that are less than the maximum priority of the `SCHED_FIFO` policy.

An implementation conforming to PSE54 shall support the `PTHREAD_SCOPE_SYSTEM` scheduling contention scope. In addition, it may support `PTHREAD_SCOPE_PROCESS`. For a description of the scheduling contention scope see the System Interfaces volume of POSIX.1, Section 2.9.2, "Thread Scheduling".

An implementation conforming to PSE54 shall provide a mechanism to configure the system so that the scheduling allocation domain has size one, and so that the binding of threads to scheduling allocation domains remains static. The mechanism by which this requirement is achieved shall be implementation defined. In addition, a PSE54 implementation may provide other configurations or facilities to change the size of the allocation domain and the bindings of threads to allocation domains. For a description of the scheduling allocation domain see the System Interfaces volume of POSIX.1, Section 2.9.2, "Thread Scheduling".

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 **9.2.2 POSIX.5c Requirements (Ada Language Option)**
 2
 3

4 The Multi-Purpose Realtime System Profile implementation shall include inter-
 5 faces as defined in POSIX.5c for the following units of functionality (see Table 1-2
 6 through Table 1-18):
 7

8 **Table 9-3: POSIX.1 Units of Functionality Requirements**

Unit of Functionality
POSIX_ADA_LANG_SUPPORT
POSIX_DEVICE_IO
POSIX_DEVICE_SPECIFIC
POSIX_EVENT_MGMT
POSIX_FD_MGMT
POSIX_FIFO
POSIX_FILE_ATTRIBUTES
POSIX_FILE_SYSTEM
POSIX_JOB_CONTROL
POSIX_MULTI_PROCESS
POSIX_NETWORKING
POSIX_PIPE
POSIX_SIGNALS
POSIX_SINGLE_PROCESS
POSIX_SYSTEM_DATABASE
POSIX_USER_GROUPS

9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28 The Multi-Purpose Realtime System Profile implementation shall support the fol-
 29 lowing options defined in POSIX.5c, by defining the associated option subtypes to
 30 have the range True..True, with the exception of the Filename Truncation option
 31 for which the associated subtype shall have the range False..False:

32 **Table 9-4: POSIX.5c Option Requirements**

POSIX.5c Option
Asynchronous I/O
Change Owner Restriction
File Synchronization
Memory Mapped Files
Memory Locking
Memory Range Locking
Memory Protection
Message Queues
Filename Truncation
Prioritized I/O
Priority Process Scheduling
Realtime Signals
Saved IDs Support
Semaphores

33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

Table 9-4: POSIX.5c Option Requirements (Continued)

POSIX.5c Option
Shared Memory Objects
Synchronized I/O
Mutexes Supported
Mutex Priority Inheritance
Mutex Priority Ceiling
Process Shared
Timers

The service `POSIX_Terminal_Functions.Disable_Control_Character` shall not raise `POSIX_Error` with an error code of `Operation_Not_Implemented`.

`POSIX_Limits.Child_Processes_Maxima'First` shall be at least 25.

`POSIX_Limits.Groups_Maxima'First` shall be at least 8.

`POSIX_Limits.Timers_Maxima'First` shall be at least 64.

`POSIX_Limits.Realtime_Signals_Maxima'First` shall be at least 16.

Regarding task priority scheduling, the implementation shall support the following requirements from POSIX.5c and the Ada95 RM:

- The implementation shall support the priority model defined in the Ada95 RM, clause D.1, and the pragmas and package interfaces defined in the Ada95 RM, clauses D.2-D.5.
- The implementation shall meet the requirements of POSIX.5c, section 13.3.1.

Implementations of the PSE54 profile shall support the `POSIX_Profiles` package defined in Annex A of this standard.

Subprograms not supported by a given profile shall raise `POSIX_Error`, returning an error code of `Operation_Not_Supported`, except as noted otherwise.

All `Image` and `Value` functions that appear in the packages supported by a profile must be implemented.

Where an overloaded subprogram is required by a unit of functionality, all forms of the subprogram appearing in the referenced clause must be supported, except as otherwise noted.

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 **9.3 Application Constraints**

2
3
4 The Multi-Purpose Realtime System profile defined in this standard requires only
5 specific units of functionality of the required standards. The absence of particular
6 elements of these standards introduces constraints on the use of some of the fea-
7 tures of particular operations. This clause defines the constraints that an applica-
8 tion strictly conforming to one of the profiles shall observe when using each of the
9 operations required by that profile.

10
11
12 **9.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)**

13
14
15 This profile has no constraints on the application related to POSIX.1 interfaces, be-
16 cause it requires the implementation to be POSIX.1 conforming.

17
18
19 **9.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option)**

20
21
22 An application strictly conforming to PSE54 shall not attempt to bind a signal to a
23 task entry.

24
25
26
27
28 **9.4 Shell and Utility Requirements**

29
30
31 An implementation of the Multi-Purpose Realtime System Profile shall provide all
32 the mandatory utilities in the Shell and Utilities volume of POSIX.1 with all the
33 functional behavior described therein. The system shall support the Large File ca-
34 pabilities described in the Shell and Utilities volume of POSIX.1.

35
36 If the C Language Option is supported, the following options of the Shell and Util-
37 ities volume of POSIX.1 shall be supported:

38
39 **Table 9-5: Shell and Utilities Option Requirements**
40 **(C Language Option)**

41

Option
POSIX2_C_BIND
POSIX2_CDEV
POSIX2_CHAR_TERM
POSIX2_FORT_RUN
POSIX2_SW_DEV
POSIX2_UPE

42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

If the Ada Language Option is supported, the following options of the Shell and Utilities volume of POSIX.1 shall be supported:

**Table 9-6: Shell and Utilities Option Requirements
(Ada Language Option)**

Option
POSIX2_CHAR_TERM
POSIX2_FORT_RUN
POSIX2_SW_DEV
POSIX2_UPE

9.5 Development Platform Requirements

One or more of the development options in 9.5.1 and 9.5.2 shall be implemented.

9.5.1 C Language Development Option

If this option is provided, the implementor shall define a Development Platform and an environment capable of preparing for execution an application conformant with this standard profile. This platform shall include the POSIX2_C_BIND, POSIX2_C_DEV, and POSIX2_SW_DEV options from the Shell and Utilities Volume of POSIX.1.

9.5.1.1 Option Indicator

The presence of the C Language Development Option shall be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_C99` being defined in the required header `<unistd.h>`. In addition, the presence of the C Language Development Option may be indicated by the subtype `POSIX_Profiles.Realtime_Lang_C99` having the range `True..True`.

9.5.2 Ada Language Development Option

If this option is provided, the implementor shall define a Development Platform and an environment capable of preparing for execution an application conformant with this profile including applicable portions of the following:

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 1 — The Ada95 RM {1}
- 2
- 3 — POSIX.5c {6}
- 4
- 5 — The POSIX2_SW_DEV option from the Shell and Utilities Volume of
- 6 POSIX.1.

7
8
9

9.5.2.1 Option Indicator

10
11
12
13
14
15
16
17
18
19

The presence of the Ada Language Development Option shall be indicated by the subtype `POSIX_Profiles.Realtime_Lang_Ada95` having the range `True..True`. In addition, the presence of the Ada Language Development Option may be indicated by the symbol `_POSIX_AEP_REALTIME_LANG_Ada95` being defined in the header `<unistd.h>`.

9.6 Rationale for Operating System Requirements (informative)

20
21
22
23

(This subclause is not a normative part of IEEE Std P1003.13)

24
25
26
27

9.6.1 Operating System Interface Requirements

28
29
30
31
32
33
34
35
36
37
38

This profile is based on existing practice in real-time systems that are built using general-purpose computers, such as workstations. These systems have general-purpose computing requirements such as a full featured file system, networking, virtual memory management, graphical user interfaces, multi-user access control, etc. In addition, they have real-time requirements, and thus the need for a real-time operating system that provides a full POSIX.1 implementation and also the realtime extensions described in this profile.

9.6.1.1 Process Primitives

39
40
41
42
43
44
45
46
47
48
49

The process control functions (which include process creation and execution) are the basic operating system services required to support multiple processes, and are therefore required by both realtime and non-realtime applications in these real-time systems.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

9.6.1.2 Signals

Signal services are a basic mechanism within POSIX-based systems and are required for error and event handling. Realtime systems typically have several logically concurrent software elements executing. Each such entity must respond to several cyclic and/or acyclic stimuli, often in a time-critical manner. Although purely synchronous models can supply such functionality via the use of additional processes or threads, the current realtime practice for asynchronous notification for events such as timeout, message arrival, and hardware interrupt can generally be expected to offer higher performance and lower latency. Realtime Signals provide the reliable high-performance mechanism to support such notification.

The minimum number of realtime signals that the implementation is required to support has been increased from the number specified in the POSIX.1 standard, 8, up to 16. The rationale for this increase is that there are many applications that have more than 8 different kinds of events. Doubling the number of required realtime signals should have a minimum impact on the signal management overhead, while significantly increases the number of event kinds that can be used by a strictly conforming application.

9.6.1.3 Process Environment

The functions from the POSIX.1 Process Environment group are deemed necessary to allow an application to determine and configure its system environment. This allows a single version of an application to be run on similar but differing platforms.

Since the systems will require multiple processes and multiple users, and because they must support both commercial-off-the-shelf (COTS) and realtime applications, the entire set of ID functions is needed.

9.6.1.4 Files and Directories

All file and directory operations are required to support system applications and their filesystems. Although only a few of the path operation functions are required to support realtime activities, the whole set is required for systems that support COTS applications.

The Advisory Information option is required to allow the application to provide hints about the way in which is going to perform file operations, so that implementations can provide a better degree of timing predictability for those operations.

The File Locking option is required in the C-language option to maintain a consistent and safe way of accessing `stdio` (*FILE* *) objects from threads, across the four realtime profiles.

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 The File Descriptor Management unit of functionality is included to aid the han-
2 dling of file descriptors across the process creation and program execution opera-
3 tions.
4

6 **9.6.1.5 Input and Output Primitives**

7

8
9 The functions contained in the Device I/O unit of functionality are required to sup-
10 port I/O on devices, files, and special files.
11

12 Although asynchronous I/O can be easily implemented using threads dedicated to
13 I/O, it is required in the PSE54 profile to support portability of applications that
14 may have been developed before POSIX threads implementations were widely
15 available.
16

17 18 **9.6.1.6 Synchronized Input and Output**

19

20
21 These realtime systems that use file management systems will frequently require
22 synchronized I/O to provide data integrity and/or relinquish resources to other pro-
23 cesses. Synchronized I/O as defined in POSIX.1 provides these mechanisms.
24

25 26 **9.6.1.7 Device- and Class-Specific Functions**

27

28
29 The terminal control functions are required for systems to support COTS applica-
30 tions and for the standard terminal devices that may be attached to the computer
31 system. To support non-standard terminal devices, additional functions may be
32 necessary.
33

34 35 **9.6.1.8 System Databases, Users and Groups**

36

37
38 The group and user database access functions are required for COTS database ap-
39 plications that may require them.
40

41 42 **9.6.1.9 Synchronization**

43

44
45 Mutexes and Condition Variables are required as part of threads model of concur-
46 rency.
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 Semaphores are required to synchronize a signal handler with some other process
2 or thread. Semaphores are also required to support portability of applications that
3 might be using this mechanism instead of the preferred mutexes and condition
4 variables. It must be noted, however, that POSIX semaphores do not have the
5 mechanisms built in to avoid unbounded priority inversion when using them for
6 mutually exclusive access to shared resources. Mutexes with the appropriate pri-
7 ority inheritance or priority ceiling (also called priority protection) protocols can be
8 used to avoid this unbounded priority inversion. The Process Shared option is re-
9 quired to support applications requiring this mechanism for synchronization
10 across different processes.

11 Barriers are not required because they can easily be implemented using mutexes
12 and condition variables. Although a direct implementation of barriers can have a
13 significant efficiency benefit in some multiprocessor architectures, a mutex-and-
14 condition-variable implementation will not be significantly slower in most archi-
15 tectures, and thus requiring barriers for all implementations is not justified.

17 Spin locks are not required because, although they are an efficient synchronization
18 mechanism, they cannot be portably used with the current POSIX.1 interfaces in
19 realtime applications. If a realtime scheduling policy such as SCHED_FIFO or
20 SCHED_RR is used, spin locks may cause deadlock on a single processor. On mul-
21 tiprocessors, to avoid deadlock, it would be necessary for threads using a given lock
22 to be allocated to different processors. There are no standard APIs in the current
23 POSIX.1 to allocate threads to specific processors.

25 Reader/Writer Locks are not required because they are not designed to avoid un-
26 bounded priority inversion, and thus very long delays could occur in realtime ap-
27 plications, with a low but nevertheless non-zero probability. It is expected that a
28 future revision of the POSIX.1 standard will add the priority inheritance and/or
29 priority ceiling options to reader/writer locks, which would eliminate the unbound-
30 ed priority inversion.

33 **9.6.1.10 Priority Scheduling**

36 This realtime environment requires the ability to do scheduling of concurrent pro-
37 cesses and threads with a preemptive priority-based scheduler to ensure that hard
38 deadlines are met. Thread and process priority scheduling are required for real-
39 time applications. The Sporadic Server Scheduling option is also required for pro-
40 cesses and threads, to enhance support of applications with aperiodic timing
41 requirements. The POSIX_PRIORITY_RANGES unit of functionality is not re-
42 quired because its functions are already included in the required
43 _POSIX_PRIORITY_SCHEDULING option.

45 A common requirement of realtime systems is that they be able to run threads or
46 processes with real-time requirements together with threads with no real-time re-
47 quirements. One common way of doing this is by having the real-time threads run
48 under the SCHED_FIFO scheduling policy, while the non real-time threads run at
49 a lower priority under the round-robin policy (SCHED_RR) to fairly share the avail-

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 able portion of the processor among them. POSIX requires each policy to have a
2 range of priorities of at least 32 distinct values, but does not impose any require-
3 ments on how these priority ranges relate to each other. It could happen that most
4 or all of the SCHED_RR priorities were larger than the SCHED_FIFO priorities,
5 thus making it impossible to mix realtime and non-realtime threads as required
6 above. To solve this problem in a portable way, this profile requires that there are
7 at least 31 SCHED_RR priority levels below the maximum priority of SCHED_FIFO.
8 In this way, a strictly conforming application can use the inclusive priority range
9 [*max_FIFO_prio*, *max_FIFO_prio*-30] with SCHED_FIFO for real-time threads
10 (with a total of 31 priority levels), and then use the priority value
11 $\min(\text{max_FIFO_prio}-31, \text{max_RR_prio})$ with the SCHED_RR policy, for the non
12 real-time threads, with guarantee that the latter priority value is valid for the
13 round-robin policy.

14
15 The implementation is required to support the PTHREAD_SCOPE_SYSTEM thread-
16 scheduling contention scope. The contention scope of a thread defines the set of
17 threads with which the thread competes for use of the processing resources. A
18 thread created with PTHREAD_SCOPE_SYSTEM scheduling contention scope con-
19 tends for resources with all other threads in the system that have the same sched-
20 uling allocation domain. This allows a consistent scheduling of threads across the
21 system and therefore a predictable timing behavior. As a consequence, this is the
22 preferred method for realtime systems.

23 The current POSIX.1 specification allows implementations to support either sys-
24 tem-wide or process-wide contention scope, or both. This represents a compromise
25 that tries to address the requirements of both realtime and non-realtime applica-
26 tions, but introduces a potential source for non portability. Because the realtime
27 profiles are specifically targeted at realtime systems, the system-wide contention
28 scope option is required in the profiles that support multiple processes. Process-
29 wide contention scope may also be provided, perhaps for the non realtime threads
30 of the application.

31
32 Support for a scheduling allocation domain of size one and static binding of threads
33 to allocation domains is required in all the realtime profiles to achieve predictable
34 scheduling behavior. The allocation domain of a thread is the set of processors on
35 which that thread can be scheduled at any given time. The POSIX.1 standard spec-
36 ifies that the scheduling rules have predictable effects only if the allocation domain
37 is of size one; hence the need for this requirement. For single-processor systems the
38 allocation domain is generally of size one and thus the application can meet the re-
39 quirement just by specifying in the conformance document that the scheduling al-
40 location domain is of size one and that static binding of threads to allocation
41 domains is the default behavior.

42 43 44 **9.6.1.11 Process Memory Locking**

45
46
47 Realtime processes must be able to guarantee memory residency to reduce the la-
48 tency for instruction fetches, data access, I/O operations, etc. The mechanism de-
49

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1 scribed in the POSIX.1 Process Memory Locking extension will satisfy this
2 requirement.
3
4

5 **9.6.1.12 Shared Memory**

6
7

8 The ability to share large volumes of data among many cooperating execution
9 streams is required. The POSIX.1 Shared Memory extension provides this capabil-
10 ity. Memory Mapped I/O may be implemented using the Shared Memory facility.
11 An implementation must provide facilities for creating a block of physical memory
12 in which the application may place devices and facilities for binding to a user-pro-
13 vided pathname through which a device may subsequently be opened as a Shared
14 Memory special file, and mapped into the process address space for the purpose of
15 performing I/O or other functions from applications programs.
16

17 The Memory Mapped Files is required because the implementation has file-system
18 capabilities, and memory-mapped files are a convenient paradigm for reading and
19 writing information in applications following this profile. In memory-mapped files,
20 data can be manipulated as memory, and I/O data movement can be significantly
21 reduced. The implementation of memory-mapped files does not require a signifi-
22 cant amount of additional memory or execution overhead to achieve the additional
23 capability.
24

25 System vendors are expected to implement the chosen interface in a manner that
26 meets the needs of the applications. In particular, a rotating media-based imple-
27 mentation is not required by the interface definition.

28 Typed Memory objects are not required because they are useful only to systems
29 with special hardware architectures that have various often specialized kinds of
30 memory. Implementors providing support for such special architectures always
31 have the option to provide typed memory objects as an extension.
32
33

34 **9.6.1.13 Clocks and Timers**

35
36

37 High-resolution timer functions are required in most realtime systems for imple-
38 menting time management operations such as periodic activations, short duration
39 time-outs, etc. The normal POSIX.1 time management functions *sleep()* and
40 *alarm()* only provide a time resolution of one second, but many realtime systems
41 require finer resolution for specifying time.
42

43 The Monotonic Clock is required for realtime applications to ensure that deadlines
44 and timing requirements are not affected by clock jumps.

45 The Clock Selection option is required to enable choosing the clock on which sleep
46 operations are performed, and to have access to an absolute sleep operation, which
47 is a common requirement in realtime applications with periodic timing require-
48 ments.
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 CPU-Time clocks and timers are required as a means to detect and handle situa-
2 tions in which a thread overruns its assigned maximum execution time. Delimiting
3 the execution times of the different threads in the application provides temporal
4 partitioning in realtime applications, and thus increases predictability and reli-
5 ability.

6
7 The Timeouts option is a general requirement for realtime applications and thus
8 is required in this profile.

9 The minimum number of per-process timers that the implementation is required
10 to support has been increased from the number specified in the POSIX.1 standard,
11 32, up to 64, which is the required minimum number of threads per process. The
12 reason for this increase is that there are many applications that require one timer
13 per thread (either realtime or CPU-time based).
14

15 16 **9.6.1.14 Message Passing**

17
18
19 These realtime systems typically include some form of message queuing mecha-
20 nism for communication among processes or threads. The POSIX.1 message pass-
21 ing offers an appropriate level of performance to provide this functionality.
22

23 24 **9.6.1.15 Threads**

25
26
27 The basic assumption in this profile is that the system will consist of one or more
28 processes with multiple threads. Therefore, all thread services are required. The
29 POSIX_THREADS_BASE unit of functionality was specified in this document in-
30 stead of the _POSIX_THREADS option, because this option requires reader/writer
31 locks, but this profile does not.
32

33 34 **9.6.1.16 Tracing**

35
36
37 Tracing is required for the PSE54 environment because it provides an excellent
38 mechanism to support post-failure analysis, particularly for failures having a low
39 probability of occurrence.
40

41 The Trace Event Filtering option is required for the system to be able to filter out
42 those trace events that are not meaningful for the application, thus making better
43 use of system resources by capturing only the interesting events.
44

45 Because the PSE54 profile requires general file system capabilities, the Trace Log
46 option is required for this profile.
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

9.6.1.17 Networking

Today, virtually all of the platforms and applications belonging to the PSE54 environment require network communications, and thus the networking unit of functionality is required in this profile. The Raw Sockets option is required to aid reconfiguration of networked applications, and to implement special protocols directly, without the weight of a full protocol stack. The Internet Protocol Version 6 option is not required because most applications are not using this version of the protocol yet.

9.6.1.18 Event Management

The *select()* function is usually associated with networking facilities, which are required for PSE54, and thus the Event Management unit of functionality is required in the PSE54 environment.

9.6.1.19 Interfaces Related to the Shell and Utilities

The interfaces defined in the POSIX_REGEX and POSIX_SHELL_FUNC are required in PSE54 environments, because of their general-purpose computing requirements.

9.6.1.20 X/Open Units of Functionality and Options

Some XSI Units of Functionality (XSI_C_LANG_SUPPORT, XSI_DEVICE_IO, XSI_DEVICE_SPECIFIC, XSI_FD_MGMT, XSI_FILE_SYSTEM, XSI_IPC, XSI_JOB_CONTROL, XSI_JUMP, XSI_MATH, XSI_MULTI_PROCESS, XSI_SIGNALS, XSI_SINGLE_PROCESS, XSI_SYSTEM_DATABASE, XSI_TIMERS, XSI_USER_GROUPS, XSI_WIDE_CHAR) have interfaces that represent extensions or alternatives to interfaces in other Units of Functionality or POSIX.1 options, and therefore are not necessary for PSE54 environments.

The XSI_DBM unit of functionality includes interfaces for database management that are not required in the PSE54 application environment.

The XSI_DYNAMIC_LINKING unit of functionality is required for PSE54 systems, which usually execute a mixture of realtime and non realtime activities in a typically dynamic context.

The XSI_I18N unit of functionality provides facilities for natural language messages to the user, which are not required all PSE54 systems. It remains as an optional feature.

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 The XSI_SYSTEM_LOGGING unit of functionality provides facilities for logging sys-
2 tem activities, which are usually required in PSE54 environments. Therefore, this
3 unit of functionality is required.

4 The XSI_THREAD_MUTEX_EXT unit of functionality is required because it has op-
5 tions for controlling the behavior of mutexes under erroneous application use. This
6 capability is interesting for any realtime application, including those targeted at
7 small embedded systems.
8

9 The XSI_THREADS_EXT unit of functionality is required because it provides func-
10 tions to better control a thread's stack. This is considered useful for any realtime
11 application.
12

13 The _XOPEN_CRYPT option provides cryptography facilities that are not required
14 in all PSE54 environments. It remains as an optional feature.

15 The _XOPEN_LEGACY option provides facilities for backwards compatibility that
16 are not required in most PSE54 environments.
17

18 The _XOPEN_STREAMS option provides facilities that are not required in most
19 PSE54 environments.
20

21 22 **9.6.1.21 Language-Specific Services for the C Programming Language**

23
24
25 Full support for the C Language standard is required in the C language option.
26
27

28 **9.6.1.22 Language-Specific Services for the Ada Programming Language**

29
30
31 Support for the Ada language-specific services defined in POSIX.5c is required in
32 the Ada language option.
33
34
35

36 **9.6.2 Shell and Utility Requirements**

37
38
39 The utilities and facilities described in the Shell and Utilities Volume of POSIX.1
40 are required in PSE54 environments.
41
42
43

44 **9.6.3 Development Platform Requirements**

45
46
47 The implementation is required to define a development environment in which a
48 PSE54 application can be prepared for execution on the target platform. For this
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1 profile, in most cases the development and the target platform roles will be com-
2 bined in the same system. |
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Annex A: POSIX Profiles Package (Ada Language)

(Normative)

The package `POSIX_Profiles` shall be supported by all profiles. The Boolean subtypes contained in this package shall indicate the profiles and options supported by the implementation. Supported profiles and options shall be indicated by the appropriate identifier having the range `True..True`; unsupported profiles and options shall have the range `False..False`.

```

package POSIX_Profiles is
  -- Profile options
  subtype Realtime_Minimal is Boolean range <Implementation Defined>;
  subtype Realtime_Controller is Boolean range <Implementation Defined>;
  subtype Realtime_Dedicated is Boolean range <Implementation Defined>;
  subtype Realtime_Multi is Boolean range <Implementation Defined>;

  -- Language development options
  subtype Realtime_Lang_C99 is Boolean range <Implementation Defined>;
  subtype Realtime_Lang_Ada95 is Boolean range <Implementation Defined>;

end POSIX_Profiles;

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Annex B: Description of Optional Interfaces (Informative)

B.1 POSIX.1 Options

The following table shows the functions included under each of the options specified in the System Interfaces volume of POSIX.1. Each row of this table contains all the functions included under the first named option, and also under combinations of that option with other options.

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

<p><u>_POSIX_ADVISORY_INFO</u> <i>posix_fadvise(), posix_fallocate(), posix_memalign()</i></p> <p><u>_POSIX_ADVISORY_INFO</u> and either <u>_POSIX_MAPPED_FILES</u> or <u>_POSIX_SHARED_MEMORY_OBJECTS</u> <i>posix_madvise()</i></p>
<p><u>_POSIX_ASYNCHRONOUS_IO</u> <i>aio_cancel(), aio_error(), aio_fsync(), aio_read(), aio_return(), aio_suspend(), aio_write(), lio_listio()</i></p>
<p><u>_POSIX_BARRIERS</u> and <u>_POSIX_THREADS</u> <i>pthread_barrier_destroy(), pthread_barrier_init(), pthread_barrier_wait(), pthread_barrierattr_destroy(), pthread_barrierattr_init(),</i></p> <p><u>POSIX_BARRIERS</u>, <u>POSIX_THREADS</u> and <u>POSIX_THREAD_PROCESS_SHARED</u> <i>pthread_barrierattr_getpshared(), pthread_barrierattr_setpshared()</i></p>
<p><u>_POSIX_CHOWN_RESTRICTED</u> No functions under this option</p>
<p><u>_POSIX_CLOCK_SELECTION</u> <i>clock_nanosleep()</i></p> <p><u>_POSIX_CLOCK_SELECTION</u> and <u>_POSIX_THREADS</u> <i>pthread_condattr_getclock(), pthread_condattr_setclock()</i></p>
<p><u>_POSIX_CPUTIME</u> <i>clock_getcpuclockid()</i></p>

**Table B-1: Functions under each POSIX.1
System Interface Option (Continued)**

1	
2	
3	
4	_POSIX_FSYNC
5	<i>fsync()</i>
6	_POSIX_IPV6
7	No functions under this option
8	_POSIX_JOB_CONTROL
9	See the POSIX_JOB_CONTROL unit of functionality
10	_POSIX_MAPPED_FILES or _POSIX_SHARED_MEMORY_OBJECTS
11	<i>mmap(), munmap()</i>
12	_POSIX_MAPPED_FILES and _POSIX_SYNCHRONIZED_IO
13	<i>msync()</i>
14	_POSIX_MAPPED_FILES and _POSIX_ADVISORY_INFO
15	<i>posix_madvise()</i>
16	_POSIX_MEMLOCK
17	<i>mlockall(), munlockall()</i>
18	_POSIX_MEMLOCK_RANGE
19	<i>mlock(), munlock()</i>
20	_POSIX_MEMORY_PROTECTION
21	<i>mprotect()</i>
22	_POSIX_MESSAGE_PASSING
23	<i>mq_close(), mq_getattr(), mq_notify(), mq_open(), mq_receive(), mq_send(),</i>
24	<i>mq_setattr(), mq_unlink(),</i>
25	_POSIX_MESSAGE_PASSING and _POSIX_TIMEOUTS
26	<i>mq_timedreceive(), mq_timedsend()</i>
27	_POSIX_MONOTONIC_CLOCK
28	No functions under this option
29	_POSIX_NO_TRUNC
30	No functions under this option
31	_POSIX_PRIORITIZED_IO
32	No functions under this option
33	_POSIX_PRIORITY_SCHEDULING
34	<i>sched_get_priority_max(), sched_get_priority_min(), sched_getparam(),</i>
35	<i>sched_getscheduler(), sched_rr_get_interval(), sched_setparam(),</i>
36	<i>sched_setscheduler()</i>
37	_POSIX_PRIORITY_SCHEDULING or _POSIX_THREADS
38	<i>sched_yield(),</i>
39	_POSIX_PRIORITY_SCHEDULING and _POSIX_SPAWN
40	<i>posix_spawnattr_getschedparam(), posix_spawnattr_setschedparam(),</i>
41	<i>posix_spawnattr_getschedpolicy(), posix_spawnattr_setschedpolicy()</i>
42	
43	
44	
45	
46	
47	
48	
49	

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

<u>_POSIX_RAW_SOCKETS</u> No functions under this option
<u>_POSIX_READER_WRITER_LOCKS</u> See the POSIX_RW_LOCKS unit of functionality
<u>_POSIX_REALTIME_SIGNALS</u> <i>sigqueue(), sigtimedwait(), sigwaitinfo()</i>
<u>_POSIX_REGEX</u> See POSIX_REGEX unit of functionality.
<u>_POSIX_SAVED_IDS</u> No functions under this option
<u>_POSIX_SEMAPHORES</u> <i>sem_close(), sem_destroy(), sem_getvalue(), sem_init(), sem_open(), sem_post(), sem_trywait(), sem_wait(), sem_unlink()</i>
<u>_POSIX_SEMAPHORES and _POSIX_TIMEOUTS</u> <i>sem_timedwait()</i>
<u>_POSIX_SHARED_MEMORY_OBJECTS</u> <i>shm_open(), shm_unlink()</i>
<u>_POSIX_SHARED_MEMORY_OBJECTS and _POSIX_ADVISORY_INFO</u> <i>posix_madvise()</i>
<u>_POSIX_SHARED_MEMORY_OBJECTS or _POSIX_MAPPED_FILES</u> <i>mmap(), munmap()</i>
<u>_POSIX_SPAWN</u> <i>posix_spawn(), posix_spawn_file_actions_addclose(), posix_spawn_file_actions_adddup2(), posix_spawn_file_actions_addopen(), posix_spawn_file_actions_destroy(), posix_spawn_file_actions_init(), posix_spawnattr_destroy(), posix_spawnattr_getflags(), posix_spawnattr_getpgroup(), posix_spawnattr_getsigdefault(), posix_spawnattr_getsigmask(), posix_spawnattr_init(), posix_spawnattr_setflags(), posix_spawnattr_setpgroup(), posix_spawnattr_setsigdefault(), posix_spawnattr_setsigmask(), posix_spawnnp()</i>
<u>_POSIX_SPAWN and _POSIX_PRIORITY_SCHEDULING</u> <i>posix_spawnattr_getschedparam(), posix_spawnattr_setschedparam(), posix_spawnattr_getschedpolicy(), posix_spawnattr_setschedpolicy()</i>
<u>_POSIX_SPIN_LOCKS and _POSIX_THREADS</u> <i>pthread_spin_destroy(), pthread_spin_init(), pthread_spin_lock(), pthread_spin_trylock(), pthread_spin_unlock()</i>
<u>_POSIX_SPORADIC_SERVER</u> No functions under this option

**Table B-1: Functions under each POSIX.1
System Interface Option (Continued)**

1	
2	
3	
4	<u>_POSIX_SYNCHRONIZED_IO</u>
5	<i>fdatasync()</i>
6	<u>_POSIX_SYNCHRONIZED_IO and _POSIX_MAPPED_FILES</u>
7	<i>msync()</i>
8	<u>_POSIX_THREAD_ATTR_STACKADDR and _POSIX_THREADS</u>
9	<i>pthread_attr_getstackaddr(), pthread_attr_setstackaddr()</i>
10	
11	<u>_POSIX_THREAD_ATTR_STACKADDR, _POSIX_THREADS and</u>
12	<u>_POSIX_THREAD_ATTR_STACKSIZE</u>
13	<i>pthread_attr_getstack(), pthread_attr_setstack()</i>
14	<u>_POSIX_THREAD_ATTR_STACKSIZE and _POSIX_THREADS</u>
15	<i>pthread_attr_getstacksize(), pthread_attr_setstacksize()^a</i>
16	
17	<u>_POSIX_THREAD_ATTR_STACKSIZE, _POSIX_THREADS and</u>
18	<u>_POSIX_THREAD_ATTR_STACKADDR</u>
19	<i>pthread_attr_getstack(), pthread_attr_setstack()</i>
20	<u>_POSIX_THREAD_CPUTIME and _POSIX_THREADS</u>
21	<i>pthread_getcpuclockid()</i>
22	<u>_POSIX_THREAD_PRIO_INHERIT and _POSIX_THREADS</u>
23	<i>pthread_mutexattr_getprotocol(), pthread_mutexattr_setprotocol()</i>
24	
25	<u>_POSIX_THREAD_PRIO_PROTECT and _POSIX_THREADS</u>
26	<i>pthread_mutex_getprioceiling(), pthread_mutex_setprioceiling(),</i>
27	<i>pthread_mutexattr_getprioceiling(), pthread_mutexattr_getprotocol(),</i>
28	<i>pthread_mutexattr_setprioceiling(), pthread_mutexattr_setprotocol()</i>
29	<u>_POSIX_THREAD_PRIORITY_SCHEDULING and _POSIX_THREADS</u>
30	<i>pthread_attr_getinheritsched(), pthread_attr_getschedpolicy(),</i>
31	<i>pthread_attr_getscope(), pthread_attr_setinheritsched(),</i>
32	<i>pthread_attr_setschedpolicy(), pthread_attr_setscope(),</i>
33	<i>pthread_getschedparam(), pthread_setschedparam(), pthread_setschedprio()</i>
34	<u>_POSIX_THREAD_PROCESS_SHARED and _POSIX_THREADS</u>
35	<i>pthread_condattr_getpshared(), pthread_condattr_setpshared(),</i>
36	<i>pthread_mutexattr_getpshared(), pthread_mutexattr_setpshared()</i>
37	<u>_POSIX_THREAD_PROCESS_SHARED, _POSIX_BARRIERS and _POSIX_THREADS</u>
38	<i>pthread_barrierattr_getpshared(), pthread_barrierattr_setpshared()</i>
39	
40	<u>_POSIX_THREAD_PROCESS_SHARED, _POSIX_READER_WRITER_LOCKS and</u>
41	<u>_POSIX_THREADS</u>
42	<i>pthread_rwlockattr_getpshared(), pthread_rwlockattr_setpshared()</i>
43	<u>_POSIX_THREAD_SAFE_FUNCTIONS</u>
44	<i>asctime_r(), ctime_r(), flockfile(), ftrylockfile(), funlockfile(), getc_unlocked(),</i>
45	<i>getchar_unlocked(), getgrgid_r(), getgrnam_r(), getlogin_r(), getpwnam_r(),</i>
46	<i>getpwuid_r(), gmtime_r(), localtime_r(), putc_unlocked(), putchar_unlocked(),</i>
47	<i>rand_r(), readdir_r(), strerror_r(), strtok_r(), ttyname_r()</i>
48	
49	

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

1	
2	
3	
4	<u>POSIX_THREAD_SPORADIC_SERVER</u>
5	No functions under this option
6	<u>POSIX_THREADS</u>
7	<i>pthread_atfork(), pthread_attr_destroy(), pthread_attr_getdetachstate(),</i>
8	<i>pthread_attr_getschedparam(), pthread_attr_init(),</i>
9	<i>pthread_attr_setdetachstate(), pthread_attr_setschedparam(), pthread_cancel(),</i>
10	<i>pthread_cleanup_pop(), pthread_cleanup_push(), pthread_cond_broadcast(),</i>
11	<i>pthread_cond_destroy(), pthread_cond_init(), pthread_cond_signal(),</i>
12	<i>pthread_cond_timedwait(), pthread_cond_wait(), pthread_condattr_destroy(),</i>
13	<i>pthread_condattr_init(), pthread_create(), pthread_detach(), pthread_equal(),</i>
14	<i>pthread_exit(), pthread_getspecific(), pthread_join(), pthread_key_create(),</i>
15	<i>pthread_key_delete(), pthread_kill(), pthread_mutex_destroy(),</i>
16	<i>pthread_mutex_init(), pthread_mutex_lock(), pthread_mutex_trylock(),</i>
17	<i>pthread_mutex_unlock(), pthread_mutexattr_destroy(),</i>
18	<i>pthread_mutexattr_init(), pthread_once(), pthread_self(),</i>
19	<i>pthread_setcancelstate(), pthread_setcanceltype(), pthread_setspecific(),</i>
20	<i>pthread_sigmask(), pthread_testcancel()</i>
21	<u>POSIX_THREADS and POSIX_CLOCK_SELECTION</u>
22	<i>pthread_condattr_getclock(), pthread_condattr_setclock()</i>
23	<u>POSIX_THREADS and POSIX_BARRIERS</u>
24	<i>pthread_barrier_destroy(), pthread_barrier_init(), pthread_barrier_wait(),</i>
25	<i>pthread_barrierattr_destroy(), pthread_barrierattr_init(),</i>
26	<u>POSIX_THREADS, POSIX_BARRIERS and POSIX_THREAD_PROCESS_SHARED</u>
27	<i>pthread_barrierattr_getpshared(), pthread_barrierattr_setpshared()</i>
28	<u>POSIX_THREADS and POSIX_SPIN_LOCKS</u>
29	<i>pthread_spin_destroy(), pthread_spin_init(), pthread_spin_lock(),</i>
30	<i>pthread_spin_trylock(), pthread_spin_unlock()</i>
31	
32	<u>POSIX_THREADS and POSIX_THREAD_ATTR_STACKADDR</u>
33	<i>pthread_attr_getstackaddr(), pthread_attr_setstackaddr()</i>
34	<u>POSIX_THREADS, POSIX_THREAD_ATTR_STACKADDR and</u>
35	<u>POSIX_THREAD_ATTR_STACKSIZE</u>
36	<i>pthread_attr_getstack(), pthread_attr_setstack()</i>
37	<u>POSIX_THREADS and POSIX_THREAD_ATTR_STACKSIZE</u>
38	<i>pthread_attr_getstacksize(), pthread_attr_setstacksize()^a</i>
39	
40	<u>POSIX_THREADS and POSIX_THREAD_CPU_TIME</u>
41	<i>pthread_getcpuclockid()</i>
42	<u>POSIX_THREADS and either POSIX_THREAD_PRIO_INHERIT or</u>
43	<u>POSIX_THREAD_PRIO_PROTECT</u>
44	<i>pthread_mutexattr_getprotocol(), pthread_mutexattr_setprotocol()</i>
45	
46	
47	<i>This table row continued on next page...</i>
48	
49	

**Table B-1: Functions under each POSIX.1
System Interface Option (Continued)**

...table row continued from previous page

POSIX_THREAD_PRIO_PROTECT and POSIX_THREADS

*pthread_mutex_getprioceiling(), pthread_mutex_setprioceiling(),
pthread_mutexattr_getprioceiling(), pthread_mutexattr_setprioceiling()*

POSIX_THREADS and POSIX_THREAD_PRIORITY_SCHEDULING

*pthread_attr_getinheritsched(), pthread_attr_getschedpolicy(),
pthread_attr_getscope(), pthread_attr_setinheritsched(),
pthread_attr_setschedpolicy(), pthread_attr_setscope(),
pthread_getschedparam(), pthread_setschedparam(), pthread_setschedprio()*

POSIX_THREADS and POSIX_THREAD_PROCESS_SHARED

*pthread_condattr_getpshared(), pthread_condattr_setpshared(),
pthread_mutexattr_getpshared(), pthread_mutexattr_setpshared(),*

**POSIX_THREADS, POSIX_THREAD_PROCESS_SHARED and
POSIX_READER_WRITER_LOCKS**

pthread_rwlockattr_getpshared(), pthread_rwlockattr_setpshared()

POSIX_THREADS and POSIX_TIMEOUTS

pthread_mutex_timedlock()

POSIX_THREADS, POSIX_TIMEOUTS and POSIX_READER_WRITER_LOCKS

pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock()

POSIX_THREADS and POSIX_READER_WRITER_LOCKS

*pthread_rwlock_destroy(), pthread_rwlock_init(), pthread_rwlock_rdlock(),
pthread_rwlock_tryrdlock(), pthread_rwlock_trywrlock(),
pthread_rwlock_unlock(), pthread_rwlock_wrlock(),
pthread_rwlockattr_destroy(), pthread_rwlockattr_init()*

POSIX_THREADS or POSIX_PRIORITY_SCHEDULING

sched_yield()

POSIX_TIMEOUTS and POSIX_MESSAGE_PASSING

mq_timedreceive(), mq_timedsend()

POSIX_TIMEOUTS, POSIX_THREADS, and POSIX_READER_WRITER_LOCKS

pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock()

POSIX_TIMEOUTS and POSIX_SEMAPHORES

sem_timedwait()

POSIX_TIMEOUTS and POSIX_THREADS

pthread_mutex_timedlock()

POSIX_TIMEOUTS and POSIX_TRACE

posix_trace_timedgetnext_event()

POSIX_TIMERS

*clock_getres(), clock_gettime(), clock_settime(), nanosleep(), timer_create(),
timer_delete(), timer_getoverrun(), timer_gettime(), timer_settime()*

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

<p><u>_POSIX_TRACE</u></p> <p><i>posix_trace_attr_destroy(), posix_trace_attr_getclockres(), posix_trace_attr_getcreatetime(), posix_trace_attr_getgenversion(), posix_trace_attr_getname(), posix_trace_attr_getstreamfullpolicy(), posix_trace_attr_getmaxdatasize(), posix_trace_attr_getmaxsystemeventsizes(), posix_trace_attr_getmaxusereventsizes(), posix_trace_attr_getstreamsize(), posix_trace_attr_init(), posix_trace_attr_setname(), posix_trace_attr_setstreamfullpolicy(), posix_trace_attr_setmaxdatasize(), posix_trace_attr_setstreamsize(), posix_trace_clear(), posix_trace_create(), posix_trace_event(), posix_trace_eventid_open(), posix_trace_eventid_equal(), posix_trace_eventid_get_name(), posix_trace_eventtypelist_getnext_id(), posix_trace_eventtypelist_rewind(), posix_trace_get_attr(), posix_trace_get_status(), posix_trace_getnext_event(), posix_trace_shutdown(), posix_trace_start(), posix_trace_stop(), posix_trace_trygetnext_event()</i></p> <p><u>_POSIX_TRACE and _POSIX_TIMEOUTS</u></p> <p><i>posix_trace_timedgetnext_event()</i></p> <p><u>_POSIX_TRACE and _POSIX_TRACE_INHERIT</u></p> <p><i>posix_trace_attr_getinherited(), posix_trace_attr_setinherited()</i></p> <p><u>_POSIX_TRACE and _POSIX_TRACE_LOG</u></p> <p><i>posix_trace_attr_getlogfullpolicy(), posix_trace_attr_getlogsize(), posix_trace_attr_setlogfullpolicy(), posix_trace_attr_setlogsize(), posix_trace_close(), posix_trace_open(), posix_trace_rewind(), posix_trace_create_withlog(), posix_trace_flush()</i></p> <p><u>_POSIX_TRACE and _POSIX_TRACE_EVENT_FILTER</u></p> <p><i>posix_trace_eventset_add(), posix_trace_eventset_del(), posix_trace_eventset_empty(), posix_trace_eventset_fill(), posix_trace_eventset_ismember(), posix_trace_get_filter(), posix_trace_set_filter(), posix_trace_trid_eventid_open()</i></p>
<p><u>_POSIX_TRACE_EVENT_FILTER and _POSIX_TRACE</u></p> <p><i>posix_trace_eventset_add(), posix_trace_eventset_del(), posix_trace_eventset_empty(), posix_trace_eventset_fill(), posix_trace_eventset_ismember(), posix_trace_get_filter(), posix_trace_set_filter(), posix_trace_trid_eventid_open()</i></p>
<p><u>_POSIX_TRACE_INHERIT and _POSIX_TRACE</u></p> <p><i>posix_trace_attr_getinherited(), posix_trace_attr_setinherited()</i></p>
<p><u>_POSIX_TRACE_LOG and _POSIX_TRACE</u></p> <p><i>posix_trace_attr_getlogfullpolicy(), posix_trace_attr_getlogsize(), posix_trace_attr_setlogfullpolicy(), posix_trace_attr_setlogsize(), posix_trace_close(), posix_trace_open(), posix_trace_rewind(), posix_trace_create_withlog(), posix_trace_flush()</i></p>
<p><u>_POSIX_TYPED_MEMORY_OBJECTS</u></p> <p><i>posix_mem_offset(), posix_typed_mem_get_info(), posix_typed_mem_open()</i></p>
<p><u>_POSIX_VDISABLE</u></p> <p>No functions under this option</p>

**Table B-1: Functions under each POSIX.1
System Interface Option (Continued)**

1	
2	
3	
4	<u>_XOPEN_CRYPT</u>
5	<i>crypt(), encrypt(), setkey()</i>
6	<u>_XOPEN_ENH_I18N</u>
7	No functions under this option
8	
9	<u>_XOPEN_LEGACY</u>
10	<i>bcmp(), bcopy(), bzero(), ecvt(), fcvt(), ftime(), gcvt(), getwd(), index(), mktemp(),</i>
11	<i>rindex(), utimes(), wcs wcs()</i>
12	<u>_XOPEN_REALTIME</u>
13	This Option Group consists of the set of the following options from within POSIX.1:
14	_POSIX_ASYNCHRONOUS_IO
15	_POSIX_FSYNC
16	_POSIX_MAPPED_FILES
17	_POSIX_MEMLOCK
18	_POSIX_MEMLOCK_RANGE
19	_POSIX_MEMORY_PROTECTION
20	_POSIX_MESSAGE_PASSING
21	_POSIX_PRIORITIZED_IO
22	_POSIX_PRIORITY_SCHEDULING
23	_POSIX_REALTIME_SIGNALS
24	_POSIX_SEMAPHORES
25	_POSIX_SHARED_MEMORY_OBJECTS
26	_POSIX_SYNCHRONIZED_IO
27	_POSIX_TIMERS
28	
29	<u>_XOPEN_REALTIME_THREADS</u>
30	This Option Group consists of the set of the following options from within POSIX.1:
31	_POSIX_THREAD_PRIO_INHERIT
32	_POSIX_THREAD_PRIO_PROTECT
33	_POSIX_THREAD_PRIORITY_SCHEDULING
34	
35	<u>_XOPEN_SHM</u>
36	This option is included in the XSI_IPC unit of functionality
37	<u>_XOPEN_STREAMS</u>
38	<i>fattach(), fdetach(), getmsg(), getpmsg(), ioctl(), isastream(), putmsg(),</i>
39	<i>putpmsg(),</i>
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

_XOPEN_UNIX
This Option Group consists of the functions included in the following units of functionality:
XSI_C_LANG_SUPPORT
XSI_DBM
XSI_DEVICE_IO
XSI_DEVICE_SPECIFIC
XSI_DYNAMIC_LINKING
XSI_FD_MGMT
XSI_FILE_SYSTEM
XSI_I18N
XSI_IPC
XSI_JOB_CONTROL
XSI_JUMP
XSI_MATH
XSI_MULTI_PROCESS
XSI_SIGNALS
XSI_SINGLE_PROCESS
XSI_SYSTEM_DATABASE
XSI_SYSTEM_LOGGING
XSI_THREAD_MUTEX_EXT
XSI_THREADS_EXT
XSI_TIMERS
XSI_USER_GROUPS
XSI_WIDE_CHAR

- a. The *pthread_attr_getstacksize()* and *pthread_attr_setstacksize()* functions are wrongly listed under the `_POSIX_THREAD_STACK_ADDRESS` option in POSIX.1, but should be under the `_POSIX_THREAD_STACK_SIZE` option.

The following table shows the utilities included under each of the options specified in the Shell and Utilities volume of POSIX.1:

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

_POSIX_SHELL
sh
_POSIX_C_BIND
No utilities under this option
_POSIX2_C_DEV
c99, lex, yacc
_POSIX2_CHAR_TERM
No utilities under this option

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

_POSIX2_FORT_DEV fort77
_POSIX2_FORT_RUN asa
_POSIX2_LOCALEDEF No utilities under this option
_POSIX2_PBS qalter, qdel, qhold, qmove, qmsg, qrerun, qrls, qselect, qsig, qstat, qsub
_POSIX2_PBS_ACCOUNTING No utilities under this option.
_POSIX2_PBS_CHECKPOINT No utilities under this option.
_POSIX2_PBS_LOCATE No utilities under this option.
_POSIX2_PBS_MESSAGE No utilities under this option.
_POSIX2_PBS_TRACK No utilities under this option.
_POSIX2_SW_DEV ar, make, strip
_POSIX2_SW_DEV and _POSIX2_UPE nm
_POSIX2_UPE alias, at, batch, bg, command, crontab, csplit, ctags, df, du, ex, expand, fc, fg, file, jobs, mesg, more, newgrp, nice, patch, ps, renice, split, strings, tabs, talk, time, tput, unalias, unexpand, uudecode, uuencode, vi, who, write
_POSIX2_UPE and _POSIX2_SW_DEV nm

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

B.2 POSIX.5c Options

The following table shows the subprograms included under each of the options specified in POSIX.5c:

Table B-3: Packages and Subprograms under each POSIX.5c Option

Package	Subprogram
Asynchronous I/O POSIX_Asynchonus_IO	All except the two subprograms below
Asynchronous I/O and Synchronized I/O POSIX_Asynchonus_IO	Synchronize_File Synchronize_Data
Change Owner Restriction	None
File Synchronization POSIX_IO	Synchronize_File
Filename Truncation	None
Memory Mapped Files or Shared Memory Objects POSIX_IO	Change_Permissions Truncate_File
POSIX_Memory_Mapping	Map_Memory ^a Unmap_Memory
Memory Mapped Files and Synchronized I/O POSIX_Memory_Mapping	Synchronize_Memory
Memory Locking POSIX_Memory_Locking	All
Memory Protection POSIX_Memory_Mapping	Change_Protection
Memory Range Locking POSIX_Memory_Range_Locking	All
Message Queues POSIX_Message_Queues	All

Table B-3: Packages and Subprograms under each POSIX.5c Option (Continued)

Package	Subprogram
Mutexes	
POSIX_Mutexes	All except the subprograms below
POSIX_Condition_Variables	All except the subprograms below
Mutexes and Process Shared	
POSIX_Mutexes	Get_Process_Shared Set_Process_Shared
POSIX_Condition_Variables	Get_Process_Shared Set_Process_Shared
Mutexes and MutexPriority Ceiling	
POSIX_Mutexes	Set_Ceiling_Priority ^a Get_Ceiling_Priority ^a
Mutexes and either Mutex Priority Inheritance or MutexPriority Ceiling	
POSIX_Mutexes	Set_Locking_Policy Get_Locking_Policy
Mutex Priority Ceiling and Mutexes	
POSIX_Mutexes	Set_Ceiling_Priority ^a Get_Ceiling_Priority ^a Set_Locking_Policy Get_Locking_Policy
Mutex Priority Inheritance and Mutexes	
POSIX_Mutexes	Set_Locking_Policy Get_Locking_Policy
Network Management and Sockets Detailed Network Interface	
POSIX_Sockets	Set_Flags Get_Flags Set_Family Get_Family Set_Socket_Type Get_Socket_Type Set_Protocol_Number Get_Protocol_Number Get_Canonical_Name Get_Socket_Address_Info Get_Socket_Address_Info For_Every_Item
Poll	
POSIX_Event_Management	Get_File Set_File Get_Events Set_Events Get_Returned_Events Set_Returned_Events Poll
Prioritized I/O	
	None

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Table B-3: Packages and Subprograms under each POSIX.5c Option (Continued)

Package	Subprogram
Priority Process Scheduling	
POSIX_Process_Scheduling	All
Process Shared and Mutexes	
POSIX_Mutexes	Get_Process_Shared Set_Process_Shared
POSIX_Condition_Variables	Get_Process_Shared Set_Process_Shared
Realtime Signals	
POSIX_Signals	Enable_Queueing Disable_Queueing Await_Signal ^b Await_Signal_Or_Timeout ^b Queue_Signal
Saved IDs Support	
	None
Select	
POSIX_Event_Management	Add Remove In_Set Select_File ^a
Semaphores	
POSIX_Semaphores	All
Shared Memory Objects	
POSIX_Shared_Memory_Objects	All
POSIX_Generic_Shared_Memory	All
Shared Memory Objects and Memory Range Locking	
POSIX_Generic_Shared_Memory	Lock_Shared_Memory Unlock_Shared_Memory
Shared Memory Objects or Memory Mapped Files	
POSIX_IO	Truncate_File

**Table B-3: Packages and Subprograms under
each POSIX.5c Option (Continued)**

Package	Subprogram
Sockets Detailed Network Interface	
POSIX_Sockets	All except the subprograms below
Sockets Detailed Network Interface and Network Management	
POSIX_Sockets	Set_Flags Get_Flags Set_Family Get_Family Set_Socket_Type Get_Socket_Type Set_Protocol_Number Get_Protocol_Number Get_Canonical_Name Get_Socket_Address_Info Get_Socket_Address_Info For_Every_Item
Synchronized I/O	
POSIX_IO	Synchronize_Data
Synchronized I/O and Memory Mapped Files	
POSIX_Memory_Mapping	Synchronize_Memory
Timers	
POSIX_Timers	All
XTI Detailed Network Interface	
POSIX_XTI	All

- a. All versions
- b. Return type Signal_Info

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Annex C: Bibliography (Informative)

This Annex contains lists of related open systems standards and suggested reading on historical implementations and application programming.

C.1 Related Open Systems Standards

- {B1} ISO 8859-1: 1987, *Information Processing—8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1.*
- {B2} ISO/IEC 10646:..., *Information processing—Multiple octet coded character set.*
- {B3} IEEE Std 100-1988, *IEEE Standard Dictionary of Electrical and Electronics Terms.*
- {B4} ISO/IEC TR 10000-2:1998 *Information technology -- Framework and taxonomy of International Standardized Profiles -- Part 2: Principles and Taxonomy for OSI Profiles.*

C.2 Other Documents

- {B5} *The Single UNIX Specification: The Authorized Guide to Version 3.* The Open Group, March 2002. UK ISBN: 1-85912-277-9. US ISBN 1-931624-13-5.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Alphabetic Topical Index

Symbols

- <limits.h> header... 36
- <unistd.h> header... 36, 39, 40, 47, 57, 58, 63, 64, 75, 76, 82, 93, 94, 100, 101
- Exit()* function... 7
- exit()* function... 7
- longjmp()* function... 9
- _POSIX_ADVISORY_INFO option... 18, 20, 49, 66, 84, 95
- _POSIX_AEP_REALTIME_format... 31
- _POSIX_AEP_REALTIME_option... 39, 57, 75, 93
- _POSIX_AEP_REALTIME_CONTROLLER option... 57
- _POSIX_AEP_REALTIME_DEDICATED option... 75
- _POSIX_AEP_REALTIME_LANG_Ada95 option... 40, 47, 58, 64, 76, 82, 94, 101
- _POSIX_AEP_REALTIME_LANG_C99 option... 40, 47, 58, 63, 76, 82, 94, 100
- _POSIX_AEP_REALTIME_MINIMAL option... 39
- _POSIX_AEP_REALTIME_MULTI option... 93
- _POSIX_ASYNCHRONOUS_IO option... 18, 20, 77, 95, 120
- _POSIX_BARRIERS option... 18, 20
- _POSIX_CHOWN_RESTRICTED option... 18, 20, 95
- _POSIX_CLOCK_SELECTION option... 18, 20, 41, 59, 77, 95
- _POSIX_CPUTIME option... 18, 20, 77, 95
- _POSIX_FSYNC option... 18, 20, 41, 59, 77, 95, 120
- _POSIX_IPV6 option... 18, 20
- _POSIX_JOB_CONTROL option... 10
- _POSIX_JOB_CONTROL option... 18, 95
- _POSIX_MAPPED_FILES option... 18, 20, 49, 59, 77, 95, 120
- _POSIX_MEMLOCK option... 18, 20, 41, 48, 59, 77, 95, 120
- _POSIX_MEMLOCK_RANGE option... 18, 20, 41, 59, 77, 95, 120
- _POSIX_MEMORY_PROTECTION option... 18, 20, 77, 95, 120
- _POSIX_MESSAGE_PASSING option... 18, 20, 59, 77, 95, 120
- _POSIX_MONOTONIC_CLOCK option... 19, 20, 41, 59, 77, 95
- _POSIX_NGROUPS_MAX limit... 96
- _POSIX_NO_TRUNC option... 19, 20, 22, 41, 49, 59, 77, 95
- _POSIX_PRIORITIZED_IO option... 19, 20, 77, 95, 120
- _POSIX_PRIORITY_SCHEDULING option... 19, 20, 77, 86, 95, 104, 120
- _POSIX_RAW_SOCKETS option... 19, 20, 77, 95
- _POSIX_READER_WRITER_LOCKS option... 10
- _POSIX_READER_WRITER_LOCKS option... 19, 20
- _POSIX_REALTIME_SIGNALS option... 19, 20, 41, 59, 77, 95, 120
- _POSIX_REGEX option... 10
- _POSIX_REGEX option... 19, 21, 95
- _POSIX_RTSIG_MAX limit... 41, 59, 77, 96
- _POSIX_SAVED_IDS option... 19, 21, 95
- _POSIX_SEMAPHORES option... 19, 21, 41, 59, 77, 95, 120
- _POSIX_SHARED_MEMORY_OBJECTS option... 19, 21, 41, 59, 77, 95, 120
- _POSIX_SHELL option... 19, 21, 96
- _POSIX_SPAWN option... 19, 21, 77, 96
- _POSIX_SPIN_LOCKS option... 19, 21
- _POSIX_SPARADIC_SERVER option... 19, 21, 77, 96
- _POSIX_SYNCHRONIZED_IO option... 19, 21, 41, 59, 77, 96, 120
- _POSIX_THREAD_ATTR_STACKADDR option... 19, 21, 41, 59, 77, 96
- _POSIX_THREAD_ATTR_STACKSIZE option... 19, 21, 41, 59, 77, 96
- _POSIX_THREAD_CPUTIME option... 19, 21, 41, 59, 77, 96
- _POSIX_THREAD_PRIO_INHERIT option... 19, 21, 41, 59, 77, 96, 120
- _POSIX_THREAD_PRIO_PROTECT option...

1	19, 21, 41, 59, 77, 96, 120	<code>_XOPEN_ENH_I18N</code> option... 20, 21
2	<code>_POSIX_THREAD_PRIORITY_SCHEDULING</code>	<code>_XOPEN_LEGACY</code> option... 20, 22, 55, 72, 90,
3	option... 19, 21, 41, 59, 77, 96, 120	109
4	<code>_POSIX_THREAD_PROCESS_SHARED</code>	<code>_XOPEN_REALTIME</code> option... 20, 22
5	option... 10	<code>_XOPEN_REALTIME_THREADS</code> option... 20,
6	<code>_POSIX_THREAD_PROCESS_SHARED</code>	22
7	option... 19, 21, 77, 96	<code>_XOPEN_SHM</code> option... 20, 22
8	<code>_POSIX_THREAD_SAFE_FUNCTIONS</code>	<code>_XOPEN_STREAMS</code> option... 20, 22, 55, 72, 90,
9	option... 19, 21, 96	109
10	<code>_POSIX_THREAD_SPORADIC_SERVER</code>	<code>_XOPEN_UNIX</code> option... 20, 22
11	option... 19, 21, 41, 59, 77, 96	
12	<code>_POSIX_THREAD_STACK_ADDRESS</code>	
13	option... 121	
14	<code>_POSIX_THREAD_STACK_SIZE</code> option...	
15	121	
16	<code>_POSIX_THREADS</code> option... 10, 19, 21, 53, 70,	
17	88, 107	
18	<code>_POSIX_TIMEOUTS</code> option... 10	
19	<code>_POSIX_TIMEOUTS</code> option... 19, 21, 41, 59,	
20	77, 96	
21	<code>_POSIX_TIMER_MAX</code> limit... 41, 59, 77, 96	
22	<code>_POSIX_TIMERS</code> option... 19, 21, 41, 59, 77,	
23	96, 120	
24	<code>_POSIX_TRACE</code> option... 19, 21, 59, 77, 96	
25	<code>_POSIX_TRACE_EVENT_FILTER</code> option...	
26	19, 21, 59, 77, 96	
27	<code>_POSIX_TRACE_INHERIT</code> option... 19, 21	
28	<code>_POSIX_TRACE_LOG</code> option... 19, 21, 59, 77,	
29	96	
30	<code>_POSIX_TYPED_MEMORY_OBJECTS</code>	
31	option... 19, 21	
32	<code>_POSIX_VDISABLE</code> option... 19, 21, 22, 96	
33	<code>_POSIX_VERSION</code> option... 44, 62, 80	
34	<code>_POSIX2_C_BIND</code> option... 19, 21	
35	<code>_POSIX2_C_DEV</code> option... 19, 21	
36	<code>_POSIX2_CHAR_TERM</code> option... 19, 21	
37	<code>_POSIX2_FORT_DEV</code> option... 19, 21	
38	<code>_POSIX2_FORT_RUN</code> option... 19, 21	
39	<code>_POSIX2_LOCALEDEF</code> option... 19, 21	
40	<code>_POSIX2_PBS</code> option... 19, 21	
41	<code>_POSIX2_PBS_ACCOUNTING</code> option... 19, 21	
42	<code>_POSIX2_PBS_CHECKPOINT</code> option... 20, 21	
43	<code>_POSIX2_PBS_LOCATE</code> option... 20, 21	
44	<code>_POSIX2_PBS_MESSAGE</code> option... 20, 21	
45	<code>_POSIX2_PBS_TRACK</code> option... 20, 21	
46	<code>_POSIX2_SW_DEV</code> option... 20, 21	
47	<code>_POSIX2_UPE</code> option... 20, 21	
48	<code>_setjmp()</code> function... 9	
49	<code>_tolower()</code> function... 9	
	<code>_toupper()</code> function... 9	
	<code>_XOPEN_CRYPT</code> option... 20, 21, 55, 72, 90,	
	109	
		A
		<code>a64l()</code> function... 9
		Abbreviations... 32
		<code>abort()</code> function... 8
		<code>abs()</code> function... 6
		<code>accept()</code> function... 7
		<code>access()</code> function... 7
		Accessibility subprogram... 13
		<code>acos()</code> function... 5
		<code>acosf()</code> function... 5
		<code>acosh()</code> function... 5
		<code>acoshf()</code> function... 5
		<code>acoshl()</code> function... 5
		<code>acosl()</code> function... 5
		Ada Language Option... 42, 45, 60, 62, 78, 80,
		97, 99, 100
		Ada Language option... 72
		Ada language option... 91, 109
		Ada_Streams package... 10
		Ada_Task_Identification package... 10
		Ada95 RM... 32
		Ada-Language option... 18, 36
		Add subprogram... 11, 125
		Add_All_Signals subprogram... 15
		Add_Signal subprogram... 15
		Advisory Information option... 102
		AEP... 26, 33
		<code>aio_cancel()</code> function... 113
		<code>aio_error()</code> function... 113
		<code>aio_fsync()</code> function... 113
		<code>aio_read()</code> function... 113
		<code>aio_return()</code> function... 113
		<code>aio_suspend()</code> function... 113
		<code>aio_write()</code> function... 113
		<code>alarm()</code> function... 8, 44, 52, 62, 69, 88, 106
		alias utility... 122
		Application Conformance... 36
		Application Environment Profile... 26
		application environment profile... 26, 27, 33

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

- 1 Application Platform... 26
 2 ar utility... 122
 3 Argument_List subprogram... 16
 4 asa utility... 122
 5 *asctime()* function... 6
 6 *asctime_r()* function... 6, 116
 7 *asin()* function... 5
 8 *asinf()* function... 5
 9 *asinh()* function... 5
 10 *asinhf()* function... 5
 11 *asinhfhl()* function... 5
 12 *asinl()* function... 5
 13 *assert()* function... 7
 14 Asynchronous I/O option... 20, 79, 97, 123
 15 at utility... 122
 16 *atan()* function... 5
 17 *atan2()* function... 5
 18 *atan2f()* function... 5
 19 *atan2l()* function... 5
 20 *atanf()* function... 5
 21 *atanh()* function... 5
 22 *atanhf()* function... 5
 23 *atanhhl()* function... 5
 24 *atanl()* function... 5
 25 *atexit()* function... 7
 26 *atof()* function... 6
 27 *atoi()* function... 6
 28 *atol()* function... 6
 29 *atoll()* function... 6
 30 Await_Signal subprogram... 15, 125
 31 Await_Signal_Or_Timeout
 32 subprogram... 15, 125
 33
 34 **B**
 35 Base Standard... 26
 36 base standard... 26
 37 *basename()* function... 9
 38 batch utility... 122
 39 *bcmp()* function... 120
 40 *bcopy()* function... 120
 41 bg utility... 122
 42 Bibliography... 127
 43 *bind()* function... 7
 44 Bits_Per_Character_Of subprogram...
 45 11
 46 Block_Signals subprogram... 15
 47 Blocked_Signals subprogram... 15
 48 Blocking_Behavior constant... 22
 49 Boolean type... 39, 40, 57, 58, 75, 76, 93, 94,
 111

bsd_signal() function... 9
bsearch() function... 6
btowc() function... 6
bzero() function... 120

C
 C Language Option... 40, 43, 61, 76, 80, 94, 99
 C Language option... 72
 C language Option... 58
 C language option... 90, 109
 C99 Standard... 32
 c99 utility... 121
cabs() function... 5
cabsf() function... 5
cabsl() function... 5
cacos() function... 5
cacosf() function... 5
cacosh() function... 5
cacoshf() function... 5
cacoshhl() function... 5
cacosl() function... 5
calloc() function... 6
carg() function... 5
cargf() function... 5
cargl() function... 5
casin() function... 5
casinf() function... 5
casinh() function... 5
casinhf() function... 5
casinhhl() function... 5
casinl() function... 5
catan() function... 5
catanf() function... 5
catanh() function... 5
catanhf() function... 5
catanhhl() function... 5
catanl() function... 5
catclose() function... 9
catgets() function... 9
catopen() function... 9
cbirt() function... 5
cbirtf() function... 5
cbirtl() function... 5
ccos() function... 5
ccosf() function... 5
ccosh() function... 5
ccoshf() function... 5
ccoshhl() function... 5
ccosl() function... 5
ceil() function... 5

1	<i>ceilf()</i> function... 5	conforming application
2	<i>ceill()</i> function... 5	strictly... 35
3	<i>cexp()</i> function... 5	conforming implementation... 35
4	<i>cexpf()</i> function... 5	<i>confstr()</i> function... 8
5	<i>cexpl()</i> function... 5	<i>conj()</i> function... 5
6	<i>cfgetispeed()</i> function... 7	<i>conjf()</i> function... 5
7	<i>cfgetospeed()</i> function... 7	<i>conjl()</i> function... 5
8	<i>cfsetispeed()</i> function... 7	<i>connect()</i> function... 7
9	<i>cfsetospeed()</i> function... 7	constant
10	Change Owner Restriction option... 20, 97, 123	Blocking_Behavior... 22
11	Change_Owner_And_Group subprogram... 12	False... 22
12	Change_Permissions subprogram... 12, 123	False..False... 42, 60, 79, 97, 111
13	Change_Protection subprogram... 123	File_Structure... 45, 63, 81
14	Change_Working_Directory subprogram... 13	Group... 45
15	<i>chdir()</i> function... 7	Operation_Not_Implemented... 22, 98
16	CHILD_MAX limit... 96	Operation_Not_Supported... 43, 61, 80, 98
17	CHILD_MAX option... 44, 62	Other... 45
18	<i>chmod()</i> function... 7	Owner... 45
19	<i>chown()</i> function... 7	POSIX_Limits.Child_Processes_ Maxima'Last... 45, 62
20	<i>cimag()</i> function... 5	POSIX_Limits.Groups_Maxima'Fi rst... 22
21	<i>cimagf()</i> function... 5	POSIX_Limits.Groups_Maxima'Fi rst... 43, 61, 79
22	<i>cimagl()</i> function... 5	PTHREAD_SCOPE_PROCESS... 78, 96
23	C-Language option... 18, 36	PTHREAD_SCOPE_SYSTEM... 78, 86, 96, 105
24	C-language option... 20	Read_Write_Execute... 45
25	Clear_Environment subprogram... 16	SCHED_FIFO... 41, 51, 59, 68, 78, 86, 96, 104, 105
26	<i>clearerr()</i> function... 6	SCHED_RR... 41, 51, 59, 68, 78, 86, 96, 104, 105
27	Clock Selection option... 52, 69, 88, 106	True..True... 39, 40, 42, 47, 57, 58, 60, 63, 64, 75, 76, 79, 82, 93, 94, 97, 100, 101, 111
28	<i>clock()</i> function... 7	constant-width format... 31
29	<i>clock_getcpuclokid()</i> function... 113	Conventions... 31
30	<i>clock_getres()</i> function... 118	Copy_Environment subprogram... 16
31	<i>clock_gettime()</i> function... 44, 62, 118	Copy_From_Current_Environment subprogram... 16
32	<i>clock_nanosleep()</i> function... 113	Copy_To_Current_Environment subprogram... 16
33	<i>clock_settime()</i> function... 118	<i>copysign()</i> function... 5
34	<i>clog()</i> function... 5	<i>copysignf()</i> function... 5
35	<i>clogf()</i> function... 5	<i>copysignl()</i> function... 5
36	<i>clogl()</i> function... 5	<i>cos()</i> function... 5
37	Close subprogram... 11	<i>cosf()</i> function... 5
38	<i>close()</i> function... xv, 6	<i>cosh()</i> function... 5
39	<i>closedir()</i> function... 7	<i>coshf()</i> function... 5
40	<i>closelog()</i> function... 9	<i>coshl()</i> function... 5
41	command utility... 122	
42	Component Profile... 26	
43	component profile... 26	
44	Conformance... 35	
45	Conformance Document... 27	
46	conformance document... 27, 28, 35	
47	Conformant Application... 37	
48	Conformant Application Using Extensions... 37	
49		

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1	<i>cosl()</i> function... 5	<i>dbm_store()</i> function... 9
2	<i>cpow()</i> function... 5	Dedicated Realtime System Profile... 3, 75
3	<i>cpowf()</i> function... 5	Define_Bits_Per_Character
4	<i>cpowl()</i> function... 5	subprogram... 11
5	<i>cproj()</i> function... 5	Define_Input_Baud_Rate subprogram...
6	<i>cprojf()</i> function... 5	11
7	<i>cprojl()</i> function... 5	Define_Input_Time subprogram... 11
8	<i>creal()</i> function... 5	Define_Minimum_Input_Count
9	<i>crealf()</i> function... 5	subprogram... 11
10	<i>creall()</i> function... 5	Define_Output_Baud_Rate
11	<i>creat()</i> function... xvii, 7	subprogram... 11
12	Create subprogram... 46	Define_Special_Control_Character
13	Create_Directory subprogram... 13	subprogram... 11
14	Create_FIFO subprogram... 12	Define_Terminal_Modes subprogram...
15	Create_Pipe subprogram... 14	11
16	Create_Process_Group subprogram... 13	Definitions... 26
17	Create_Session subprogram... 16	Delete subprogram... 46
18	crontab utility... 122	Delete_All_Signals subprogram... 15
19	<i>crypt()</i> function... 120	Delete_Environment_Variable
20	<i>csin()</i> function... 5	subprogram... 16
21	<i>csinf()</i> function... 5	Delete_Signal subprogram... 15
22	<i>csinh()</i> function... 5	Development Environment... 16
23	<i>csinhf()</i> function... 5	Development Platform... 27
24	<i>csinhl()</i> function... 5	development platform... 27
25	<i>csinl()</i> function... 5	df utility... 122
26	csplit utility... 122	<i>difftime()</i> function... 6
27	<i>csqrt()</i> function... 5	<i>dirname()</i> function... 9
28	<i>csqrtf()</i> function... 5	Disable_Control_Character
29	<i>csqrtl()</i> function... 5	subprogram... 11
30	ctags utility... 122	Disable_Queueing subprogram... 125
31	<i>ctan()</i> function... 5	Discard_Data subprogram... 11
32	<i>ctanf()</i> function... 5	<i>div()</i> function... 6
33	<i>ctanh()</i> function... 5	<i>dlclose()</i> function... 9
34	<i>ctanhf()</i> function... 5	<i>dlderror()</i> function... 9
35	<i>ctanhl()</i> function... 5	<i>dlopen()</i> function... 9
36	<i>ctanl()</i> function... 5	<i>dlsym()</i> function... 9
37	<i>ctermid()</i> function... 7	document
38	<i>ctime()</i> function... 6	conformance... 27
39	<i>ctime_r()</i> function... 6, 116	Documentation... 35
40		documentation
41	D	system... 28
42	<i>daylight()</i> function... 9	Drain subprogram... 11
43	<i>dbm_clearerr()</i> function... 9	<i>drand48()</i> function... 9
44	<i>dbm_close()</i> function... 9	du utility... 122
45	<i>dbm_delete()</i> function... 9	<i>dup()</i> function... 7
46	<i>dbm_error()</i> function... 9	<i>dup2()</i> function... 7
47	<i>dbm_fetch()</i> function... 9	Duplicate subprogram... 12
48	<i>dbm_firstkey()</i> function... 9	Duplicate_And_Close subprogram... 12
49	<i>dbm_nextkey()</i> function... 9	
	<i>dbm_open()</i> function... 9	

- 1 **E**
- 2
- 3 *ecvt()* function... 120
- 4 Embedded Computer System... 27
- 5 *Enable_Queueing* subprogram... 125
- 6 *encrypt()* function... 120
- 7 *endgrent()* function... 10
- 8 *endhostent()* function... 7
- 9 *endnetent()* function... 7
- 10 *endprotoent()* function... 7
- 11 *endpwent()* function... 9
- 12 *endservent()* function... 7
- 13 *endutxent()* function... 10
- 14 environment
- 15 open system... 28
- 16 environment, open system... 33
- 17 *Environment_Value_Of* subprogram... 16
- 18 *erand48()* function... 9
- 19 *erf()* function... 5
- 20 *erfc()* function... 5
- 21 *erfcf()* function... 5
- 22 *erfcl()* function... 5
- 23 *erff()* function... 5
- 24 *erfl()* function... 5
- 25 **ERRNO** format... 31
- 26 *ex* utility... 122
- 27 exception
- 28 *POSIX_Error*... 22, 43, 61, 80, 98
- 29 *Use_Error*... 45, 63, 81
- 30 *execl()* function... 7
- 31 *execle()* function... 7
- 32 *execlp()* function... 7
- 33 *execv()* function... 7
- 34 *execve()* function... 7
- 35 *execvp()* function... 7
- 36 *Existence* subprogram... 13
- 37 *exit()* function... 7
- 38 *exp()* function... 5
- 39 *exp2()* function... 5
- 40 *exp2f()* function... 5
- 41 *exp2l()* function... 5
- 42 *expand* utility... 122
- 43 *expf()* function... 5
- 44 *expl()* function... 5
- 45 *expm1()* function... 5
- 46 *expm1f()* function... 5
- 47 *expm1l()* function... 5
- 48 **F**
- 49 *fabs()* function... 5
- fabsf()* function... 5
- fabsl()* function... 5
- False* constant... 22
- False . . False* constant... 42, 60, 79, 97, 111
- fattach()* function... 120
- fc* utility... 122
- fchdir()* function... 9
- fchmod()* function... 7
- fchown()* function... 7
- fclose()* function... 6, 44
- fcntl()* function... 7
- fcvt()* function... 120
- FD_CLR()* function... 7
- FD_ISSET()* function... 7
- FD_SET()* function... 7
- FD_ZERO()* function... 7
- fdatasync()* function... 116
- fdetach()* function... 120
- fdim()* function... 5
- fdimf()* function... 5
- fdiml()* function... 5
- fdopen()* function... 6
- feclearexcept()* function... 6
- fegetenv()* function... 6
- fegetexceptflag()* function... 6
- fegetround()* function... 6
- feholdexcept()* function... 6
- feof()* function... 6
- feraiseexcept()* function... 6
- ferror()* function... 6
- fesetenv()* function... 6
- fesetexceptflag()* function... 6
- fesetround()* function... 6
- fetestexcept()* function... 6
- feupdateenv()* function... 6
- fflush()* function... 6, 44
- ffs()* function... 9
- fg* utility... 122
- fgetc()* function... 6, 44
- fgetpos()* function... 7
- fgets()* function... 6, 44
- fgetwc()* function... 9
- fgetws()* function... 9
- FILE** * type... 49, 66, 84, 102
- File Locking... 84
- File Locking option... 49, 66, 84, 102
- File Synchronization option... 49, 66, 84
- File Synchronization option... 20, 42, 60, 79, 97, 123
- file utility... 122
- File_Position* subprogram... 12
- File_Size* subprogram... 12

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	File_Structure constant... 45, 63, 81	<i>frexp()</i> function... 5
2	Filename Truncation option... 42, 60, 79, 97	<i>frexpf()</i> function... 5
3	Filename Truncation option... 20, 22, 42, 60,	<i>frexpl()</i> function... 5
4	79, 97, 123	<i>fscanf()</i> function... 6, 44
5	Filename_Of subprogram... 13	<i>fsck</i> utility... xvii
6	<i>fileno()</i> function... 6	<i>fseek()</i> function... 7
7	<i>flockfile()</i> function... 7, 116	<i>fseeko()</i> function... 7
8	<i>floor()</i> function... 5	<i>fsetpos()</i> function... 7
9	<i>floorf()</i> function... 5	<i>fstat()</i> function... 7
10	<i>floorl()</i> function... 5	<i>fstatvfs()</i> function... 9
11	Flow subprogram... 11	<i>fsync()</i> function... 114
12	<i>fma()</i> function... 5	<i>ftell()</i> function... 7
13	<i>maf()</i> function... 5	<i>ftello()</i> function... 7
14	<i>mal()</i> function... 5	<i>ftime()</i> function... 120
15	<i>fmax()</i> function... 5	<i>ftok()</i> function... 9
16	<i>fmaxf()</i> function... 5	<i>ftruncate()</i> function... 7
17	<i>fmaxl()</i> function... 5	<i>ftrylockfile()</i> function... 7, 116
18	<i>fmin()</i> function... 5	<i>ftw()</i> function... 9
19	<i>fminf()</i> function... 5	function
20	<i>fminl()</i> function... 5	<i>_Exit()</i> ... 7
21	<i>fmod()</i> function... 5	<i>_exit()</i> ... 7
22	<i>fmodf()</i> function... 5	<i>_longjmp()</i> ... 9
23	<i>fmodl()</i> function... 5	<i>_setjmp()</i> ... 9
24	<i>fntmsg()</i> function... 9	<i>_tolower()</i> ... 9
25	<i>fnmatch()</i> function... 8	<i>_toupper()</i> ... 9
26	<i>fopen()</i> function... 6, 44	<i>a64l()</i> ... 9
27	For_Every_Current_Environment_Va	<i>abort()</i> ... 8
28	riable subprogram... 16	<i>abs()</i> ... 6
29	For_Every_Directory_Entry	<i>accept()</i> ... 7
30	subprogram... 13	<i>access()</i> ... 7
31	For_Every_Environment_Variable	<i>acos()</i> ... 5
32	subprogram... 16	<i>acosf()</i> ... 5
33	For_Every_File_In subprogram... 11	<i>acosh()</i> ... 5
34	For_Every_Item subprogram... 124, 126	<i>acoshf()</i> ... 5
35	<i>fork()</i> function... 7	<i>acoshl()</i> ... 5
36	format	<i>acosl()</i> ... 5
37	_POSIX_AEP_REALTIME_... 31	<i>aio_cancel()</i> ... 113
38	constant-width... 31	<i>aio_error()</i> ... 113
39	ERRNO... 31	<i>aio_fsync()</i> ... 113
40	<i>format</i> function family... 31	<i>aio_read()</i> ... 113
41	<i>fort77</i> utility... 122	<i>aio_return()</i> ... 113
42	<i>fpathconf()</i> function... 7	<i>aio_suspend()</i> ... 113
43	<i>fpclassify()</i> function... 5	<i>aio_write()</i> ... 113
44	<i>fprintf()</i> function... 6, 44	<i>alarm()</i> ... 8, 44, 52, 62, 69, 88, 106
45	<i>fputc()</i> function... 6, 44	<i>asctime()</i> ... 6
46	<i>fputs()</i> function... 6, 44	<i>asctime_r()</i> ... 6, 116
47	<i>fputwc()</i> function... 9	<i>asin()</i> ... 5
48	<i>fputws()</i> function... 9	<i>asinf()</i> ... 5
49	<i>fread()</i> function... 6, 44	<i>asinh()</i> ... 5
	<i>free()</i> function... 6	<i>asinhf()</i> ... 5
	<i>freeaddrinfo()</i> function... 7	<i>asinhhl()</i> ... 5
	<i>freopen()</i> function... 6, 44	<i>asinl()</i> ... 5

1	<i>assert()</i> ... 7	<i>cbrtf()</i> ... 5
2	<i>atan()</i> ... 5	<i>cbrtl()</i> ... 5
3	<i>atan2()</i> ... 5	<i>ccos()</i> ... 5
4	<i>atan2f()</i> ... 5	<i>ccosf()</i> ... 5
5	<i>atan2l()</i> ... 5	<i>ccosh()</i> ... 5
6	<i>atanf()</i> ... 5	<i>ccoshf()</i> ... 5
7	<i>atanh()</i> ... 5	<i>ccoshl()</i> ... 5
8	<i>atanhf()</i> ... 5	<i>ccosl()</i> ... 5
9	<i>atanhl()</i> ... 5	<i>ceil()</i> ... 5
10	<i>atanl()</i> ... 5	<i>ceilf()</i> ... 5
11	<i>atexit()</i> ... 7	<i>ceill()</i> ... 5
12	<i>atof()</i> ... 6	<i>cexp()</i> ... 5
13	<i>atoi()</i> ... 6	<i>cexpf()</i> ... 5
14	<i>atol()</i> ... 6	<i>cexpl()</i> ... 5
15	<i>atoll()</i> ... 6	<i>cfgetispeed()</i> ... 7
16	<i>basename()</i> ... 9	<i>cfgetospeed()</i> ... 7
17	<i>bcmp()</i> ... 120	<i>cfsetispeed()</i> ... 7
18	<i>bcopy()</i> ... 120	<i>cfsetospeed()</i> ... 7
19	<i>bind()</i> ... 7	<i>chdir()</i> ... 7
20	<i>bsd_signal()</i> ... 9	<i>chmod()</i> ... 7
21	<i>bsearch()</i> ... 6	<i>chown()</i> ... 7
22	<i>btowc()</i> ... 6	<i>cimag()</i> ... 5
23	<i>bzero()</i> ... 120	<i>cimagf()</i> ... 5
24	<i>cabs()</i> ... 5	<i>cimagl()</i> ... 5
25	<i>cabsf()</i> ... 5	<i>clearerr()</i> ... 6
26	<i>cabsl()</i> ... 5	<i>clock()</i> ... 7
27	<i>cacos()</i> ... 5	<i>clock_getcpuclokid()</i> ... 113
28	<i>cacosf()</i> ... 5	<i>clock_getres()</i> ... 118
29	<i>cacosh()</i> ... 5	<i>clock_gettime()</i> ... 44, 62, 118
30	<i>cacoshf()</i> ... 5	<i>clock_nanosleep()</i> ... 113
31	<i>cacoshl()</i> ... 5	<i>clock_settime()</i> ... 118
32	<i>cacosl()</i> ... 5	<i>clog()</i> ... 5
33	<i>calloc()</i> ... 6	<i>clogf()</i> ... 5
34	<i>carg()</i> ... 5	<i>clogl()</i> ... 5
35	<i>cargf()</i> ... 5	<i>close()</i> ... xv, 6
36	<i>cargl()</i> ... 5	<i>closedir()</i> ... 7
37	<i>casin()</i> ... 5	<i>closelog()</i> ... 9
38	<i>casinf()</i> ... 5	<i>confstr()</i> ... 8
39	<i>casinh()</i> ... 5	<i>conj()</i> ... 5
40	<i>casinhf()</i> ... 5	<i>conjf()</i> ... 5
41	<i>casinhl()</i> ... 5	<i>conjl()</i> ... 5
42	<i>casinl()</i> ... 5	<i>connect()</i> ... 7
43	<i>catan()</i> ... 5	<i>copysign()</i> ... 5
44	<i>catanf()</i> ... 5	<i>copysignf()</i> ... 5
45	<i>catanh()</i> ... 5	<i>copysignl()</i> ... 5
46	<i>catanhf()</i> ... 5	<i>cos()</i> ... 5
47	<i>catanhl()</i> ... 5	<i>cosf()</i> ... 5
48	<i>catanl()</i> ... 5	<i>cosh()</i> ... 5
49	<i>catclose()</i> ... 9	<i>coshf()</i> ... 5
	<i>catgets()</i> ... 9	<i>coshl()</i> ... 5
	<i>catopen()</i> ... 9	<i>cosl()</i> ... 5
	<i>cbrt()</i> ... 5	<i>cpow()</i> ... 5

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	<i>cpowf()</i> ... 5	<i>endnetent()</i> ... 7
2	<i>cpowl()</i> ... 5	<i>endprotoent()</i> ... 7
3	<i>cproj()</i> ... 5	<i>endpwent()</i> ... 9
4	<i>cprojf()</i> ... 5	<i>endservent()</i> ... 7
5	<i>cprojl()</i> ... 5	<i>endutxent()</i> ... 10
6	<i>creal()</i> ... 5	<i>erand48()</i> ... 9
7	<i>crealf()</i> ... 5	<i>erf()</i> ... 5
8	<i>creall()</i> ... 5	<i>erfc()</i> ... 5
9	<i>creat()</i> ... xvii, 7	<i>erfcf()</i> ... 5
10	<i>crypt()</i> ... 120	<i>erfcl()</i> ... 5
11	<i>csin()</i> ... 5	<i>erff()</i> ... 5
12	<i>csinf()</i> ... 5	<i>erfl()</i> ... 5
13	<i>csinh()</i> ... 5	<i>execl()</i> ... 7
14	<i>csinhf()</i> ... 5	<i>execle()</i> ... 7
15	<i>csinhl()</i> ... 5	<i>execlp()</i> ... 7
16	<i>csinl()</i> ... 5	<i>execv()</i> ... 7
17	<i>csqrt()</i> ... 5	<i>execve()</i> ... 7
18	<i>csqrtf()</i> ... 5	<i>execvp()</i> ... 7
19	<i>csqrtl()</i> ... 5	<i>exit()</i> ... 7
20	<i>ctan()</i> ... 5	<i>exp()</i> ... 5
21	<i>ctanf()</i> ... 5	<i>exp2()</i> ... 5
22	<i>ctanh()</i> ... 5	<i>exp2f()</i> ... 5
23	<i>ctanhf()</i> ... 5	<i>exp2l()</i> ... 5
24	<i>ctanhl()</i> ... 5	<i>expf()</i> ... 5
25	<i>ctanl()</i> ... 5	<i>expl()</i> ... 5
26	<i>ctermid()</i> ... 7	<i>expm1()</i> ... 5
27	<i>ctime()</i> ... 6	<i>expm1f()</i> ... 5
28	<i>ctime_r()</i> ... 6, 116	<i>expm1l()</i> ... 5
29	<i>daylight()</i> ... 9	<i>fabs()</i> ... 5
30	<i>dbm_clearerr()</i> ... 9	<i>fabsf()</i> ... 5
31	<i>dbm_close()</i> ... 9	<i>fabsl()</i> ... 5
32	<i>dbm_delete()</i> ... 9	<i>fattach()</i> ... 120
33	<i>dbm_error()</i> ... 9	<i>fchdir()</i> ... 9
34	<i>dbm_fetch()</i> ... 9	<i>fchmod()</i> ... 7
35	<i>dbm_firstkey()</i> ... 9	<i>fchown()</i> ... 7
36	<i>dbm_nextkey()</i> ... 9	<i>fclose()</i> ... 6, 44
37	<i>dbm_open()</i> ... 9	<i>fcntl()</i> ... 7
38	<i>dbm_store()</i> ... 9	<i>fcvt()</i> ... 120
39	<i>difftime()</i> ... 6	<i>FD_CLR()</i> ... 7
40	<i>dirname()</i> ... 9	<i>FD_ISSET()</i> ... 7
41	<i>div()</i> ... 6	<i>FD_SET()</i> ... 7
42	<i>dlclose()</i> ... 9	<i>FD_ZERO()</i> ... 7
43	<i>dLError()</i> ... 9	<i>fdatasync()</i> ... 116
44	<i>dlopen()</i> ... 9	<i>fdetach()</i> ... 120
45	<i>dlsym()</i> ... 9	<i>fdim()</i> ... 5
46	<i>drand48()</i> ... 9	<i>fdimf()</i> ... 5
47	<i>dup()</i> ... 7	<i>fdiml()</i> ... 5
48	<i>dup2()</i> ... 7	<i>fdopen()</i> ... 6
49	<i>ecvt()</i> ... 120	<i>feclearexcept()</i> ... 6
	<i>encrypt()</i> ... 120	<i>fegetenv()</i> ... 6
	<i>endgrent()</i> ... 10	<i>fegetexceptflag()</i> ... 6
	<i>endhostent()</i> ... 7	<i>fegetround()</i> ... 6

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1	<i>fehldexcept()</i> ... 6	<i>fseek()</i> ... 7
2	<i>feof()</i> ... 6	<i>fseeko()</i> ... 7
3	<i>feraiseexcept()</i> ... 6	<i>fsetpos()</i> ... 7
4	<i>ferror()</i> ... 6	<i>fstat()</i> ... 7
5	<i>fesetenv()</i> ... 6	<i>fstatvfs()</i> ... 9
6	<i>fesetexceptflag()</i> ... 6	<i>fsync()</i> ... 114
7	<i>fesetround()</i> ... 6	<i>ftell()</i> ... 7
8	<i>fetetestexcept()</i> ... 6	<i>ftello()</i> ... 7
9	<i>feupdateenv()</i> ... 6	<i>ftime()</i> ... 120
10	<i>fflush()</i> ... 6, 44	<i>ftok()</i> ... 9
11	<i>ffs()</i> ... 9	<i>ftruncate()</i> ... 7
12	<i>fgetc()</i> ... 6, 44	<i>ftrylockfile()</i> ... 7, 116
13	<i>fgetpos()</i> ... 7	<i>ftw()</i> ... 9
14	<i>fgets()</i> ... 6, 44	<i>funlockfile()</i> ... 7, 116
15	<i>fgetwc()</i> ... 9	<i>fwide()</i> ... 9
16	<i>fgetws()</i> ... 9	<i>fwprintf()</i> ... 9
17	<i>fileno()</i> ... 6	<i>fwrite()</i> ... 6, 44
18	<i>flockfile()</i> ... 7, 116	<i>fwscanf()</i> ... 9
19	<i>floor()</i> ... 5	<i>gai_strerror()</i> ... 7
20	<i>floorf()</i> ... 5	<i>gcvt()</i> ... 120
21	<i>floorl()</i> ... 5	<i>getaddrinfo()</i> ... 7
22	<i>fma()</i> ... 5	<i>getc()</i> ... 6, 44
23	<i>fmaf()</i> ... 5	<i>getc_unlocked()</i> ... 7, 116
24	<i>fmal()</i> ... 5	<i>getchar()</i> ... 6, 44
25	<i>fmax()</i> ... 5	<i>getchar_unlocked()</i> ... 7, 116
26	<i>fmaxf()</i> ... 5	<i>getcontext()</i> ... 9
27	<i>fmaxl()</i> ... 5	<i>getcwd()</i> ... 7
28	<i>fmin()</i> ... 5	<i>getdate()</i> ... 9
29	<i>fminf()</i> ... 5	<i>getegid()</i> ... 9
30	<i>fminl()</i> ... 5	<i>getenv()</i> ... 8
31	<i>fmod()</i> ... 5	<i>geteuid()</i> ... 9
32	<i>fmodf()</i> ... 5	<i>getgid()</i> ... 9
33	<i>fmodl()</i> ... 5	<i>getgrent()</i> ... 10
34	<i>fntmsg()</i> ... 9	<i>getgrgid()</i> ... 8
35	<i>fnmatch()</i> ... 8	<i>getgrgid_r()</i> ... 8, 116
36	<i>fopen()</i> ... 6, 44	<i>getgrnam()</i> ... 8
37	<i>fork()</i> ... 7	<i>getgrnam_r()</i> ... 8, 116
38	<i>fpathconf()</i> ... 7	<i>getgroups()</i> ... 9
39	<i>fpclassify()</i> ... 5	<i>gethostbyaddr()</i> ... 7
40	<i>fprintf()</i> ... 6, 44	<i>gethostbyname()</i> ... 7
41	<i>fputc()</i> ... 6, 44	<i>gethostent()</i> ... 7
42	<i>fputs()</i> ... 6, 44	<i>gethostid()</i> ... 9
43	<i>fputwc()</i> ... 9	<i>gethostname()</i> ... 7
44	<i>fputws()</i> ... 9	<i>getitimer()</i> ... 10
45	<i>fread()</i> ... 6, 44	<i>getlogin()</i> ... 9
46	<i>free()</i> ... 6	<i>getlogin_r()</i> ... 9, 116
47	<i>freeaddrinfo()</i> ... 7	<i>getmsg()</i> ... 120
48	<i>freopen()</i> ... 6, 44	<i>getnameinfo()</i> ... 7
49	<i>frexp()</i> ... 5	<i>getnetbyaddr()</i> ... 7
	<i>frexpf()</i> ... 5	<i>getnetbyname()</i> ... 7
	<i>frexpl()</i> ... 5	<i>getnetent()</i> ... 7
	<i>fscanf()</i> ... 6, 44	<i>getopt()</i> ... 8

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	<i>getpeername()</i> ... 7	<i>if_nametoindex()</i> ... 7
2	<i>getpgid()</i> ... 9	<i>ilogb()</i> ... 5
3	<i>getpgrp()</i> ... 7	<i>ilogbf()</i> ... 5
4	<i>getpid()</i> ... 7	<i>ilogbl()</i> ... 5
5	<i>getpmsg()</i> ... 120	<i>imaxabs()</i> ... 6
6	<i>getppid()</i> ... 7	<i>imaxdiv()</i> ... 6
7	<i>getpriority()</i> ... 9	<i>index()</i> ... 120
8	<i>getprotobyname()</i> ... 7	<i>inet_addr()</i> ... 7
9	<i>getprotobynumber()</i> ... 7	<i>inet_ntoa()</i> ... 7
10	<i>getprotoent()</i> ... 7	<i>inet_ntop()</i> ... 7
11	<i>getpwent()</i> ... 9	<i>inet_pton()</i> ... 7
12	<i>getpwnam()</i> ... 8	<i>initstate()</i> ... 9
13	<i>getpwnam_r()</i> ... 8, 116	<i>insque()</i> ... 9
14	<i>getpwuid()</i> ... 8	<i>ioctl()</i> ... xv, 120
15	<i>getpwuid_r()</i> ... 8, 116	<i>isalnum()</i> ... 6
16	<i>getrlimit()</i> ... 9	<i>isalpha()</i> ... 6
17	<i>getrusage()</i> ... 9	<i>isascii()</i> ... 9
18	<i>gets()</i> ... 6, 44	<i>isastream()</i> ... 120
19	<i>getservbyname()</i> ... 7	<i>isatty()</i> ... 7
20	<i>getservbyport()</i> ... 7	<i>isblank()</i> ... 6
21	<i>getservent()</i> ... 7	<i>iscntrl()</i> ... 6
22	<i>getsid()</i> ... 9	<i>isdigit()</i> ... 6
23	<i>getsockname()</i> ... 7	<i>isfinite()</i> ... 5
24	<i>getsockopt()</i> ... 7	<i>isgraph()</i> ... 6
25	<i>getsubopt()</i> ... 9	<i>isgreater()</i> ... 5
26	<i>gettimeofday()</i> ... 9	<i>isgreaterequal()</i> ... 5
27	<i>getuid()</i> ... 9	<i>isinf()</i> ... 5
28	<i>getutxent()</i> ... 10	<i>isless()</i> ... 5
29	<i>getutxid()</i> ... 10	<i>islessequal()</i> ... 5
30	<i>getutxline()</i> ... 10	<i>islessgreater()</i> ... 5
31	<i>getwc()</i> ... 9	<i>islower()</i> ... 6
32	<i>getwchar()</i> ... 9	<i>isnan()</i> ... 5
33	<i>getwd()</i> ... 120	<i>isnormal()</i> ... 5
34	<i>glob()</i> ... 7	<i>isprint()</i> ... 6
35	<i>globfree()</i> ... 7	<i>ispunct()</i> ... 6
36	<i>gmtime()</i> ... 6	<i>isspace()</i> ... 6
37	<i>gmtime_r()</i> ... 6, 116	<i>isunordered()</i> ... 5
38	<i>grantpt()</i> ... 9	<i>isupper()</i> ... 6
39	<i>hcreate()</i> ... 9	<i>iswalnum()</i> ... 6
40	<i>hdestroy()</i> ... 9	<i>iswalpha()</i> ... 6
41	<i>hsearch()</i> ... 9	<i>iswblank()</i> ... 6
42	<i>htonl()</i> ... 7	<i>iswcntrl()</i> ... 6
43	<i>htons()</i> ... 7	<i>iswctype()</i> ... 6
44	<i>hypot()</i> ... 5	<i>iswdigit()</i> ... 6
45	<i>hypotf()</i> ... 5	<i>iswgraph()</i> ... 6
46	<i>hypotl()</i> ... 5	<i>iswlower()</i> ... 6
47	<i>iconv()</i> ... 9	<i>iswprint()</i> ... 6
48	<i>iconv_close()</i> ... 9	<i>iswpunct()</i> ... 6
49	<i>iconv_open()</i> ... 9	<i>iswspace()</i> ... 6
	<i>if_freenameindex()</i> ... 7	<i>iswupper()</i> ... 6
	<i>if_indextoname()</i> ... 7	<i>iswxdigit()</i> ... 6
	<i>if_nameindex()</i> ... 7	<i>isxdigit()</i> ... 6

1	<i>j0()</i> ... 9	<i>lrintl()</i> ... 5
2	<i>j1()</i> ... 9	<i>lround()</i> ... 5
3	<i>jn()</i> ... 9	<i>lroundf()</i> ... 5
4	<i>jrands48()</i> ... 9	<i>lroundl()</i> ... 5
5	<i>kill()</i> ... 8, 44, 62, 80	<i>lsearch()</i> ... 9
6	<i>killpg()</i> ... 9	<i>lseek()</i> ... 7
7	<i>l64a()</i> ... 9	<i>lstat()</i> ... 8
8	<i>labs()</i> ... 6	<i>main()</i> ... 48, 65
9	<i>lchown()</i> ... 9	<i>makecontext()</i> ... 9
10	<i>lcong48()</i> ... 9	<i>malloc()</i> ... 6
11	<i>ldexp()</i> ... 5	<i>mblen()</i> ... 6
12	<i>ldexpf()</i> ... 5	<i>mbrlen()</i> ... 6
13	<i>ldexpl()</i> ... 5	<i>mbtowc()</i> ... 6
14	<i>ldiv()</i> ... 6	<i>mbsinit()</i> ... 6
15	<i>lfind()</i> ... 9	<i>mbsrtowcs()</i> ... 6
16	<i>lgamma()</i> ... 5	<i>mbstowcs()</i> ... 6
17	<i>lgammaf()</i> ... 5	<i>mbtowc()</i> ... 6
18	<i>lgammal()</i> ... 5	<i>memccpy()</i> ... 9
19	<i>link()</i> ... xvii, 7	<i>memchr()</i> ... 6
20	<i>lio_listio()</i> ... 113	<i>memcmp()</i> ... 6
21	<i>listen()</i> ... 7	<i>memcpy()</i> ... 6
22	<i>llabs()</i> ... 6	<i>memmove()</i> ... 6
23	<i>lldiv()</i> ... 6	<i>memset()</i> ... 6
24	<i>llrint()</i> ... 5	<i>mkdir()</i> ... xvii, 7
25	<i>llrintf()</i> ... 5	<i>mkfifo()</i> ... 7
26	<i>llrintl()</i> ... 5	<i>mknod()</i> ... 9
27	<i>llround()</i> ... 5	<i>mkstemp()</i> ... 9
28	<i>llroundf()</i> ... 5	<i>mktemp()</i> ... 120
29	<i>llroundl()</i> ... 5	<i>mkttime()</i> ... 6
30	<i>localeconv()</i> ... 6	<i>mlock()</i> ... 114
31	<i>localtime()</i> ... 6	<i>mlockall()</i> ... 114
32	<i>localtime_r()</i> ... 6, 116	<i>mmap()</i> ... 114, 115
33	<i>lockf()</i> ... 9	<i>modf()</i> ... 5
34	<i>log()</i> ... 5	<i>modff()</i> ... 5
35	<i>log10()</i> ... 5	<i>modfl()</i> ... 5
36	<i>log10f()</i> ... 5	<i>mprotect()</i> ... 114
37	<i>log10l()</i> ... 5	<i>mq_close()</i> ... 114
38	<i>log1p()</i> ... 5	<i>mq_getattr()</i> ... 114
39	<i>log1pf()</i> ... 5	<i>mq_notify()</i> ... 114
40	<i>log1pl()</i> ... 5	<i>mq_open()</i> ... 114
41	<i>log2()</i> ... 5	<i>mq_receive()</i> ... 114
42	<i>log2f()</i> ... 5	<i>mq_send()</i> ... 114
43	<i>log2l()</i> ... 5	<i>mq_setattr()</i> ... 114
44	<i>logb()</i> ... 5	<i>mq_timedreceive()</i> ... 114, 118
45	<i>logbf()</i> ... 5	<i>mq_timedsend()</i> ... 114, 118
46	<i>logbl()</i> ... 5	<i>mq_unlink()</i> ... 114
47	<i>logf()</i> ... 5	<i>mrands48()</i> ... 9
48	<i>logl()</i> ... 5	<i>msgctl()</i> ... 9
49	<i>longjmp()</i> ... 4	<i>msgget()</i> ... 9
	<i>lrands48()</i> ... 9	<i>msgrcv()</i> ... 9
	<i>lrint()</i> ... 5	<i>msgsnd()</i> ... 9
	<i>lrintf()</i> ... 5	<i>msync()</i> ... 114, 116

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	<i>munlock()</i> ... 114	114, 115
2	<i>munlockall()</i> ... 114	<i>posix_spawnattr_getschedpolicy()</i> ...
3	<i>munmap()</i> ... 114, 115	114, 115
4	<i>nan()</i> ... 5	<i>posix_spawnattr_getsigdefault()</i> ... 115
5	<i>nanf()</i> ... 5	<i>posix_spawnattr_getsigmask()</i> ... 115
6	<i>nanl()</i> ... 5	<i>posix_spawnattr_init()</i> ... 115
7	<i>nanosleep()</i> ... 118	<i>posix_spawnattr_setflags()</i> ... 115
8	<i>nearbyint()</i> ... 5	<i>posix_spawnattr_setpgroup()</i> ... 115
9	<i>nearbyintf()</i> ... 5	<i>posix_spawnattr_setschedparam()</i> ...
10	<i>nearbyintl()</i> ... 5	114, 115
11	<i>nextafter()</i> ... 5	<i>posix_spawnattr_setschedpolicy()</i> ...
12	<i>nextafterf()</i> ... 5	114, 115
13	<i>nextafterl()</i> ... 5	<i>posix_spawnattr_setsigdefault()</i> ... 115
14	<i>nexttoward()</i> ... 5	<i>posix_spawnattr_setsigmask()</i> ... 115
15	<i>nexttowardf()</i> ... 5	<i>posix_spawnnp()</i> ... 115
16	<i>nexttowardl()</i> ... 5	<i>posix_trace_attr_destroy()</i> ... 119
17	<i>nftw()</i> ... 9	<i>posix_trace_attr_getclockres()</i> ... 119
18	<i>nice()</i> ... 9	<i>posix_trace_attr_getcreatetime()</i> ... 119
19	<i>nl_langinfo()</i> ... 9	<i>posix_trace_attr_getgenversion()</i> ... 119
20	<i>nrnd48()</i> ... 9	<i>posix_trace_attr_getinherited()</i> ... 119
21	<i>ntohl()</i> ... 7	<i>posix_trace_attr_getlogfullpolicy()</i> ...
22	<i>ntohs()</i> ... 7	119
23	<i>open()</i> ... xv, 6, 44, 49	<i>posix_trace_attr_getlogsize()</i> ... 119
24	<i>opendir()</i> ... 7	<i>posix_trace_attr_getmaxdatasize()</i> ...
25	<i>openlog()</i> ... 9	119
26	<i>pathconf()</i> ... 7	<i>posix_trace_attr_getmaxsystemeventsi</i>
27	<i>pause()</i> ... 8	<i>ze()</i> ... 119
28	<i>pclose()</i> ... 8	<i>posix_trace_attr_getmaxusereventsiz</i>
29	<i>perror()</i> ... 6, 44	<i>ze()</i> ... 119
30	<i>pipe()</i> ... 7	<i>posix_trace_attr_getname()</i> ... 119
31	<i>poll()</i> ... 9	<i>posix_trace_attr_getstreamfullpolicy()</i> .
32	<i>popen()</i> ... 8	.. 119
33	<i>posix_fadvise()</i> ... 113	<i>posix_trace_attr_getstreamsize()</i> ... 119
34	<i>posix_fallocate()</i> ... 113	<i>posix_trace_attr_init()</i> ... 119
35	<i>posix_madvise()</i> ... 113, 114, 115	<i>posix_trace_attr_setinherited()</i> ... 119
36	<i>posix_mem_offset()</i> ... 119	<i>posix_trace_attr_setlogfullpolicy()</i> ...
37	<i>posix_memalign()</i> ... 113	119
38	<i>posix_openpt()</i> ... 9	<i>posix_trace_attr_setlogsize()</i> ... 119
39	<i>posix_spawn()</i> ... 115	<i>posix_trace_attr_setmaxdatasize()</i> ...
40	<i>posix_spawn_file_actions_addclose()</i> ...	119
41	115	<i>posix_trace_attr_setname()</i> ... 119
42	<i>posix_spawn_file_actions_adddup2()</i> ...	<i>posix_trace_attr_setstreamfullpolicy()</i> .
43	115	.. 119
44	<i>posix_spawn_file_actions_addopen()</i> ...	<i>posix_trace_attr_setstreamsize()</i> ... 119
45	115	<i>posix_trace_clear()</i> ... 119
46	<i>posix_spawn_file_actions_destroy()</i> ...	<i>posix_trace_close()</i> ... 119
47	115	<i>posix_trace_create()</i> ... 119
48	<i>posix_spawn_file_actions_init()</i> ... 115	<i>posix_trace_create_withlog()</i> ... 119
49	<i>posix_spawnattr_destroy()</i> ... 115	<i>posix_trace_event()</i> ... 119
	<i>posix_spawnattr_getflags()</i> ... 115	<i>posix_trace_eventid_equal()</i> ... 119
	<i>posix_spawnattr_getpgroup()</i> ... 115	<i>posix_trace_eventid_get_name()</i> ... 119
	<i>posix_spawnattr_getschedparam()</i> ...	<i>posix_trace_eventid_open()</i> ... 119

1	<i>posix_trace_eventset_add()</i> ... 119	<i>pthread_attr_setschedpolicy()</i> ... 116,
2	<i>posix_trace_eventset_del()</i> ... 119	118
3	<i>posix_trace_eventset_empty()</i> ... 119	<i>pthread_attr_setscope()</i> ... 116, 118
4	<i>posix_trace_eventset_fill()</i> ... 119	<i>pthread_attr_setstack()</i> ... 10, 116, 117
5	<i>posix_trace_eventset_ismember()</i> ... 119	<i>pthread_attr_setstackaddr()</i> ... 116, 117
6	<i>posix_trace_eventtypelist_getnext_id()</i>	<i>pthread_attr_setstacksize()</i> ... 116, 117,
7	.. 119	121
8	<i>posix_trace_eventtypelist_rewind()</i> ...	<i>pthread_barrier_destroy()</i> ... 113, 117
9	119	<i>pthread_barrier_init()</i> ... 113, 117
10	<i>posix_trace_flush()</i> ... 119	<i>pthread_barrier_wait()</i> ... 113, 117
11	<i>posix_trace_get_attr()</i> ... 119	<i>pthread_barrierattr_destroy()</i> ... 113,
12	<i>posix_trace_get_filter()</i> ... 119	117
13	<i>posix_trace_get_status()</i> ... 119	<i>pthread_barrierattr_getpshared()</i> ...
14	<i>posix_trace_getnext_event()</i> ... 119	113, 116, 117
15	<i>posix_trace_open()</i> ... 119	<i>pthread_barrierattr_init()</i> ... 113, 117
16	<i>posix_trace_rewind()</i> ... 119	<i>pthread_barrierattr_setpshared()</i> ...
17	<i>posix_trace_set_filter()</i> ... 119	113, 116, 117
18	<i>posix_trace_shutdown()</i> ... 119	<i>pthread_cancel()</i> ... 8, 117
19	<i>posix_trace_start()</i> ... 119	<i>pthread_cleanup_pop()</i> ... 8, 117
20	<i>posix_trace_stop()</i> ... 119	<i>pthread_cleanup_push()</i> ... 8, 117
21	<i>posix_trace_timedgetnext_event()</i> ...	<i>pthread_cond_broadcast()</i> ... 8, 117
22	118, 119	<i>pthread_cond_destroy()</i> ... 8, 117
23	<i>posix_trace_trid_eventid_open()</i> ... 119	<i>pthread_cond_init()</i> ... 8, 117
24	<i>posix_trace_trygetnext_event()</i> ... 119	<i>pthread_cond_signal()</i> ... 8, 117
25	<i>posix_typed_mem_get_info()</i> ... 119	<i>pthread_cond_timedwait()</i> ... 8, 117
26	<i>posix_typed_mem_open()</i> ... 119	<i>pthread_cond_wait()</i> ... 8, 117
27	<i>pow()</i> ... 5	<i>pthread_condattr_destroy()</i> ... 8, 117
28	<i>powf()</i> ... 5	<i>pthread_condattr_getclock()</i> ... 113, 117
29	<i>powl()</i> ... 5	<i>pthread_condattr_getpshared()</i> ... 116,
30	<i>pread()</i> ... 9	118
31	<i>printf()</i> ... 6, 44	<i>pthread_condattr_init()</i> ... 8, 117
32	<i>pselect()</i> ... 7	<i>pthread_condattr_setclock()</i> ... 113, 117
33	<i>pthread_atfork()</i> ... 8, 117	<i>pthread_condattr_setpshared()</i> ... 116,
34	<i>pthread_attr_destroy()</i> ... 8, 117	118
35	<i>pthread_attr_getdetachstate()</i> ... 8, 117	<i>pthread_create()</i> ... 8, 117
36	<i>pthread_attr_getguardsize()</i> ... 10	<i>pthread_detach()</i> ... 8, 117
37	<i>pthread_attr_getinheritsched()</i> ... 116,	<i>pthread_equal()</i> ... 8, 117
38	118	<i>pthread_exit()</i> ... 8, 117
39	<i>pthread_attr_getschedparam()</i> ... 8, 117	<i>pthread_getconcurrency()</i> ... 10
40	<i>pthread_attr_getschedpolicy()</i> ... 116,	<i>pthread_getcpuclid()</i> ... 116, 117
41	118	<i>pthread_getschedparam()</i> ... 116, 118
42	<i>pthread_attr_getscope()</i> ... 116, 118	<i>pthread_getspecific()</i> ... 8, 117
43	<i>pthread_attr_getstack()</i> ... 10, 116, 117	<i>pthread_join()</i> ... 8, 117
44	<i>pthread_attr_getstackaddr()</i> ... 116, 117	<i>pthread_key_create()</i> ... 8, 117
45	<i>pthread_attr_getstacksize()</i> ... 116, 117,	<i>pthread_key_delete()</i> ... 8, 117
46	121	<i>pthread_kill()</i> ... 8, 117
47	<i>pthread_attr_init()</i> ... 8, 117	<i>pthread_mutex_destroy()</i> ... 8, 117
48	<i>pthread_attr_setdetachstate()</i> ... 8, 117	<i>pthread_mutex_getprioceiling()</i> ... 116,
49	<i>pthread_attr_setguardsize()</i> ... 10	118
	<i>pthread_attr_setinheritsched()</i> ... 116,	<i>pthread_mutex_init()</i> ... 8, 117
	118	<i>pthread_mutex_lock()</i> ... 8, 117
	<i>pthread_attr_setschedparam()</i> ... 8, 117	<i>pthread_mutex_setprioceiling()</i> ... 116,

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	118	<i>putc_unlocked()</i> ... 7, 116
2	<i>pthread_mutex_timedlock()</i> ... 118	<i>putchar()</i> ... 6, 44
3	<i>pthread_mutex_trylock()</i> ... 8, 117	<i>putchar_unlocked()</i> ... 7, 116
4	<i>pthread_mutex_unlock()</i> ... 8, 117	<i>putenv()</i> ... 9
5	<i>pthread_mutexattr_destroy()</i> ... 8, 117	<i>putmsg()</i> ... 120
6	<i>pthread_mutexattr_getprioceiling()</i> ...	<i>putpmsg()</i> ... 120
7	116, 118	<i>puts()</i> ... 6, 44
8	<i>pthread_mutexattr_getprotocol()</i> ... 116,	<i>pututxline()</i> ... 10
9	117	<i>putwc()</i> ... 9
10	<i>pthread_mutexattr_getpshared()</i> ... 116,	<i>putwchar()</i> ... 9
11	118	<i>pwrite()</i> ... 9
12	<i>pthread_mutexattr_gettype()</i> ... 10	<i>qsort()</i> ... 6
13	<i>pthread_mutexattr_init()</i> ... 8, 117	<i>raise()</i> ... 8, 44, 62
14	<i>pthread_mutexattr_setprioceiling()</i> ...	<i>rand()</i> ... 6
15	116, 118	<i>rand_r()</i> ... 6, 116
16	<i>pthread_mutexattr_setprotocol()</i> ... 116,	<i>random()</i> ... 9
17	117	<i>read()</i> ... xv, xvii, 6, 44
18	<i>pthread_mutexattr_setpshared()</i> ... 116,	<i>readdir()</i> ... 7
19	118	<i>readdir_r()</i> ... 7, 116
20	<i>pthread_mutexattr_settype()</i> ... 10	<i>readlink()</i> ... 8
21	<i>pthread_once()</i> ... 8, 117	<i>readv()</i> ... 9
22	<i>pthread_rwlock_destroy()</i> ... 8, 118	<i>realloc()</i> ... 6
23	<i>pthread_rwlock_init()</i> ... 8, 118	<i>realpath()</i> ... 9
24	<i>pthread_rwlock_rdlock()</i> ... 8, 118	<i>recv()</i> ... 7
25	<i>pthread_rwlock_timedrdlock()</i> ... 8, 118	<i>recvfrom()</i> ... 7
26	<i>pthread_rwlock_timedwrlock()</i> ... 8, 118	<i>recvmsg()</i> ... 7
27	<i>pthread_rwlock_tryrdlock()</i> ... 8, 118	<i>regcomp()</i> ... 7
28	<i>pthread_rwlock_trywrlock()</i> ... 8, 118	<i>regerror()</i> ... 7
29	<i>pthread_rwlock_unlock()</i> ... 8, 118	<i>regexexec()</i> ... 7
30	<i>pthread_rwlock_wrlock()</i> ... 8, 118	<i>regfree()</i> ... 7
31	<i>pthread_rwlockattr_destroy()</i> ... 8, 118	<i>remainder()</i> ... 5
32	<i>pthread_rwlockattr_getpshared()</i> ... 8,	<i>remainderf()</i> ... 5
33	116, 118	<i>remainderl()</i> ... 5
34	<i>pthread_rwlockattr_init()</i> ... 8, 118	<i>remove()</i> ... 7
35	<i>pthread_rwlockattr_setpshared()</i> ... 8,	<i>remque()</i> ... 9
36	116, 118	<i>remquo()</i> ... 5
37	<i>pthread_self()</i> ... 8, 117	<i>remquof()</i> ... 5
38	<i>pthread_setcancelstate()</i> ... 8, 117	<i>remquol()</i> ... 5
39	<i>pthread_setcanceltype()</i> ... 8, 117	<i>rename()</i> ... xvii, 7
40	<i>pthread_setconcurrency()</i> ... 10	<i>rewind()</i> ... 7
41	<i>pthread_setschedparam()</i> ... 116, 118	<i>rewinddir()</i> ... 7
42	<i>pthread_setschedprio()</i> ... 116, 118	<i>rindex()</i> ... 120
43	<i>pthread_setspecific()</i> ... 8, 117	<i>rint()</i> ... 5
44	<i>pthread_sigmask()</i> ... 8, 117	<i>rintf()</i> ... 5
45	<i>pthread_spin_destroy()</i> ... 115, 117	<i>rintl()</i> ... 5
46	<i>pthread_spin_init()</i> ... 115, 117	<i>rmdir()</i> ... xvii, 7
47	<i>pthread_spin_lock()</i> ... 115, 117	<i>round()</i> ... 5
48	<i>pthread_spin_trylock()</i> ... 115, 117	<i>roundf()</i> ... 5
49	<i>pthread_spin_unlock()</i> ... 115, 117	<i>roundl()</i> ... 5
	<i>pthread_testcancel()</i> ... 8, 117	<i>scalb()</i> ... 9
	<i>ptsname()</i> ... 9	<i>scalbln()</i> ... 5
	<i>putc()</i> ... 6, 44	<i>scalblnf()</i> ... 5

1	<i>scalbnl()</i> ... 5	<i>setreuid()</i> ... 10
2	<i>scalbn()</i> ... 5	<i>setrlimit()</i> ... 9
3	<i>scalbnf()</i> ... 5	<i>setservent()</i> ... 7
4	<i>scalbnl()</i> ... 5	<i>setsid()</i> ... 7
5	<i>scanf()</i> ... 6, 44	<i>setsockopt()</i> ... 7
6	<i>sched_get_priority_max()</i> ... 7, 114	<i>setstate()</i> ... 9
7	<i>sched_get_priority_min()</i> ... 7, 114	<i>setuid()</i> ... 9
8	<i>sched_getparam()</i> ... 114	<i>setutxent()</i> ... 10
9	<i>sched_getscheduler()</i> ... 114	<i>setvbuf()</i> ... 6
10	<i>sched_rr_get_interval()</i> ... 7, 114	<i>shm_open()</i> ... 115
11	<i>sched_setparam()</i> ... 114	<i>shm_unlink()</i> ... 115
12	<i>sched_setscheduler()</i> ... 114	<i>shmat()</i> ... 9
13	<i>sched_yield()</i> ... 114, 118	<i>shmctl()</i> ... 9
14	<i>seed48()</i> ... 9	<i>shmdt()</i> ... 9
15	<i>seekdir()</i> ... 9	<i>shmget()</i> ... 9
16	<i>select()</i> ... 7, 54, 71, 89, 108	<i>shutdown()</i> ... 7
17	<i>sem_close()</i> ... 115	<i>sigaction()</i> ... 8, 44, 62
18	<i>sem_destroy()</i> ... 115	<i>sigaddset()</i> ... 8, 44, 62
19	<i>sem_getvalue()</i> ... 115	<i>sigaltstack()</i> ... 9
20	<i>sem_init()</i> ... 115	<i>sigdelset()</i> ... 8, 62
21	<i>sem_open()</i> ... 115	<i>sigemptyset()</i> ... 8, 44, 62
22	<i>sem_post()</i> ... 44, 62, 115	<i>sigfillset()</i> ... 8, 44, 62
23	<i>sem_timedwait()</i> ... 115, 118	<i>sighold()</i> ... 9
24	<i>sem_trywait()</i> ... 115	<i>sigignore()</i> ... 9
25	<i>sem_unlink()</i> ... 115	<i>siginterrupt()</i> ... 9
26	<i>sem_wait()</i> ... 115	<i>sigismember()</i> ... 8, 44, 62
27	<i>semctl()</i> ... 9	<i>siglongjmp()</i> ... 8
28	<i>semget()</i> ... 9	<i>signal()</i> ... 8, 44, 62
29	<i>semop()</i> ... 9	<i>signbit()</i> ... 5
30	<i>send()</i> ... 7	<i>sigpause()</i> ... 9
31	<i>sendmsg()</i> ... 7	<i>sigpending()</i> ... 8, 44, 62
32	<i>sendto()</i> ... 7	<i>sigprocmask()</i> ... 8, 44, 62
33	<i>setbuf()</i> ... 6	<i>sigqueue()</i> ... 44, 62, 115
34	<i>setcontext()</i> ... 9	<i>sigrelse()</i> ... 9
35	<i>setegid()</i> ... 9	<i>sigset()</i> ... 44, 62
36	<i>setenv()</i> ... 8	<i>sigsetjmp()</i> ... 8
37	<i>seteuid()</i> ... 9	<i>sigsuspend()</i> ... 8
38	<i>setgid()</i> ... 9	<i>sigtimedwait()</i> ... 115
39	<i>setgrent()</i> ... 10	<i>sigwait()</i> ... 8
40	<i>sethostent()</i> ... 7	<i>sigwaitinfo()</i> ... 115
41	<i>setitimer()</i> ... 10	<i>sin()</i> ... 5
42	<i>setjmp()</i> ... 4	<i>sinf()</i> ... 5
43	<i>setkey()</i> ... 120	<i>sinh()</i> ... 5
44	<i>setlocale()</i> ... 6	<i>sinhf()</i> ... 5
45	<i>setlogmask()</i> ... 9	<i>sinhl()</i> ... 5
46	<i>setnetent()</i> ... 7	<i>sinl()</i> ... 5
47	<i>setpgid()</i> ... 7	<i>sleep()</i> ... 7, 52, 69, 88, 106
48	<i>setpgrp()</i> ... 9	<i>snprintf()</i> ... 6
49	<i>setpriority()</i> ... 9	<i>socketatmark()</i> ... 7
	<i>setprotoent()</i> ... 7	<i>socket()</i> ... 7
	<i>setpwent()</i> ... 9	<i>socketpair()</i> ... 7
	<i>setregid()</i> ... 10	<i>sprintf()</i> ... 6

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	<i>sqrt()</i> ... 5	<i>tan()</i> ... 5
2	<i>sqrtf()</i> ... 5	<i>tanf()</i> ... 5
3	<i>sqrtl()</i> ... 5	<i>tanh()</i> ... 5
4	<i>srand()</i> ... 6	<i>tanhf()</i> ... 5
5	<i>srand48()</i> ... 9	<i>tanhhl()</i> ... 5
6	<i>srandom()</i> ... 9	<i>tanl()</i> ... 5
7	<i>sscanf()</i> ... 6	<i>tcdrain()</i> ... 7
8	<i>stat()</i> ... 7	<i>tcflow()</i> ... 7
9	<i>statvfs()</i> ... 9	<i>tcflush()</i> ... 7
10	<i>strcasecmp()</i> ... 9	<i>tcgetattr()</i> ... 7
11	<i>strcat()</i> ... 6	<i>tcgetpgrp()</i> ... 7
12	<i>strchr()</i> ... 6	<i>tcgetsid()</i> ... 9
13	<i>strcmp()</i> ... 6	<i>tcsendbreak()</i> ... 7
14	<i>strcoll()</i> ... 6	<i>tcsetattr()</i> ... 7
15	<i>strcpy()</i> ... 6	<i>tcsetpgrp()</i> ... 7
16	<i>strcspn()</i> ... 6	<i>tdelete()</i> ... 9
17	<i>strdup()</i> ... 9	<i>telldir()</i> ... 9
18	<i>strerror()</i> ... 6	<i>tempnam()</i> ... 9
19	<i>strerror_r()</i> ... 6, 116	<i>tfind()</i> ... 9
20	<i>strfmon()</i> ... 9	<i>tgamma()</i> ... 5
21	<i>strftime()</i> ... 6	<i>tgammaf()</i> ... 5
22	<i>strlen()</i> ... 6	<i>tgammaL()</i> ... 5
23	<i>strncasecmp()</i> ... 9	<i>time()</i> ... 6, 44, 62
24	<i>strncat()</i> ... 6	<i>timer_create()</i> ... 118
25	<i>strncpy()</i> ... 6	<i>timer_delete()</i> ... 118
26	<i>strpbrk()</i> ... 6	<i>timer_getoverrun()</i> ... 44, 62, 118
27	<i>strptime()</i> ... 9	<i>timer_gettime()</i> ... 44, 62, 118
28	<i>strrchr()</i> ... 6	<i>timer_settime()</i> ... 44, 62, 118
29	<i>strspn()</i> ... 6	<i>times()</i> ... 7, 44, 62
30	<i>strstr()</i> ... 6	<i>timezone()</i> ... 9
31	<i>strtod()</i> ... 6	<i>tmpfile()</i> ... 7
32	<i>strtof()</i> ... 6	<i>tmpnam()</i> ... 7
33	<i>strtoimax()</i> ... 6	<i>toascii()</i> ... 9
34	<i>strtok()</i> ... 6	<i>tolower()</i> ... 6
35	<i>strtok_r()</i> ... 6, 116	<i>toupper()</i> ... 6
36	<i>strtol()</i> ... 6	<i>towctrans()</i> ... 6
37	<i>strtold()</i> ... 6	<i>towlower()</i> ... 6
38	<i>strtoll()</i> ... 6	<i>toupper()</i> ... 6
39	<i>strtoul()</i> ... 6	<i>trunc()</i> ... 5
40	<i>strtoull()</i> ... 6	<i>truncate()</i> ... 9
41	<i>strtoimax()</i> ... 6	<i>truncf()</i> ... 5
42	<i>strxfrm()</i> ... 6	<i>truncl()</i> ... 5
43	<i>swab()</i> ... 9	<i>tsearch()</i> ... 9
44	<i>swapcontext()</i> ... 9	<i>ttynam()</i> ... 7
45	<i>swprintf()</i> ... 6	<i>ttynam_r()</i> ... 7, 116
46	<i>swscanf()</i> ... 6	<i>twalk()</i> ... 9
47	<i>symlink()</i> ... 8	<i>tzname()</i> ... 6
48	<i>sync()</i> ... 9	<i>tzset()</i> ... 6
49	<i>sysconf()</i> ... 8, 44, 62, 80	<i>ualarm()</i> ... 9
	<i>syslog()</i> ... 9	<i>ulimit()</i> ... 9
	<i>system()</i> ... 8	<i>umask()</i> ... 7
		<i>uname()</i> ... 8, 44, 62

1	<i>ungetc()</i> ... 6	<i>wcstoll()</i> ... 6
2	<i>ungetwc()</i> ... 9	<i>wcstombs()</i> ... 6
3	<i>unlink()</i> ... xvii, 7	<i>wcstoul()</i> ... 6
4	<i>unlockpt()</i> ... 9	<i>wcstoull()</i> ... 6
5	<i>unsetenv()</i> ... 8	<i>wcstoumax()</i> ... 6
6	<i>usleep()</i> ... 9	<i>wcswcs()</i> ... 120
7	<i>utime()</i> ... 7	<i>wcswidth()</i> ... 10
8	<i>utimes()</i> ... 120	<i>wcsxfrm()</i> ... 6
9	<i>va_arg()</i> ... 6	<i>wctob()</i> ... 6
10	<i>va_copy()</i> ... 6	<i>wctomb()</i> ... 6
11	<i>va_end()</i> ... 6	<i>wctrans()</i> ... 6
12	<i>va_start()</i> ... 6	<i>wctype()</i> ... 6
13	<i>vfork()</i> ... 9	<i>wcwidth()</i> ... 10
14	<i>vfprintf()</i> ... 6, 44	<i>wmemchr()</i> ... 6
15	<i>vfscanf()</i> ... 6, 44	<i>wmemcmp()</i> ... 6
16	<i>vfwprintf()</i> ... 9	<i>wmemcpy()</i> ... 6
17	<i>vfscanf()</i> ... 9	<i>wmemmove()</i> ... 6
18	<i>vprintf()</i> ... 6, 44	<i>wmemset()</i> ... 6
19	<i>vscanf()</i> ... 6, 44	<i>wordexp()</i> ... 8
20	<i>vsnprintf()</i> ... 6	<i>wordfree()</i> ... 8
21	<i>vsprintf()</i> ... 6	<i>wprintf()</i> ... 9
22	<i>vsscanf()</i> ... 6	<i>write()</i> ... xv, xvii, 6, 44
23	<i>vsuprintf()</i> ... 6	<i>writev()</i> ... 9
24	<i>vsuscanf()</i> ... 6	<i>wscanf()</i> ... 9
25	<i>vwprintf()</i> ... 9	<i>y0()</i> ... 9
26	<i>vwscanf()</i> ... 9	<i>y1()</i> ... 9
27	<i>wait()</i> ... 7	<i>yn()</i> ... 9
28	<i>waitid()</i> ... 9	function family
29	<i>waitpid()</i> ... 7	<i>format</i> ... 31
30	<i>wcrtomb()</i> ... 6	functionality
31	<i>wcscat()</i> ... 6	unit of... 29
32	<i>wcschr()</i> ... 6	<i>funlockfile()</i> function... 7, 116
33	<i>wscmp()</i> ... 6	<i>fwide()</i> function... 9
34	<i>wscoll()</i> ... 6	<i>fwprintf()</i> function... 9
35	<i>wscopy()</i> ... 6	<i>fwrite()</i> function... 6, 44
36	<i>wscspn()</i> ... 6	<i>fwscanf()</i> function... 9
37	<i>wcsftime()</i> ... 6	
38	<i>wcslen()</i> ... 6	G
39	<i>wcsncat()</i> ... 6	<i>gai_strerror()</i> function... 7
40	<i>wcsncmp()</i> ... 6	<i>gcvt()</i> function... 120
41	<i>wcsncpy()</i> ... 6	Generic Application Environment Profile... 27
42	<i>wcsprbk()</i> ... 6	generic application environment profile... 27
43	<i>wcsrchr()</i> ... 6	generic environment profile... 33
44	<i>wcsrtombs()</i> ... 6	generic interface profile... 27
45	<i>wcsspn()</i> ... 6	Generic_Read subprogram... 11
46	<i>wcsstr()</i> ... 6	Generic_Write subprogram... 11
47	<i>wcstod()</i> ... 6	Get subprogram... 46
48	<i>wcstof()</i> ... 6	Get_Allowed_Process_Permissions
49	<i>wcstoimax()</i> ... 6	subprogram... 12
	<i>wcstok()</i> ... 6	Get_Buffer subprogram... 14
	<i>wcstol()</i> ... 6	
	<i>wcstold()</i> ... 6	

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	Get_Canonical_Name subprogram... 124,	<i>getcwd()</i> function... 7
2	126	<i>getdate()</i> function... 9
3	Get_Ceiling_Priority subprogram... 124	<i>getegid()</i> function... 9
4	Get_Close_On_Exec subprogram... 12	<i>getenv()</i> function... 8
5	Get_Controlling_Terminal_Name	<i>geteuid()</i> function... 9
6	subprogram... 11	<i>getgid()</i> function... 9
7	Get_Data subprogram... 15	<i>getgrent()</i> function... 10
8	Get_Effective_Group_ID subprogram...	<i>getgrgid()</i> function... 8
9	16	<i>getgrgid_r()</i> function... 8, 116
10	Get_Effective_User_ID subprogram...	<i>getgrnam()</i> function... 8
11	16	<i>getgrnam_r()</i> function... 8, 116
12	Get_Events subprogram... 124	<i>getgroups()</i> function... 9
13	Get_Family subprogram... 124, 126	<i>gethostbyaddr()</i> function... 7
14	Get_File subprogram... 124	<i>gethostbyname()</i> function... 7
15	Get_File_Control subprogram... 12	<i>gethostent()</i> function... 7
16	Get_Flags subprogram... 124, 126	<i>gethostid()</i> function... 9
17	Get_Groups subprogram... 16	<i>gethostname()</i> function... 7
18	Get_Locking_Policy subprogram... 124	<i>getitimer()</i> function... 10
19	Get_Login_Name subprogram... 16	<i>getlogin()</i> function... 9
20	Get_Maximum_Priority subprogram... 14	<i>getlogin_r()</i> function... 9, 116
21	Get_Minimum_Priority subprogram... 14	<i>getmsg()</i> function... 120
22	Get_Notification subprogram... 15	<i>getnameinfo()</i> function... 7
23	Get_Owner subprogram... 14	<i>getnetbyaddr()</i> function... 7
24	Get_Parent_Process_Id subprogram...	<i>getnetbyname()</i> function... 7
25	13	<i>getnetent()</i> function... 7
26	Get_Process_Group_ID subprogram... 16	<i>getopt()</i> function... 8
27	Get_Process_Group_Id subprogram... 13	<i>getpeername()</i> function... 7
28	Get_Process_Id subprogram... 13	<i>getpgid()</i> function... 9
29	Get_Process_Shared subprogram... 124,	<i>getpgrp()</i> function... 7
30	125	<i>getpid()</i> function... 7
31	Get_Protocol_Number subprogram... 124,	<i>getpmsg()</i> function... 120
32	126	<i>getppid()</i> function... 7
33	Get_Real_Group_ID subprogram... 16	<i>getpriority()</i> function... 9
34	Get_Real_User_ID subprogram... 16	<i>getprotobyname()</i> function... 7
35	Get_Returned_Events subprogram... 124	<i>getprotobynumber()</i> function... 7
36	Get_Round_Robin_Interval	<i>getprotoent()</i> function... 7
37	subprogram... 14	<i>getpwent()</i> function... 9
38	Get_Signal subprogram... 15	<i>getpwnam()</i> function... 8
39	Get_Socket_Address_Info	<i>getpwnam_r()</i> function... 8, 116
40	subprogram... 124, 126	<i>getpwuid()</i> function... 8
41	Get_Socket_Type subprogram... 124, 126	<i>getpwuid_r()</i> function... 8, 116
42	Get_Terminal_Characteristics	<i>getrlimit()</i> function... 9
43	subprogram... 11	<i>getrusage()</i> function... 9
44	Get_Terminal_Name subprogram... 11	<i>gets()</i> function... 6, 44
45	Get_Working_Directory subprogram...	<i>getservbyname()</i> function... 7
46	13	<i>getservbyport()</i> function... 7
47	<i>getaddrinfo()</i> function... 7	<i>getservent()</i> function... 7
48	<i>getc()</i> function... 6, 44	<i>getsid()</i> function... 9
49	<i>getc_unlocked()</i> function... 7, 116	<i>getsockname()</i> function... 7
	<i>getchar()</i> function... 6, 44	<i>getsockopt()</i> function... 7
	<i>getchar_unlocked()</i> function... 7, 116	<i>getsubopt()</i> function... 9
	<i>getcontext()</i> function... 9	<i>gettimeofday()</i> function... 9

- 1 *getuid()* function... 9
 2 *getutxent()* function... 10
 3 *getutxid()* function... 10
 4 *getutxline()* function... 10
 5 *getwc()* function... 9
 6 *getwchar()* function... 9
 7 *getwd()* function... 120
 8 *glob()* function... 7
 9 *globfree()* function... 7
 10 *gmtime()* function... 6
 11 *gmtime_r()* function... 6, 116
 12 *grantpt()* function... 9
 13 Group constant... 45
- 14
 15 **H**
- 16 *hcreate()* function... 9
 17 *hdestroy()* function... 9
 18 header
 19 <limits.h>... 36
 20 <unistd.h>... 36, 39, 40, 47, 57, 58, 63,
 21 64, 75, 76, 82, 93, 94, 100, 101
 22 *hsearch()* function... 9
 23 *htonl()* function... 7
 24 *htons()* function... 7
 25 *hypot()* function... 5
 26 *hypotf()* function... 5
 27 *hypotl()* function... 5
- 28
 29 **I**
- 30
 31 *iconv()* function... 9
 32 *iconv_close()* function... 9
 33 *iconv_open()* function... 9
 34 *if_freenameindex()* function... 7
 35 *if_indebyname()* function... 7
 36 *if_nameindex()* function... 7
 37 *if_nametoindex()* function... 7
 38 Ignore_Signal subprogram... 15
 39 *ilogb()* function... 5
 40 *ilogbf()* function... 5
 41 *ilogbl()* function... 5
 42 Image subprogram... 10, 43, 61, 80, 98
 43 *imaxabs()* function... 6
 44 *imaxdiv()* function... 6
 45 Implementation Conformance... 35
 46 implementation defined... 36, 37, 41, 59, 78, 96
 47 terminology... 25
 48 In_Set subprogram... 11, 125
 49 *index()* function... 120
- industry specific interface profile... 27
 industry specific profile... 27
inet_addr() function... 7
inet_ntoa() function... 7
inet_ntop() function... 7
inet_pton() function... 7
initstate() function... 9
 Input_Baud_Rate_Of subprogram... 11
 Input_Time_Of subprogram... 11
insque() function... 9
 Install_Empty_Handler subprogram...
 15
 interface profile... 27
 international standardized profile... 27, 33
 Internet Datagram option... 14
 Internet Protocol option... 14
 Internet Protocol Version 6 option... 89, 108
 Internet Stream option... 14
 Interrupt_Task subprogram... 15
ioctl() function... xv, 120
 Is... 15
 Is_A_Terminal subprogram... 11
 Is_Accessible subprogram... 13
 Is_Block_Special_File subprogram...
 13
 Is_Character_Special_File
 subprogram... 13
 Is_Directory subprogram... 13
 Is_Environment_Variable
 subprogram... 16
 Is_FIFO subprogram... 13
 Is_File subprogram... 13
 Is_File_Present subprogram... 13
 Is_Ignored subprogram... 15
 Is_Member subprogram... 15
 Is_Open subprogram... 11
 Is_Socket subprogram... 13
isalnum() function... 6
isalpha() function... 6
isascii() function... 9
isastream() function... 120
isatty() function... 7
isblank() function... 6
iscntrl() function... 6
isdigit() function... 6
isfinite() function... 5
isgraph() function... 6
isgreater() function... 5
isgreaterequal() function... 5
isinf() function... 5
isless() function... 5
islessequal() function... 5

Copyright © 2003 IEEE. All rights reserved.
 This is an unapproved IEEE Standards Draft, subject to change.

1	<i>islessgreater()</i> function... 5	<i>ldiv()</i> function... 6
2	<i>islower()</i> function... 6	Length subprogram... 16
3	<i>isnan()</i> function... 5	lex utility... 121
4	<i>isnormal()</i> function... 5	<i>lfind()</i> function... 9
5	ISO/IEC Conformant Application... 37	<i>lgamma()</i> function... 5
6	ISP... 27, 33	<i>lgammaf()</i> function... 5
7	<i>isprint()</i> function... 6	<i>lgammal()</i> function... 5
8	<i>ispunct()</i> function... 6	limit
9	<i>isspace()</i> function... 6	_POSIX_NGROUPS_MAX... 96
10	<i>isunordered()</i> function... 5	_POSIX_RTSIG_MAX... 41, 59, 77, 96
11	<i>isupper()</i> function... 6	_POSIX_TIMER_MAX... 41, 59, 77, 96
12	<i>iswalnum()</i> function... 6	CHILD_MAX... 96
13	<i>iswalpha()</i> function... 6	Link subprogram... 13
14	<i>iswblank()</i> function... 6	<i>link()</i> function... xvii, 7
15	<i>iswcntrl()</i> function... 6	<i>lio_listio()</i> function... 113
16	<i>iswctype()</i> function... 6	<i>listen()</i> function... 7
17	<i>iswdigit()</i> function... 6	<i>llabs()</i> function... 6
18	<i>iswgraph()</i> function... 6	<i>lldiv()</i> function... 6
19	<i>iswlower()</i> function... 6	<i>llrint()</i> function... 5
20	<i>iswprint()</i> function... 6	<i>llrintf()</i> function... 5
21	<i>iswpunct()</i> function... 6	<i>llrintl()</i> function... 5
22	<i>iswspace()</i> function... 6	<i>llround()</i> function... 5
23	<i>iswupper()</i> function... 6	<i>llroundf()</i> function... 5
24	<i>iswxdigit()</i> function... 6	<i>llroundl()</i> function... 5
25		<i>localeconv()</i> function... 6
26	J	<i>localtime()</i> function... 6
27		<i>localtime_r()</i> function... 6, 116
28	<i>j0()</i> function... 9	Lock_Shared_Memory subprogram... 125
29	<i>j1()</i> function... 9	<i>lockf()</i> function... 9
30	<i>jn()</i> function... 9	<i>log()</i> function... 5
31	Job Control option... 13	<i>log10()</i> function... 5
32	jobs utility... 122	<i>log10f()</i> function... 5
33	<i>jrnd48()</i> function... 9	<i>log10l()</i> function... 5
34		<i>log1p()</i> function... 5
35	K	<i>log1pf()</i> function... 5
36		<i>log1pl()</i> function... 5
37	<i>kill()</i> function... 8, 44, 62, 80	<i>log2()</i> function... 5
38	<i>killpg()</i> function... 9	<i>log2f()</i> function... 5
39		<i>log2l()</i> function... 5
40	L	<i>logb()</i> function... 5
41		<i>logbf()</i> function... 5
42	<i>l64a()</i> function... 9	<i>logbl()</i> function... 5
43	<i>labs()</i> function... 6	<i>logf()</i> function... 5
44	<i>lchown()</i> function... 9	<i>logl()</i> function... 5
45	<i>lcong48()</i> function... 9	<i>long</i> type... 96
46	<i>ldexp()</i> function... 5	<i>longjmp()</i> function... 4
47	<i>ldexpf()</i> function... 5	<i>lrnd48()</i> function... 9
48	<i>ldexpl()</i> function... 5	<i>lrint()</i> function... 5
49		<i>lrintf()</i> function... 5
		<i>lrintl()</i> function... 5
		<i>lround()</i> function... 5
		<i>lroundf()</i> function... 5

- 1 *lroundl()* function... 5
 2 *lsearch()* function... 9
 3 *lseek()* function... 7
 4 *lstat()* function... 8
 5
 6 **M**
 7
 8 macro
 9 S-IRWXU... 44, 62, 80
 10 *main()* function... 48, 65
 11 make utility... 122
 12 Make_Empty subprogram... 11
 13 *makecontext()* function... 9
 14 *malloc()* function... 6
 15 Map_Memory subprogram... 123
 16 may
 17 terminology... 25
 18 *mblen()* function... 6
 19 *mbrlen()* function... 6
 20 *mbrtowc()* function... 6
 21 *mbsinit()* function... 6
 22 *mbsrtowcs()* function... 6
 23 *mbstowcs()* function... 6
 24 *mbtowc()* function... 6
 25 *memccpy()* function... 9
 26 *memchr()* function... 6
 27 *memcmp()* function... 6
 28 *memcpy()* function... 6
 29 *memmove()* function... 6
 30 Memory Locking option... 20, 42, 60, 79, 97,
 31 123
 32 Memory Mapped option... 125
 33 Memory Mapped Files option... 20, 60, 79,
 34 97, 123, 126
 35 Memory Protection option... 20, 79, 97, 123
 36 Memory Range option... 125
 37 Memory Range Locking option... 20, 42, 60,
 38 79, 97, 123
 39 Memory-Mapped Files option... 69
 40 *memset()* function... 6
 41 msg utility... 122
 42 Message Queues option... 20, 60, 79, 97, 123
 43 Minimal Realtime System Profile... 2, 39
 44 Minimum_Input_Count_Of subprogram...
 45 11
 46 *mkdir()* function... xvii, 7
 47 *mkfifo()* function... 7
 48 *mknod()* function... 9
 49 *mkstemp()* function... 9
 50 *mktemp()* function... 120
 51 *mktime()* function... 6
 52 *mlock()* function... 114
 53 *mlockall()* function... 114
 54 *mmap()* function... 114, 115
 55 MMU... 32
 56 *modf()* function... 5
 57 *modff()* function... 5
 58 *modfl()* function... 5
 59 Monotonic Clock option... 52, 69, 88, 106
 60 more utility... 122
 61 *mprotect()* function... 114
 62 *mq_close()* function... 114
 63 *mq_getattr()* function... 114
 64 *mq_notify()* function... 114
 65 *mq_open()* function... 114
 66 *mq_receive()* function... 114
 67 *mq_send()* function... 114
 68 *mq_setattr()* function... 114
 69 *mq_timedreceive()* function... 114, 118
 70 *mq_timedsend()* function... 114, 118
 71 *mq_unlink()* function... 114
 72 *rand48()* function... 9
 73 *msgctl()* function... 9
 74 *msgget()* function... 9
 75 *msgrcv()* function... 9
 76 *msgsnd()* function... 9
 77 *msync()* function... 114, 116
 78 Multi-Purpose Realtime System Profile... 3, 93
 79 *munlock()* function... 114
 80 *munlockall()* function... 114
 81 *munmap()* function... 114, 115
 82 Mutex Priority Ceiling option... 21, 42, 60,
 83 79, 98, 124
 84 Mutex Priority Inherit option... 124
 85 Mutex Priority Inheritance option... 21, 42,
 86 60, 79, 98, 124
 87 Mutexes option... 124
 88 Mutexes Support option... 42, 60, 79
 89 Mutexes Supported option... 98
 90 MutexPriority Ceiling option... 124
 91
 92 **N**
 93 *nan()* function... 5
 94 *nanf()* function... 5
 95 *nanl()* function... 5
 96 *nanosleep()* function... 118
 97 National Body Conformant POSIX.13
 98 Application... 37
 99 *nearbyint()* function... 5
 100 *nearbyintf()* function... 5
 101 *nearbyintl()* function... 5

1	Network Management option... 124, 126	95
2	<i>newgrp</i> utility... 122	<code>_POSIX_CLOCK_SELECTION...</code> 18, 20,
3	<i>nextafter()</i> function... 5	41, 59, 77, 95
4	<i>nextafterf()</i> function... 5	<code>_POSIX_CPUTIME...</code> 18, 20, 77, 95
5	<i>nextafterl()</i> function... 5	<code>_POSIX_FSYNC...</code> 18, 20, 41, 59, 77, 95,
6	<i>nexttoward()</i> function... 5	120
7	<i>nexttowardf()</i> function... 5	<code>_POSIX_IPV6...</code> 18, 20
8	<i>nexttowardl()</i> function... 5	<code>_POSIX_JOB_CONTROL...</code> 10
9	<i>nftw()</i> function... 9	<code>_POSIX_JOB_CONTROL...</code> 18, 95
10	<i>nice</i> utility... 122	<code>_POSIX_MAPPED_FILES...</code> 18, 20, 49,
11	<i>nice()</i> function... 9	59, 77, 95, 120
12	<i>nl_langinfo()</i> function... 9	<code>_POSIX_MEMLOCK...</code> 18, 20, 41, 48, 59,
13	<i>nm</i> utility... 122	77, 95, 120
14	Normative References... 23	<code>_POSIX_MEMLOCK_RANGE...</code> 18, 20, 41,
15	<i>nrand48()</i> function... 9	59, 77, 95, 120
16	<i>ntohl()</i> function... 7	<code>_POSIX_MEMORY_PROTECTION...</code> 18,
17	<i>ntohs()</i> function... 7	20, 77, 95, 120
18		<code>_POSIX_MESSAGE_PASSING...</code> 18, 20,
19	O	59, 77, 95, 120
20	<i>off_t</i> type... 96	<code>_POSIX_MONOTONIC_CLOCK...</code> 19, 20,
21	Open subprogram... 11, 46	41, 59, 77, 95
22	open system environment... 28, 33	<code>_POSIX_NO_TRUNC...</code> 19, 20, 22, 41, 49,
23	<i>open()</i> function... xv, 6, 44, 49	59, 77, 95
24	Open_Or_Create subprogram... 13	<code>_POSIX_PRIORITIZED_IO...</code> 19, 20,
25	<i>opendir()</i> function... 7	77, 95, 120
26	<i>openlog()</i> function... 9	<code>_POSIX_PRIORITY_SCHEDULING...</code>
27	Operation_Not_Implemented constant... 22, 98	19, 20, 77, 86, 95, 104, 120
28	Operation_Not_Supported constant... 43, 61, 80, 98	<code>_POSIX_RAW_SOCKETS...</code> 19, 20, 77, 95
29	option	<code>_POSIX_READER_WRITER_LOCKS...</code>
30	<code>_POSIX_ADVISORY_INFO...</code> 18, 20, 49,	10
31	66, 84, 95	<code>_POSIX_READER_WRITER_LOCKS...</code>
32	<code>_POSIX_AEP_REALTIME...</code> 39, 57, 75,	19, 20
33	93	<code>_POSIX_REALTIME_SIGNALS...</code> 19, 20,
34	<code>_POSIX_AEP_REALTIME_CONTROLLE</code>	41, 59, 77, 95, 120
35	<code>R...</code> 57	<code>_POSIX_REGEXP...</code> 10
36	<code>_POSIX_AEP_REALTIME_DEDICATED</code>	<code>_POSIX_REGEXP...</code> 19, 21, 95
37	<code>...</code> 75	<code>_POSIX_SAVED_IDS...</code> 19, 21, 95
38	<code>_POSIX_AEP_REALTIME_LANG_Ada9</code>	<code>_POSIX_SEMAPHORES...</code> 19, 21, 41, 59,
39	5... 40, 47, 58, 64, 76, 82, 94, 101	77, 95, 120
40	<code>_POSIX_AEP_REALTIME_LANG_C99...</code>	<code>_POSIX_SHARED_MEMORY_OBJECTS...</code>
41	40, 47, 58, 63, 76, 82, 94, 100	19, 21, 41, 59, 77, 95, 120
42	<code>_POSIX_AEP_REALTIME_MINIMAL...</code>	<code>_POSIX_SHELL...</code> 19, 21, 96
43	39	<code>_POSIX_SPAWN...</code> 19, 21, 77, 96
44	<code>_POSIX_AEP_REALTIME_MULTI...</code> 93	<code>_POSIX_SPIN_LOCKS...</code> 19, 21
45	<code>_POSIX_ASYNCHRONOUS_IO...</code> 18, 20,	<code>_POSIX_SPORADIC_SERVER...</code> 19, 21,
46	77, 95, 120	77, 96
47	<code>_POSIX_BARRIERS...</code> 18, 20	<code>_POSIX_SYNCHRONIZED_IO...</code> 19, 21,
48	<code>_POSIX_CHOWN_RESTRICTED...</code> 18, 20,	41, 59, 77, 96, 120
49		<code>_POSIX_THREAD_ATTR_STACKADDR...</code>
		19, 21, 41, 59, 77, 96
		<code>_POSIX_THREAD_ATTR_STACKSIZE...</code>
		19, 21, 41, 59, 77, 96

1	<code>_POSIX_THREAD_CPUTIME...</code> 19, 21,	<code>_XOPEN_REALTIME...</code> 20, 22
2	41, 59, 77, 96	<code>_XOPEN_REALTIME_THREADS...</code> 20, 22
3	<code>_POSIX_THREAD_PRIO_INHERIT...</code>	<code>_XOPEN_SHM...</code> 20, 22
4	19, 21, 41, 59, 77, 96, 120	<code>_XOPEN_STREAMS...</code> 20, 22, 55, 72, 90,
5	<code>_POSIX_THREAD_PRIO_PROTECT...</code>	109
6	19, 21, 41, 59, 77, 96, 120	<code>_XOPEN_UNIX...</code> 20, 22
7	<code>_POSIX_THREAD_PRIORITY_SCHEDU</code>	Ada Language... 42, 45, 60, 62, 72, 78, 80,
8	<code>LING...</code> 19, 21, 41, 59, 77, 96, 120	97, 99, 100
9	<code>_POSIX_THREAD_PROCESS_SHARED.</code>	Ada language... 91, 109
10	.. 10	Ada-Language... 18, 36
11	<code>_POSIX_THREAD_PROCESS_SHARED...</code>	Advisory Information... 102
12	19, 21, 77, 96	Asynchronous I/O... 20, 79, 97, 123
13	<code>_POSIX_THREAD_SAFE_FUNCTIONS...</code>	C Language... 40, 43, 61, 72, 76, 80, 94, 99
14	19, 21, 96	C language... 58, 90, 109
15	<code>_POSIX_THREAD_SPORADIC_SERVER</code>	Change Owner Restriction... 20, 97, 123
16	<code>...</code> 19, 21, 41, 59, 77, 96	<code>CHILD_MAX...</code> 44, 62
17	<code>_POSIX_THREAD_STACK_ADDRESS...</code>	C-Language... 18, 36
18	121	C-language... 20
19	<code>_POSIX_THREAD_STACK_SIZE...</code> 121	Clock Selection... 52, 69, 88, 106
20	<code>_POSIX_THREADS...</code> 10, 19, 21, 53, 70,	File Locking... 49, 66, 102
21	88, 107	File Synchronization... 49, 66, 84
22	<code>_POSIX_TIMEOUTS...</code> 10	File Synchronization... 20, 42, 60, 79, 97,
23	<code>_POSIX_TIMEOUTS...</code> 19, 21, 41, 59, 77,	123
24	96	Filename Truncation... 42, 60, 79, 97
25	<code>_POSIX_TIMERS...</code> 19, 21, 41, 59, 77, 96,	Filename Truncation... 20, 22, 42, 60, 79,
26	120	97, 123
27	<code>_POSIX_TRACE...</code> 19, 21, 59, 77, 96	Internet Datagram... 14
28	<code>_POSIX_TRACE_EVENT_FILTER...</code> 19,	Internet Protocol... 14
29	21, 59, 77, 96	Internet Protocol Version 6... 89, 108
30	<code>_POSIX_TRACE_INHERIT...</code> 19, 21	Internet Stream... 14
31	<code>_POSIX_TRACE_LOG...</code> 19, 21, 59, 77, 96	Job Control... 13
32	<code>_POSIX_TYPED_MEMORY_OBJECTS...</code>	Memory Locking... 20, 42, 60, 79, 97, 123
33	19, 21	Memory Mapped... 125
34	<code>_POSIX_VDISABLE...</code> 19, 21, 22, 96	Memory Mapped Files... 20, 60, 79, 97,
35	<code>_POSIX_VERSION...</code> 44, 62, 80	123, 126
36	<code>_POSIX2_C_BIND...</code> 19, 21	Memory Protection... 20, 79, 97, 123
37	<code>_POSIX2_C_DEV...</code> 19, 21	Memory Range... 125
38	<code>_POSIX2_CHAR_TERM...</code> 19, 21	Memory Range Locking... 20, 42, 60, 79,
39	<code>_POSIX2_FORT_DEV...</code> 19, 21	97, 123
40	<code>_POSIX2_FORT_RUN...</code> 19, 21	Memory-Mapped Files... 69
41	<code>_POSIX2_LOCALEDEF...</code> 19, 21	Message Queues... 20, 60, 79, 97, 123
42	<code>_POSIX2_PBS...</code> 19, 21	Monotonic Clock... 52, 69, 88, 106
43	<code>_POSIX2_PBS_ACCOUNTING...</code> 19, 21	Mutex Priority Ceiling... 21, 42, 60, 79,
44	<code>_POSIX2_PBS_CHECKPOINT...</code> 20, 21	98, 124
45	<code>_POSIX2_PBS_LOCATE...</code> 20, 21	Mutex Priority Inherit... 124
46	<code>_POSIX2_PBS_MESSAGE...</code> 20, 21	Mutex Priority Inheritance... 21, 42, 60,
47	<code>_POSIX2_PBS_TRACK...</code> 20, 21	79, 98, 124
48	<code>_POSIX2_SW_DEV...</code> 20, 21	Mutexes... 124
49	<code>_POSIX2_UPE...</code> 20, 21	Mutexes Support... 42, 60, 79
	<code>_XOPEN_CRYPT...</code> 20, 21, 55, 72, 90, 109	Mutexes Supported... 98
	<code>_XOPEN_ENH_I18N...</code> 20, 21	MutexPriority Ceiling... 124
	<code>_XOPEN_LEGACY...</code> 20, 22, 55, 72, 90, 109	Network Management... 124, 126

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	Poll... 124	POSIX_Calendar... 15
2	POSIX2_C_BIND... 16, 46, 63, 81, 99, 100	POSIX_Condition_Variables... 124, 125
3	POSIX2_C_DEV... 16, 46, 63, 81, 100	POSIX_Configurable_File_Limits... 12
4	POSIX2_CDEV... 99	POSIX_Configurable_System_Limits... 15
5	POSIX2_CHAR_TERM... 99, 100	POSIX_Event_Management... 11, 124, 125
6	POSIX2_FORT_RUN... 99, 100	POSIX_File_Locking... 12
7	POSIX2_SW_DEV... 16, 46, 47, 63, 64, 81, 82, 99, 100, 101	POSIX_File_Status... 12
8	POSIX2_UPE... 99, 100	POSIX_Files... 12, 13
9	Prioritized I/O... 20, 79, 97, 124	POSIX_Generic_Shared_Memory... 125
10	Priority Process Scheduling... 20, 79, 97, 125	POSIX_Group_Database... 16
11	Process Shared... 50, 67, 85, 104	POSIX_IO... 11, 12, 13, 14, 125, 126
12	Process Shared... 21, 79, 98, 124	POSIX_Limits... 15
13	Process Shared and Mutexes... 125	POSIX_Memory_Mapping... 126
14	Raw Sockets... 89, 108	POSIX_Mutexes... 124, 125
15	Realtime Signals... 20, 42, 60, 79, 97, 125	POSIX_Options... 15, 39, 57, 75, 93
16	Required... 18	POSIX_Page_Alignment... 10
17	Saved IDs Support... 21, 97, 125	POSIX_Permissions... 12
18	Select... 12	POSIX_Process_Environment... 13, 16
19	Select... 125	POSIX_Process_Identification... 13, 16
20	Semaphores... 21, 42, 60, 79, 97, 125	POSIX_Process_Primitives... 13
21	Server Scheduling... 67	POSIX_Process_Scheduling... 14, 125
22	Shared Memory... 123	POSIX_Process_Times... 13
23	Shared Memory Objects... 87	POSIX_Profiles... 15, 16, 43, 61, 79, 98, 111
24	Shared Memory Objects... 21, 42, 60, 79, 98, 125	POSIX_Semaphores... 125
25	Sockets Detailed... 124	POSIX_Shared_Memory_Objects... 125
26	Sockets Detailed Network Interface... 14	POSIX_Signals... 13, 15, 125
27	Sockets Detailed Network Interface... 126	POSIX_Sockets... 14, 124, 126
28	spawn... 65	POSIX_Sockets_Internet... 14
29	Sporadic Server Scheduling... 51, 86, 104	POSIX_Sockets_Local... 14
30	Synchronized I/O... 21, 42, 60, 79, 98, 123, 126	POSIX_Supplement_To_Ada_IO... 10
31	Timeouts... 52, 70, 88, 107	POSIX_Terminal_Functions... 11, 13
32	Timers... 21, 42, 60, 79, 98, 126	POSIX_Timers... 126
33	Trace Event Filtering... 70, 89, 107	POSIX_Unsafe_Process_Primitives... 13, 78
34	Trace Log... 70, 89, 107	POSIX_User_Database... 16
35	XTI Detailed Network Interface... 126	POSIX_XTI... 126
36	OSE... 28, 33	System... 10
37	Other constant... 45	System_Storage_Elements... 10
38	Output_Baud_Rate_Of subprogram... 11	patch utility... 122
39	Owner constant... 45	<i>pathconf()</i> function... 7
40		<i>pause()</i> function... 8
41		
42		
43	P	
44	package	
45	Ada_Streams... 10	
46	Ada_Task_Identification... 10	
47	POSIX... 15	
48		
49		

1	<i>pclose()</i> function... 8	<i>posix_fallocate()</i> function... 113
2	Pending_Signals subprogram... 15	POSIX_FD_MGMT unit of functionality... 7, 12,
3	<i>perror()</i> function... 6, 44	17, 49, 58, 60, 76, 78, 94, 97
4	<i>pipe()</i> function... 7	POSIX_FIFO unit of functionality... 7, 12, 17,
5	platform	49, 66, 84, 94, 97
6	application... 26	POSIX_FILE_ATTRIBUTES unit of
7	development... 27	functionality... 7, 12, 17, 49, 66, 84, 94, 97
8	Poll option... 124	POSIX_File_Locking package... 12
9	Poll subprogram... 124	POSIX_FILE_LOCKING unit of
10	<i>poll()</i> function... 9	functionality... 7, 17, 40, 42, 58, 76, 94
11	<i>popen()</i> function... 8	POSIX_File_Status package... 12
12	POSIX package... 15	POSIX_FILE_SYSTEM unit of functionality... 7,
13	POSIX.1... 32	12, 17, 49, 58, 60, 76, 78, 94, 97
14	POSIX.5c... 32	POSIX_FILE_SYSTEM_EXT unit of
15	POSIX_ADA_LANG_SUPPORT unit of	functionality... 7, 17, 49, 66, 84, 94
16	functionality... 10, 17, 42, 60, 78, 97	POSIX_Files package... 12, 13
17	POSIX_C_LANG_JUMP unit of functionality... 4,	POSIX_Generic_Shared_Memory
18	17, 40, 58, 76, 94	package... 125
19	POSIX_C_LANG_MAT unit of functionality... 58	POSIX_Group_Database package... 16
20	POSIX_C_LANG_MATH unit of functionality... 5,	POSIX_IO package... 11, 12, 13, 14, 125, 126
21	17, 55, 76, 94	POSIX_IO.Generic_Read subprogram... 45
22	POSIX_C_LANG_SUPPOR unit of	POSIX_IO.Generic_Write subprogram... 45
23	functionality... 58	POSIX_IO.Open subprogram... 45
24	POSIX_C_LANG_SUPPORT unit of	POSIX_IO.Open_Or_Create
25	functionality... 6, 17, 40, 76, 94	subprogram... 45
26	POSIX_C_LANG_WIDE_CHAR unit of	POSIX_IO.Read subprogram... 45
27	functionality... 6, 17, 55, 72, 90, 94	POSIX_IO.Write subprogram... 45
28	POSIX_C_LIB_EXT unit of functionality... 40	POSIX_JOB_CONTROL unit of functionality... 7,
29	POSIX_Calendar package... 15	13, 17, 22, 94, 97, 114
30	POSIX_Condition_Variables package... 124, 125	POSIX_Limits package... 15
31	POSIX_Configurable_File_Limits	POSIX_Limits.Child_Processes_Max
32	package... 12	ima'Last constant... 45, 62
33	POSIX_Configurable_System_Limits	POSIX_Limits.Child_Processes_Max
34	package... 15	ima'First type... 98
35	POSIX_Configurable_System_Limits	POSIX_Limits.Groups_Maxima'First
36	.System_POSIX_Ada_Version	constant... 22
37	subprogram... 45, 62, 81	POSIX_Limits.Groups_Maxima'First
38	POSIX_Configurable_System_Limits	constant... 43, 61, 79
39	.System_POSIX_Version	POSIX_Limits.Groups_Maxima'First
40	subprogram... 45, 62, 80	type... 98
41	POSIX_DEVICE_IO unit of functionality... 6,	POSIX_Limits.Realtime_Signals_Ma
42	11, 17, 40, 42, 58, 60, 76, 78, 94, 97	xima'First type... 42, 60, 79, 98
43	POSIX_DEVICE_SPECIFIC unit of	POSIX_Limits.Timers_Maxima'First
44	functionality... 7, 11, 17, 94, 97	type... 42, 60, 79, 98
45	POSIX_Error exception... 22, 43, 61, 80, 98	<i>posix_madvise()</i> function... 113, 114, 115
46	POSIX_Event_Management package... 11,	<i>posix_mem_offset()</i> function... 119
47	124, 125	<i>posix_memalign()</i> function... 113
48	POSIX_EVENT_MGMT unit of functionality... 7,	POSIX_Memory_Mapping package... 126
49	11, 17, 76, 78, 94, 97	POSIX_MULTI_PROCESS unit of
	<i>posix_fadvise()</i> function... 113	functionality... 7, 13, 17, 76, 78, 94, 97

1	POSIX_Mutexes package... 124, 125	POSIX_SIGNALS unit of functionality... 8, 15,
2	POSIX_NETWORKING unit of functionality... 7,	17, 40, 42, 58, 60, 76, 78, 95, 97
3	14, 17, 76, 78, 94, 97	POSIX_Signals.Set_Stopped_Child_
4	<i>posix_openpt()</i> function... 9	Signal subprogram... 22
5	POSIX_Options package... 15, 39, 57, 75, 93	POSIX_Signals.Set_Stopped_Child_
6	POSIX_Page_Alignment package... 10	Signal subprogram... 43, 61, 79
7	POSIX_Permissions package... 12	POSIX_Signals.Stopped_Child_Sign
8	POSIX_PIPE unit of functionality... 7, 17, 76,	al_Enabled subprogram... 22
9	78, 95, 97	POSIX_Signals.Stopped_Child_Sign
10	POSIX_PIPES unit of functionality... 14	al_Enabled subprogram... 43, 61, 79
11	POSIX_PRIORITY_RANGES unit of	POSIX_SINGLE_PROCESS unit of
12	functionality... 7, 14, 17, 40, 51, 58, 67, 86,	functionality... 8, 15, 17, 40, 42, 58, 60, 76,
13	104	78, 95, 97
14	POSIX_Process_Environment package... 13, 16	POSIX_Sockets package... 14, 124, 126
15	POSIX_Process_Identification	POSIX_Sockets_Internet package... 14
16	package... 13, 16	POSIX_Sockets_Local package... 14
17	POSIX_Process_Primitives package... 13	<i>posix_spawn()</i> function... 115
18	Posix_Process_Primitives.Start_P	<i>posix_spawn_file_actions_addclose()</i>
19	rocess subprogram... 81	function... 115
20	Posix_Process_Primitives.Start_P	<i>posix_spawn_file_actions_adddup2()</i>
21	rocess_Search subprogram... 81	function... 115
22	POSIX_Process_Scheduling package... 14, 125	<i>posix_spawn_file_actions_addopen()</i>
23	14, 125	function... 115
24	POSIX_Process_Times package... 13	<i>posix_spawn_file_actions_destroy()</i>
25	POSIX_Profiles package... 15, 16, 43, 61,	function... 115
26	79, 98, 111	<i>posix_spawn_file_actions_init()</i> function... 115
27	POSIX_Profiles.type... 40, 47, 58, 63, 76,	<i>posix_spawnattr_destroy()</i> function... 115
28	82, 94, 100	<i>posix_spawnattr_getflags()</i> function... 115
29	POSIX_Profiles.Realtime_Controller type... 57	<i>posix_spawnattr_getpgroup()</i> function... 115
30	POSIX_Profiles.Realtime_Dedicat	<i>posix_spawnattr_getschedparam()</i>
31	ed type... 75	function... 114, 115
32	POSIX_Profiles.Realtime_Lang_Ada	<i>posix_spawnattr_getschedpolicy()</i>
33	95 type... 40, 47, 58, 64, 76, 82, 94, 101	function... 114, 115
34	POSIX_Profiles.Realtime_Minimal	<i>posix_spawnattr_getsigdefault()</i> function... 115
35	type... 39	<i>posix_spawnattr_getsigmask()</i> function... 115
36	POSIX_Profiles.Realtime_Multi	<i>posix_spawnattr_init()</i> function... 115
37	type... 93	<i>posix_spawnattr_setflags()</i> function... 115
38	POSIX_REGEX unit of functionality... 7, 17,	<i>posix_spawnattr_setpgroup()</i> function... 115
39	54, 71, 89, 95, 108, 115	<i>posix_spawnattr_setschedparam()</i>
40	POSIX_RW_LOCKS unit of functionality... 8,	function... 114, 115
41	10, 17, 115	<i>posix_spawnattr_setschedpolicy()</i>
42	POSIX_Semaphores package... 125	function... 114, 115
43	POSIX_Shared_Memory_Objects	<i>posix_spawnattr_setsigdefault()</i> function... 115
44	package... 125	<i>posix_spawnattr_setsigmask()</i> function... 115
45	POSIX_SHELL_FUNC unit of functionality... 8,	<i>posix_spawnnp()</i> function... 115
46	17, 54, 71, 89, 95, 108	POSIX_STRING_MATCHING unit of
47	POSIX_SIGNAL_JUMP unit of functionality... 8, 17, 76, 95	functionality... 8, 17, 95
48	POSIX_Signals package... 13, 15, 125	
49		

1	POSIX_Supplement_To_Ada_IO	119
2	package... 10	<i>posix_trace_clear()</i> function... 119
3	POSIX_SYMBOLIC_LINKS unit of	<i>posix_trace_close()</i> function... 119
4	functionality... 8, 17, 95	<i>posix_trace_create()</i> function... 119
5	POSIX_SYSTEM_DATABASE unit of	<i>posix_trace_create_withlog()</i> function... 119
6	functionality... 8, 16, 17, 95, 97	<i>posix_trace_event()</i> function... 119
7	POSIX_Terminal_Functions package...	<i>posix_trace_eventid_equal()</i> function... 119
8	11, 13	<i>posix_trace_eventid_get_name()</i> function...
9	POSIX_Terminal_Functions.Disable	119
10	_Control_Character subprogram...	<i>posix_trace_eventid_open()</i> function... 119
11	22	<i>posix_trace_eventset_add()</i> function... 119
12	POSIX_Terminal_Functions.Disable	<i>posix_trace_eventset_del()</i> function... 119
13	_Control_Character subprogram...	<i>posix_trace_eventset_empty()</i> function... 119
14	22, 98	<i>posix_trace_eventset_fill()</i> function... 119
15	POSIX_THREADS_BASE unit of	<i>posix_trace_eventset_ismember()</i> function...
16	functionality... 8, 10, 17, 40, 53, 58, 70, 76,	119
17	88, 95, 107	<i>posix_trace_eventtypelist_getnext_id()</i>
18	POSIX_Timers package... 126	function... 119
19	<i>posix_trace_attr_destroy()</i> function... 119	<i>posix_trace_eventtypelist_rewind()</i>
20	<i>posix_trace_attr_getclockres()</i> function... 119	function... 119
21	<i>posix_trace_attr_getcreatetime()</i> function...	<i>posix_trace_flush()</i> function... 119
22	119	<i>posix_trace_get_attr()</i> function... 119
23	<i>posix_trace_attr_getgenversion()</i> function...	<i>posix_trace_get_filter()</i> function... 119
24	119	<i>posix_trace_get_status()</i> function... 119
25	<i>posix_trace_attr_getinherited()</i> function...	<i>posix_trace_getnext_event()</i> function... 119
26	119	<i>posix_trace_open()</i> function... 119
27	<i>posix_trace_attr_getlogfullpolicy()</i>	<i>posix_trace_rewind()</i> function... 119
28	function... 119	<i>posix_trace_set_filter()</i> function... 119
29	<i>posix_trace_attr_getlogsize()</i> function... 119	<i>posix_trace_shutdown()</i> function... 119
30	<i>posix_trace_attr_getmaxdatasize()</i>	<i>posix_trace_start()</i> function... 119
31	function... 119	<i>posix_trace_stop()</i> function... 119
32	<i>posix_trace_attr_getmaxsystemeventsizes()</i>	<i>posix_trace_timedgetnext_event()</i> function...
33	function... 119	118, 119
34	<i>posix_trace_attr_getmaxusereventsizes()</i>	<i>posix_trace_trid_eventid_open()</i> function...
35	function... 119	119
36	<i>posix_trace_attr_getname()</i> function... 119	<i>posix_trace_trygetnext_event()</i> function...
37	<i>posix_trace_attr_getstreamfullpolicy()</i>	119
38	function... 119	<i>posix_typed_mem_get_info()</i> function... 119
39	<i>posix_trace_attr_getstreamsize()</i> function...	<i>posix_typed_mem_open()</i> function... 119
40	119	POSIX_Unsafe_Process_Primitives
41	<i>posix_trace_attr_init()</i> function... 119	package... 13, 78
42	<i>posix_trace_attr_setinherited()</i> function...	Posix_Unsafe_Process_Primitives
43	119	subprogram... 81
44	<i>posix_trace_attr_setlogfullpolicy()</i>	POSIX_User_Database package... 16
45	function... 119	POSIX_USER_GROUPS unit of functionality...
46	<i>posix_trace_attr_setlogsize()</i> function... 119	9, 16, 17, 95, 97
47	<i>posix_trace_attr_setmaxdatasize()</i>	POSIX_WIDE_CHAR_IO unit of
48	function... 119	functionality... 9, 17, 95
49	<i>posix_trace_attr_setname()</i> function... 119	POSIX_XTI package... 126
	<i>posix_trace_attr_setstreamfullpolicy()</i>	POSIX2_C_BIND option... 16, 46, 63, 81, 99,
	function... 119	100
	<i>posix_trace_attr_setstreamsize()</i> function...	POSIX2_C_DEV option... 16, 46, 63, 81, 100

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1	POSIX2_CDEV option... 99	116, 118
2	POSIX2_CHAR_TERM option... 99, 100	<i>pthread_attr_getschedparam()</i> function... 8,
3	POSIX2_FORT_RUN option... 99, 100	117
4	POSIX2_SW_DEV option... 16, 46, 47, 63, 64,	<i>pthread_attr_getschedpolicy()</i> function...
5	81, 82, 99, 100, 101	116, 118
6	POSIX2_UPE option... 99, 100	<i>pthread_attr_getscope()</i> function... 116, 118
7	<i>pow()</i> function... 5	<i>pthread_attr_getstack()</i> function... 10, 116,
8	<i>powf()</i> function... 5	117
9	<i>powl()</i> function... 5	<i>pthread_attr_getstackaddr()</i> function... 116,
10	<i>pread()</i> function... 9	117
11	<i>printf()</i> function... 6, 44	<i>pthread_attr_getstacksize()</i> function... 116,
12	Prioritized I/O option... 20, 79, 97, 124	117, 121
13	priority ceiling protocol... 50, 67, 85, 104	<i>pthread_attr_init()</i> function... 8, 117
14	priority inheritance protocol... 50, 67, 85, 104	<i>pthread_attr_setdetachstate()</i> function... 8,
15	Priority Inversion... 28	117
16	priority inversion... 50, 67, 85, 104	<i>pthread_attr_setguardsize()</i> function... 10
17	Priority Process Scheduling option... 20, 79,	<i>pthread_attr_setinheritsched()</i> function...
18	97, 125	116, 118
19	priority protection protocol	<i>pthread_attr_setschedparam()</i> function... 8,
20	see priority ceiling protocol	117
21	Process Shared and Mutexes option... 125	<i>pthread_attr_setschedpolicy()</i> function...
22	Process Shared option... 50, 67, 85, 104	116, 118
23	Process Shared option... 21, 79, 98, 124	<i>pthread_attr_setscope()</i> function... 116, 118
24	profile	<i>pthread_attr_setstack()</i> function... 10, 116,
25	application environment... 26, 27	117
26	component... 26	<i>pthread_attr_setstackaddr()</i> function... 116,
27	for ISO standardization... 28	117
28	generic application environment... 27	<i>pthread_attr_setstacksize()</i> function... 116,
29	generic interface... 27	117, 121
30	industry specific interface... 27	<i>pthread_barrier_destroy()</i> function... 113,
31	interface... 27	117
32	international standardized... 27, 33	<i>pthread_barrier_init()</i> function... 113, 117
33	realtime environment... 28	<i>pthread_barrier_wait()</i> function... 113, 117
34	system... 28	<i>pthread_barrierattr_destroy()</i> function...
35	profile, application environment... 33	113, 117
36	profile, generic environment... 33	<i>pthread_barrierattr_getpshared()</i>
37	protocol	function... 113, 116, 117
38	priority ceiling... 50, 67, 85, 104	<i>pthread_barrierattr_init()</i> function... 113,
39	priority inheritance... 50, 67, 85, 104	117
40	ps utility... 122	<i>pthread_barrierattr_setpshared()</i> function...
41	PSE... 33	113, 116, 117
42	PSE51... 2, 33, 39	<i>pthread_cancel()</i> function... 8, 117
43	PSE52... 3, 33, 57	<i>pthread_cleanup_pop()</i> function... 8, 117
44	PSE53... 3, 33, 75	<i>pthread_cleanup_push()</i> function... 8, 117
45	PSE54... 3, 33, 93	<i>pthread_cond_broadcast()</i> function... 8, 117
46	<i>pselect()</i> function... 7	<i>pthread_cond_destroy()</i> function... 8, 117
47	<i>pthread_atfork()</i> function... 8, 117	<i>pthread_cond_init()</i> function... 8, 117
48	<i>pthread_attr_destroy()</i> function... 8, 117	<i>pthread_cond_signal()</i> function... 8, 117
49	<i>pthread_attr_getdetachstate()</i> function... 8,	<i>pthread_cond_timedwait()</i> function... 8, 117
	117	<i>pthread_cond_wait()</i> function... 8, 117
	<i>pthread_attr_getguardsize()</i> function... 10	<i>pthread_condattr_destroy()</i> function... 8, 117
	<i>pthread_attr_getinheritsched()</i> function...	<i>pthread_condattr_getclock()</i> function... 113,

1	117	<i>pthread_rwlock_timedrdlock()</i> function... 8,
2	<i>pthread_condattr_getpshared()</i> function...	118
3	116, 118	<i>pthread_rwlock_timedwrlock()</i> function... 8,
4	<i>pthread_condattr_init()</i> function... 8, 117	118
5	<i>pthread_condattr_setclock()</i> function... 113,	<i>pthread_rwlock_tryrdlock()</i> function... 8, 118
6	117	<i>pthread_rwlock_trywrlock()</i> function... 8,
7	<i>pthread_condattr_setpshared()</i> function...	118
8	116, 118	<i>pthread_rwlock_unlock()</i> function... 8, 118
9	<i>pthread_create()</i> function... 8, 117	<i>pthread_rwlock_wrlock()</i> function... 8, 118
10	<i>pthread_detach()</i> function... 8, 117	<i>pthread_rwlockattr_destroy()</i> function... 8,
11	<i>pthread_equal()</i> function... 8, 117	118
12	<i>pthread_exit()</i> function... 8, 117	<i>pthread_rwlockattr_getpshared()</i> function...
13	<i>pthread_getconcurrency()</i> function... 10	8, 116, 118
14	<i>pthread_getcpuclockid()</i> function... 116, 117	<i>pthread_rwlockattr_init()</i> function... 8, 118
15	<i>pthread_getschedparam()</i> function... 116,	<i>pthread_rwlockattr_setpshared()</i> function...
16	118	8, 116, 118
17	<i>pthread_getspecific()</i> function... 8, 117	PTHREAD_SCOPE_PROCESS constant... 78,
18	<i>pthread_join()</i> function... 8, 117	96
19	<i>pthread_key_create()</i> function... 8, 117	PTHREAD_SCOPE_SYSTEM constant... 78, 86,
20	<i>pthread_key_delete()</i> function... 8, 117	96, 105
21	<i>pthread_kill()</i> function... 8, 117	<i>pthread_self()</i> function... 8, 117
22	<i>pthread_mutex_destroy()</i> function... 8, 117	<i>pthread_setcancelstate()</i> function... 8, 117
23	<i>pthread_mutex_getprioceiling()</i> function...	<i>pthread_setcanceltype()</i> function... 8, 117
24	116, 118	<i>pthread_setconcurrency()</i> function... 10
25	<i>pthread_mutex_init()</i> function... 8, 117	<i>pthread_setschedparam()</i> function... 116,
26	<i>pthread_mutex_lock()</i> function... 8, 117	118
27	<i>pthread_mutex_setprioceiling()</i> function...	<i>pthread_setschedprio()</i> function... 116, 118
28	116, 118	<i>pthread_setspecific()</i> function... 8, 117
29	<i>pthread_mutex_timedlock()</i> function... 118	<i>pthread_sigmask()</i> function... 8, 117
30	<i>pthread_mutex_trylock()</i> function... 8, 117	<i>pthread_spin_destroy()</i> function... 115, 117
31	<i>pthread_mutex_unlock()</i> function... 8, 117	<i>pthread_spin_init()</i> function... 115, 117
32	<i>pthread_mutexattr_destroy()</i> function... 8,	<i>pthread_spin_lock()</i> function... 115, 117
33	117	<i>pthread_spin_trylock()</i> function... 115, 117
34	<i>pthread_mutexattr_getprioceiling()</i>	<i>pthread_spin_unlock()</i> function... 115, 117
35	function... 116, 118	<i>pthread_testcancel()</i> function... 8, 117
36	<i>pthread_mutexattr_getprotocol()</i> function...	<i>ptsname()</i> function... 9
37	116, 117	Put subprogram... 46
38	<i>pthread_mutexattr_getpshared()</i> function...	<i>putc()</i> function... 6, 44
39	116, 118	<i>putc_unlocked()</i> function... 7, 116
40	<i>pthread_mutexattr_gettype()</i> function... 10	<i> putchar()</i> function... 6, 44
41	<i>pthread_mutexattr_init()</i> function... 8, 117	<i> putchar_unlocked()</i> function... 7, 116
42	<i>pthread_mutexattr_setprioceiling()</i>	<i> putenv()</i> function... 9
43	function... 116, 118	<i> putmsg()</i> function... 120
44	<i>pthread_mutexattr_setprotocol()</i> function...	<i> putpmsg()</i> function... 120
45	116, 117	<i> puts()</i> function... 6, 44
46	<i>pthread_mutexattr_setpshared()</i> function...	<i> pututxline()</i> function... 10
47	116, 118	<i> putwc()</i> function... 9
48	<i>pthread_mutexattr_settype()</i> function... 10	<i> putwchar()</i> function... 9
49	<i>pthread_once()</i> function... 8, 117	<i> pwrite()</i> function... 9
	<i>pthread_rwlock_destroy()</i> function... 8, 118	
	<i>pthread_rwlock_init()</i> function... 8, 118	
	<i>pthread_rwlock_rdlock()</i> function... 8, 118	

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 1 **Q**
- 2
- 3 *qalter* utility... 122
- 4 *qdel* utility... 122
- 5 *qhold* utility... 122
- 6 *qmove* utility... 122
- 7 *qmsg* utility... 122
- 8 *qrerun* utility... 122
- 9 *qrls* utility... 122
- 10 *qselect* utility... 122
- 11 *qsig* utility... 122
- 12 *qsort()* function... 6
- 13 *qstat* utility... 122
- 14 *qsub* utility... 122
- 15 *Queue_Signal* subprogram... 125
- 16
- 17 **R**
- 18
- 19 *raise()* function... 8, 44, 62
- 20 *rand()* function... 6
- 21 *rand_r()* function... 6, 116
- 22 *random()* function... 9
- 23 Raw Sockets option... 89, 108
- 24 *Read* subprogram... 11, 46
- 25 *read()* function... xv, xvii, 6, 44
- 26 *Read_Write_Execute* constant... 45
- 27 *readdir()* function... 7
- 28 *readdir_r()* function... 7, 116
- 29 *readlink()* function... 8
- 30 *readv()* function... 9
- 31 *realloc()* function... 6
- 32 *realpath()* function... 9
- 33 Realtime Controllor System Profile... 3, 57
- 34 realtime environment profile... 28
- 35 Realtime Signals option... 20, 42, 60, 79, 97, 125
- 36 Realtime System Profiles... 2
- 37 *Realtime_Lang_C99* type... 40, 47, 58, 63, 76, 82, 94, 100
- 38 *recv()* function... 7
- 39 *recvfrom()* function... 7
- 40 *recvmsg()* function... 7
- 41 *regcomp()* function... 7
- 42 *regerror()* function... 7
- 43 *regexec()* function... 7
- 44 *regfree()* function... 7
- 45 Related Open Systems Standards... 127
- 46 *remainder()* function... 5
- 47 *remainderf()* function... 5
- 48 *remainderl()* function... 5
- 49 *Remove* subprogram... 11, 125
- remove()* function... 7
- Remove_Directory* subprogram... 13
- remque()* function... 9
- remquo()* function... 5
- remquof()* function... 5
- remquol()* function... 5
- Rename* subprogram... 13
- rename()* function... xvii, 7
- renice* utility... 122
- Required option... 18
- rewind()* function... 7
- rewinddir()* function... 7
- rindex()* function... 120
- rint()* function... 5
- rintf()* function... 5
- rintl()* function... 5
- rmdir()* function... xvii, 7
- round()* function... 5
- roundf()* function... 5
- roundl()* function... 5
- 20 **S**
- Saved IDs Support option... 21, 97, 125
- scalb()* function... 9
- scalbln()* function... 5
- scalblnf()* function... 5
- scalblnl()* function... 5
- scalbn()* function... 5
- scalbnf()* function... 5
- scalbnl()* function... 5
- scanf()* function... 6, 44
- SCHED_FIFO* constant... 41, 51, 59, 68, 78, 86, 96, 104, 105
- sched_get_priority_max()* function... 7, 114
- sched_get_priority_min()* function... 7, 114
- sched_getparam()* function... 114
- sched_getscheduler()* function... 114
- SCHED_RR* constant... 41, 51, 59, 68, 78, 86, 96, 104, 105
- sched_rr_get_interval()* function... 7, 114
- sched_setparam()* function... 114
- sched_setscheduler()* function... 114
- sched_yield()* function... 114, 118
- seed48()* function... 9
- Seek* subprogram... 12
- seekdir()* function... 9
- Select option... 12
- Select option... 125
- select()* function... 7, 54, 71, 89, 108
- Select_File* subprogram... 11, 125

1	<i>sem_close()</i> function... 115	<i>Set_Terminal_Characteristics</i>
2	<i>sem_destroy()</i> function... 115	subprogram... 11
3	<i>sem_getvalue()</i> function... 115	<i>Set_User_ID</i> subprogram... 16
4	<i>sem_init()</i> function... 115	<i>setbuf()</i> function... 6
5	<i>sem_open()</i> function... 115	<i>setcontext()</i> function... 9
6	<i>sem_post()</i> function... 44, 62, 115	<i>setgid()</i> function... 9
7	<i>sem_timedwait()</i> function... 115, 118	<i>setenv()</i> function... 8
8	<i>sem_trywait()</i> function... 115	<i>seteuid()</i> function... 9
9	<i>sem_unlink()</i> function... 115	<i>setgid()</i> function... 9
10	<i>sem_wait()</i> function... 115	<i>setgrent()</i> function... 10
11	Semaphores option... 21, 42, 60, 79, 97, 125	<i>sethostent()</i> function... 7
12	<i>semctl()</i> function... 9	<i>setitimer()</i> function... 10
13	<i>semget()</i> function... 9	<i>setjmp()</i> function... 4
14	<i>semop()</i> function... 9	<i>setkey()</i> function... 120
15	<i>send()</i> function... 7	<i>setlocale()</i> function... 6
16	Send_Break subprogram... 11	<i>setlogmask()</i> function... 9
17	Send_Signal subprogram... 15	<i>setnetent()</i> function... 7
18	<i>sendmsg()</i> function... 7	<i>setpgid()</i> function... 7
19	<i>sendto()</i> function... 7	<i>setpgrp()</i> function... 9
20	Server Scheduling option... 67	<i>setpriority()</i> function... 9
21	Set_Allowed_Process_Permissions	<i>setprotoent()</i> function... 7
22	subprogram... 12	<i>setpwent()</i> function... 9
23	Set_Blocked_Signals subprogram... 15	<i>setregid()</i> function... 10
24	Set_Buffer subprogram... 14	<i>setreuid()</i> function... 10
25	Set_Ceiling_Priority subprogram... 124	<i>setrlimit()</i> function... 9
26	Set_Close_On_Exec subprogram... 12	<i>setservent()</i> function... 7
27	Set_Data subprogram... 15	<i>setsid()</i> function... 7
28	Set_Environment_Variable	<i>setsockopt()</i> function... 7
29	subprogram... 16	<i>setstate()</i> function... 9
30	Set_Events subprogram... 124	<i>setuid()</i> function... 9
31	Set_Family subprogram... 124, 126	<i>setutxent()</i> function... 10
32	Set_File subprogram... 124	<i>setvbuf()</i> function... 6
33	Set_File_Control subprogram... 12	sh utility... 121
34	Set_File_Times subprogram... 13	shall
35	Set_Flags subprogram... 124, 126	terminology... 25
36	Set_Group_ID subprogram... 16	Shared Memory option... 123
37	Set_Locking_Policy subprogram... 124	Shared Memory Objects option... 87
38	Set_Notification subprogram... 15	Shared Memory Objects option... 21, 42, 60,
39	Set_Process_Group_Id subprogram... 13	79, 98, 125
40	Set_Process_Shared subprogram... 124,	<i>shm_open()</i> function... 115
41	125	<i>shm_unlink()</i> function... 115
42	Set_Protocol_Number subprogram... 124,	<i>shmat()</i> function... 9
43	126	<i>shmctl()</i> function... 9
44	Set_Returned_Events subprogram... 124	<i>shmdt()</i> function... 9
45	Set_Signal subprogram... 15	<i>shmget()</i> function... 9
46	Set_Socket_Group_Owner subprogram...	should
47	14	terminology... 25
48	Set_Socket_Process_Owner	<i>shutdown()</i> function... 7
49	subprogram... 14	<i>sigaction()</i> function... 8, 44, 62
	Set_Socket_Type subprogram... 124, 126	<i>sigaddset()</i> function... 8, 44, 62
	Set_Stopped_Child_Signal	<i>sigaltstack()</i> function... 9
	subprogram... 13	<i>sigdelset</i> function... 44

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1	<i>sigdelset()</i> function... 8, 62	<i>srandom()</i> function... 9
2	<i>sigemptyset()</i> function... 8, 44, 62	<i>sscanf()</i> function... 6
3	<i>sigfillset()</i> function... 8, 44, 62	standard
4	<i>sighold()</i> function... 9	base... 26
5	<i>sigignore()</i> function... 9	<i>stat()</i> function... 7
6	<i>siginterrupt()</i> function... 9	<i>statvfs()</i> function... 9
7	<i>sigismember()</i> function... 8, 44, 62	Stopped_Child_Signal_Enabled
8	<i>siglongjmp()</i> function... 8	subprogram... 13
9	Signal type... 15	<i>strcasecmp()</i> function... 9
10	<i>signal()</i> function... 8, 44, 62	<i>strcat()</i> function... 6
11	Signal_Event type... 15	<i>strchr()</i> function... 6
12	Signal_Info type... 126	<i>strcmp()</i> function... 6
13	<i>signbit()</i> function... 5	<i>strcoll()</i> function... 6
14	<i>sigpause()</i> function... 9	<i>strcpy()</i> function... 6
15	<i>sigpending()</i> function... 8, 44, 62	<i>strcspn()</i> function... 6
16	<i>sigprocmask()</i> function... 8, 44, 62	<i>strdup()</i> function... 9
17	<i>sigqueue()</i> function... 44, 62, 115	<i>strerror()</i> function... 6
18	<i>sigrelse()</i> function... 9	<i>strerror_r()</i> function... 6, 116
19	<i>sigset</i> type... 9	<i>strfmon()</i> function... 9
20	<i>sigset()</i> function... 44, 62	<i>strftime()</i> function... 6
21	<i>sigsetjmp()</i> function... 8	Strictly Conforming Application... 37
22	<i>sigsuspend()</i> function... 8	strings utility... 122
23	<i>sigtimedwait()</i> function... 115	strip utility... 122
24	<i>sigwait()</i> function... 8	<i>strlen()</i> function... 6
25	<i>sigwaitinfo()</i> function... 115	<i>strncasecmp()</i> function... 9
26	<i>sin()</i> function... 5	<i>strncat()</i> function... 6
27	<i>sinf()</i> function... 5	<i>strncmp()</i> function... 6
28	<i>sinh()</i> function... 5	<i>strncpy()</i> function... 6
29	<i>sinhf()</i> function... 5	<i>strpbrk()</i> function... 6
30	<i>sinhl()</i> function... 5	<i>strptime()</i> function... 9
31	<i>sinl()</i> function... 5	<i>strrchr()</i> function... 6
32	S-IRWXU macro... 44, 62, 80	<i>strspn()</i> function... 6
33	<i>sleep()</i> function... 7, 52, 69, 88, 106	<i>strstr()</i> function... 6
34	<i>snprintf()</i> function... 6	<i>strtod()</i> function... 6
35	<i>socketmark()</i> function... 7	<i>strtof()</i> function... 6
36	<i>socket()</i> function... 7	<i>strtoimax()</i> function... 6
37	<i>socketpair()</i> function... 7	<i>strtok()</i> function... 6
38	Sockets Detailed option... 124	<i>strtok_r()</i> function... 6, 116
39	Sockets Detailed Network Interface option... 14	<i>strtol()</i> function... 6
40	Sockets Detailed Network Interface	<i>strtold()</i> function... 6
41	option... 126	<i>strtoll()</i> function... 6
42	spawn option... 65	<i>strtoul()</i> function... 6
43	Special_Control_Character_Of	<i>strtoull()</i> function... 6
44	subprogram... 11	<i>strtoumax()</i> function... 6
45	split utility... 122	<i>strxfrm()</i> function... 6
46	Sporadic Server Scheduling option... 51, 86, 104	Subprofiling Option Group... 28
47	<i>sprintf()</i> function... 6	subprogram
48	<i>sqrt()</i> function... 5	Accessibility... 13
49	<i>sqrtf()</i> function... 5	Add... 11, 125
	<i>sqrtl()</i> function... 5	Add_All_Signals... 15
	<i>srand()</i> function... 6	Add_Signal... 15
	<i>srand48()</i> function... 9	Argument_List... 16

1	Await_Signal... 15, 125	For_Every_Environment_Variabl
2	Await_Signal_Or_Timeout... 15,	e... 16
3	125	For_Every_File_In... 11
4	Bits_Per_Character_Of... 11	For_Every_Item... 124, 126
5	Block_Signals... 15	Generic_Read... 11
6	Blocked_Signals... 15	Generic_Write... 11
7	Change_Owner_And_Group... 12	Get... 46
8	Change_Permissions... 12, 123	Get_Allowed_Process_Permissio
9	Change_Protection... 123	ns... 12
10	Change_Working_Directory... 13	Get_Buffer... 14
11	Clear_Environment... 16	Get_Canonical_Name... 124, 126
12	Close... 11	Get_Ceiling_Priority... 124
13	Copy_Environment... 16	Get_Close_On_Exec... 12
14	Copy_From_Current_Environment	Get_Controlling_Terminal_Name
15	... 16	... 11
16	Copy_To_Current_Environment...	Get_Data... 15
17	16	Get_Effective_Group_ID... 16
18	Create... 46	Get_Effective_User_ID... 16
19	Create_Directory... 13	Get_Events... 124
20	Create_FIFO... 12	Get_Family... 124, 126
21	Create_Pipe... 14	Get_File... 124
22	Create_Process_Group... 13	Get_File_Control... 12
23	Create_Session... 16	Get_Flags... 124, 126
24	Define_Bits_Per_Character... 11	Get_Groups... 16
25	Define_Input_Baud_Rate... 11	Get_Locking_Policy... 124
26	Define_Input_Time... 11	Get_Login_Name... 16
27	Define_Minimum_Input_Count... 11	Get_Maximum_Priority... 14
28	Define_Output_Baud_Rate... 11	Get_Minimum_Priority... 14
29	Define_Special_Control_Charac	Get_Notification... 15
30	ter... 11	Get_Owner... 14
31	Define_Terminal_Modes... 11	Get_Parent_Process_Id... 13
32	Delete... 46	Get_Process_Group_ID... 16
33	Delete_All_Signals... 15	Get_Process_Group_Id... 13
34	Delete_Environment_Variable...	Get_Process_Id... 13
35	16	Get_Process_Shared... 124, 125
36	Delete_Signal... 15	Get_Protocol_Number... 124, 126
37	Disable_Control_Character... 11	Get_Real_Group_ID... 16
38	Disable_Queueing... 125	Get_Real_User_ID... 16
39	Discard_Data... 11	Get_Returned_Events... 124
40	Drain... 11	Get_Round_Robin_Interval... 14
41	Duplicate... 12	Get_Signal... 15
42	Duplicate_And_Close... 12	Get_Socket_Address_Info... 124,
43	Enable_Queueing... 125	126
44	Environment_Value_Of... 16	Get_Socket_Type... 124, 126
45	Existence... 13	Get_Terminal_Characteristics...
46	File_Position... 12	11
47	File_Size... 12	Get_Terminal_Name... 11
48	Filename_Of... 13	Get_Working_Directory... 13
49	Flow... 11	Ignore_Signal... 15
	For_Every_Current_Environment	Image... 10, 43, 61, 80, 98
	_Variable... 16	In_Set... 11, 125
	For_Every_Directory_Entry... 13	Input_Baud_Rate_Of... 11

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	Input_Time_Of... 11	es... 81
2	Install_Empty_Handler... 15	Put... 46
3	Interrupt_Task... 15	Queue_Signal... 125
4	Is_A_Terminal... 11	Read... 11, 46
5	Is_Accessible... 13	Remove... 11, 125
6	Is_Block_Special_File... 13	Remove_Directory... 13
7	Is_Character_Special_File... 13	Rename... 13
8	Is_Directory... 13	Seek... 12
9	Is_Environment_Variable... 16	Select_File... 11, 125
10	Is_FIFO... 13	Send_Break... 11
11	Is_File... 13	Send_Signal... 15
12	Is_File_Present... 13	Set_Allowed_Process_Permissio ns... 12
13	Is_Ignored... 15	Set_Blocked_Signals... 15
14	Is_Member... 15	Set_Buffer... 14
15	Is_Open... 11	Set_Ceiling_Priority... 124
16	Is_Socket... 13	Set_Close_On_Exec... 12
17	Length... 16	Set_Data... 15
18	Link... 13	Set_Environment_Variable... 16
19	Lock_Shared_Memory... 125	Set_Events... 124
20	Make_Empty... 11	Set_Family... 124, 126
21	Map_Memory... 123	Set_File... 124
22	Minimum_Input_Count_Of... 11	Set_File_Control... 12
23	Open... 11, 46	Set_File_Times... 13
24	Open_Or_Create... 13	Set_Flags... 124, 126
25	Output_Baud_Rate_Of... 11	Set_Group_ID... 16
26	Pending_Signals... 15	Set_Locking_Policy... 124
27	Poll... 124	Set_Notification... 15
28	POSIX_Configurable_System_Lim its.System_POSIX_Ada_Ve rsion... 45, 62, 81	Set_Process_Group_Id... 13
29	POSIX_Configurable_System_Lim its.System_POSIX_Versio n... 45, 62, 80	Set_Process_Shared... 124, 125
30	POSIX_IO.Generic_Read... 45	Set_Protocol_Number... 124, 126
31	POSIX_IO.Generic_Write... 45	Set_Returned_Events... 124
32	POSIX_IO.Open... 45	Set_Signal... 15
33	POSIX_IO.Open_Or_Create... 45	Set_Socket_Group_Owner... 14
34	POSIX_IO.Read... 45	Set_Socket_Process_Owner... 14
35	POSIX_IO.Write... 45	Set_Socket_Type... 124, 126
36	Posix_Process_Primitives.Star t_Process... 81	Set_Stopped_Child_Signal... 13
37	Posix_Process_Primitives.Star t_Process_Search... 81	Set_Terminal_Characteristics... 11
38	POSIX_Signals.Set_Stopped_Chi ld_Signal... 22, 43, 61, 79	Set_User_ID... 16
39	POSIX_Signals.Stopped_Child_S ignal_Enabled... 22, 43, 61, 79	Special_Control_Character_Of... 11
40	POSIX_Terminal_Functions.Disa ble_Control_Character... 22, 98	Stopped_Child_Signal_Enabled... 13
41	Posix_Unsafe_Process_Primitiv	Synchronize_Data... 123, 126
42		Synchronize_File... 123
43		Synchronize_Memory... 123, 126
44		Terminal_Modes_Of... 11
45		Truncate_File... 123, 125
46		Unblock_Signals... 15
47		Unignore_Signal... 15
48		Unlink... 13
49		

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1	Unlock_Shared_Memory... 125	<i>tgammal()</i> function... 5
2	Unmap_Memory... 123	time utility... 122
3	Value... 10, 43, 61, 80, 98	<i>time()</i> function... 6, 44, 62
4	Write... 11, 46	Timeouts option... 52, 70, 88, 107
5	Summary of Profile Features... 17	<i>timer_create()</i> function... 118
6	<i>swab()</i> function... 9	<i>timer_delete()</i> function... 118
7	<i>swapcontext()</i> function... 9	<i>timer_getoverrun()</i> function... 44, 62, 118
8	<i>swprintf()</i> function... 6	<i>timer_gettime()</i> function... 44, 62, 118
9	<i>swscanf()</i> function... 6	<i>timer_settime()</i> function... 44, 62, 118
10	<i>symlink()</i> function... 8	Timers option... 21, 42, 60, 79, 98, 126
11	<i>sync()</i> function... 9	<i>times()</i> function... 7, 44, 62
12	Synchronize_Data subprogram... 123, 126	<i>timezone()</i> function... 9
13	Synchronize_File subprogram... 123	<i>tmpfile()</i> function... 7
14	Synchronize_Memory subprogram... 123, 126	<i>tmpnam()</i> function... 7
15	Synchronized I/O option... 21, 42, 60, 79, 98, 123, 126	<i>toascii()</i> function... 9
16	<i>sysconf()</i> function... 8, 44, 62, 80	<i>tolower()</i> function... 6
17	<i>syslog()</i> function... 9	<i>toupper()</i> function... 6
18	System package... 10	<i>towctrans()</i> function... 6
19	system profile... 28	<i>towlower()</i> function... 6
20	<i>system()</i> function... 8	<i>toupper()</i> function... 6
21	System_Storage_Elements package... 10	tput utility... 122
22		Trace Event Filtering option... 70, 89, 107
23		Trace Log option... 70, 89, 107
24	T	True . . True constant... 39, 40, 42, 47, 57, 58, 60, 63, 64, 75, 76, 79, 82, 93, 94, 97, 100, 101, 111
25		<i>trunc()</i> function... 5
26	tabs utility... 122	<i>truncate()</i> function... 9
27	talk utility... 122	Truncate_File subprogram... 123, 125
28	<i>tan()</i> function... 5	<i>truncf()</i> function... 5
29	<i>tanf()</i> function... 5	<i>truncl()</i> function... 5
30	<i>tanh()</i> function... 5	<i>tsearch()</i> function... 9
31	<i>tanhf()</i> function... 5	<i>ttyname()</i> function... 7
32	<i>tanhf_l()</i> function... 5	<i>ttyname_r()</i> function... 7, 116
33	<i>tanl()</i> function... 5	<i>twalk()</i> function... 9
34	<i>tcdrain()</i> function... 7	type
35	<i>tcflow()</i> function... 7	Boolean... 39, 40, 57, 58, 75, 76, 93, 94, 111
36	<i>tcflush()</i> function... 7	FILE *... 49, 66, 84, 102
37	<i>tcgetattr()</i> function... 7	long... 96
38	<i>tcgetpgrp()</i> function... 7	off_t... 96
39	<i>tcgetsid()</i> function... 9	POSIX_Limits.Child_Processes_Maxima'First... 98
40	<i>tcsendbreak()</i> function... 7	POSIX_Limits.Groups_Maxima'First... 98
41	<i>tcsetattr()</i> function... 7	POSIX_Limits.Realtime_Signals_Maxima'First... 42, 60, 79, 98
42	<i>tcsetpgrp()</i> function... 7	POSIX_Limits.Timers_Maxima'First... 42, 60, 79, 98
43	<i>tdelete()</i> function... 9	POSIX_Profiles . . . 40, 47, 58, 63, 76, 82, 94, 100
44	<i>telldir()</i> function... 9	
45	<i>tempnam()</i> function... 9	
46	Terminal_Modes_Of subprogram... 11	
47	Terminology... 25	
48	<i>tfind()</i> function... 9	
49	<i>tgamma()</i> function... 5	
	<i>tgammaf()</i> function... 5	

Copyright © 2003 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1	POSIX_Profiles.Realtime_Contr	94, 97
2	oller... 57	POSIX_EVENT_MGMT... 7, 11, 17, 76, 78,
3	POSIX_Profiles.Realtime_Dedic	94, 97
4	ated... 75	POSIX_FD_MGMT... 7, 12, 17, 49, 58, 60,
5	POSIX_Profiles.Realtime_Lang_	76, 78, 94, 97
6	Ada95... 40, 47, 58, 64, 76, 82, 94,	POSIX_FIFO... 7, 12, 17, 49, 66, 84, 94, 97
7	101	POSIX_FILE_ATTRIBUTES... 7, 12, 17,
8	POSIX_Profiles.Realtime_Minim	49, 66, 84, 94, 97
9	al... 39	POSIX_FILE_LOCKING... 7, 17, 40, 42,
10	POSIX_Profiles.Realtime_Multi	58, 76, 94
11	... 93	POSIX_FILE_SYSTEM... 7, 12, 17, 49, 58,
12	Realtime_Lang_C99... 40, 47, 58, 63,	60, 76, 78, 94, 97
13	76, 82, 94, 100	POSIX_FILE_SYSTEM_EXT... 7, 17, 49,
14	Signal... 15	66, 84, 94
15	Signal_Event... 15	POSIX_JOB_CONTROL... 7, 13, 17, 22, 94,
16	Signal_Info... 126	97, 114
17	<i>sigset</i> ... 9	POSIX_MULTI_PROCESS... 7, 13, 17, 76,
18	<i>tzname</i> () function... 6	78, 94, 97
19	<i>tzset</i> () function... 6	POSIX_NETWORKING... 7, 14, 17, 76, 78,
20		94, 97
21	U	POSIX_PIPE... 7, 17, 76, 78, 95, 97
22	<i>ualarm</i> () function... 9	POSIX_PIPES... 14
23	<i>ulimit</i> () function... 9	POSIX_PRIORITY_RANGES... 7, 14, 17,
24	<i>umask</i> () function... 7	40, 51, 58, 67, 86, 104
25	unalias utility... 122	POSIX_REGEX... 7, 17, 54, 71, 89, 95,
26	<i>uname</i> () function... 8, 44, 62	108, 115
27	Unblock_Signals subprogram... 15	POSIX_RW_LOCKS... 8, 10, 17, 115
28	Unbounded Priority Inversion... 28	POSIX_SHELL_FUNC... 8, 17, 54, 71, 89,
29	undefined... 36	95, 108
30	terminology... 26	POSIX_SIGNAL_JUMP... 8, 17, 76, 95
31	unexpand utility... 122	POSIX_SIGNALS... 8, 15, 17, 40, 42, 58,
32	<i>ungetc</i> () function... 6	60, 76, 78, 95, 97
33	<i>ungetwc</i> () function... 9	POSIX_SINGLE_PROCESS... 8, 15, 17,
34	Unignore_Signal subprogram... 15	40, 42, 58, 60, 76, 78, 95, 97
35	unit of functionality... 29	POSIX_STRING_MATCHING... 8, 17, 95
36	POSIX_ADA_LANG_SUPPORT... 10, 17,	POSIX_SYMBOLIC_LINKS... 8, 17, 95
37	42, 60, 78, 97	POSIX_SYSTEM_DATABASE... 8, 16, 17,
38	POSIX_C_LANG_JUMP... 4, 17, 40, 58, 76,	95, 97
39	94	POSIX_THREADS_BASE... 8, 10, 17, 40,
40	POSIX_C_LANG_MAT... 58	53, 58, 70, 76, 88, 95, 107
41	POSIX_C_LANG_MATH... 5, 17, 55, 76, 94	POSIX_USER_GROUPS... 9, 16, 17, 95, 97
42	POSIX_C_LANG_SUPPOR... 58	POSIX_WIDE_CHAR_IO... 9, 17, 95
43	POSIX_C_LANG_SUPPORT... 6, 17, 40,	XSI_C_LANG_SUPPORT... 9, 17, 54, 71,
44	76, 94	90, 108, 121
45	POSIX_C_LANG_WIDE_CHAR... 6, 17,	XSI_DBM... 9, 17, 54, 71, 90, 108, 121
46	55, 72, 90, 94	XSI_DEVICE_IO... 9, 18, 54, 71, 90, 108,
47	POSIX_C_LIB_EXT... 40	121
48	POSIX_DEVICE_IO... 6, 11, 17, 40, 42,	XSI_DEVICE_SPECIFIC... 9, 18, 54, 71,
49	58, 60, 76, 78, 94, 97	90, 108, 121
	POSIX_DEVICE_SPECIFIC... 7, 11, 17,	XSI_DYNAMIC_LINKING... 9, 18, 54, 71,
		90, 95, 108, 121
		XSI_FD_MGMT... 9, 18, 54, 71, 90, 108,

1	121	du... 122
2	XSI_FILE_SYSTEM... 9, 18, 54, 71, 90,	ex... 122
3	108, 121	expand... 122
4	XSI_I18N... 9, 18, 54, 71, 90, 108, 121	fc... 122
5	XSI_IPC... 9, 18, 54, 71, 90, 108, 120, 121	fg... 122
6	XSI_JOB_CONTROL... 9, 18, 54, 71, 90,	file... 122
7	108, 121	fort77... 122
8	XSI_JUMP... 9, 18, 54, 71, 90, 108, 121	fsck... xvii
9	XSI_MATH... 9, 18, 54, 71, 90, 108, 121	jobs... 122
10	XSI_MULTI_PROCESS... 9, 18, 54, 71, 90,	lex... 121
11	108, 121	make... 122
12	XSI_SIGNALS... 9, 18, 54, 71, 90, 108,	mesg... 122
13	121	more... 122
14	XSI_SINGLE_PROCESS... 9, 18, 54, 71,	newgrp... 122
15	90, 108, 121	nice... 122
16	XSI_SYSTEM_DATABASE... 9, 18, 54, 71,	nm... 122
17	90, 108, 121	patch... 122
18	XSI_SYSTEM_LOGGING... 9, 18, 54, 71,	ps... 122
19	90, 95, 109, 121	qalter... 122
20	XSI_THREAD_MUTEX_EXT... 10, 18, 40,	qdel... 122
21	54, 58, 72, 76, 90, 95, 109, 121	qhold... 122
22	XSI_THREADS_EXT... 10, 18, 40, 54, 58,	qmove... 122
23	72, 77, 90, 95, 109, 121	qmsg... 122
24	XSI_TIMERS... 10, 18, 54, 71, 90, 108, 121	qrerun... 122
25	XSI_USER_GROUPS... 10, 18, 54, 71, 90,	qrls... 122
26	108, 121	qselect... 122
27	XSI_WIDE_CHAR... 10, 18, 54, 71, 90,	qsig... 122
28	108, 121	qstat... 122
29	Units of Functionality... 4	qsub... 122
30	Unlink subprogram... 13	renice... 122
31	<i>unlink()</i> function... xvii, 7	sh... 121
32	Unlock_Shared_Memory subprogram... 125	split... 122
33	<i>unlockpt()</i> function... 9	strings... 122
34	Unmap_Memory subprogram... 123	strip... 122
35	<i>unsetenv()</i> function... 8	tabs... 122
36	unspecified... 36, 37	talk... 122
37	terminology... 26	time... 122
38	Use_Error exception... 45, 63, 81	tput... 122
39	<i>usleep()</i> function... 9	unalias... 122
40	utility	unexpand... 122
41	alias... 122	uudecode... 122
42	ar... 122	uencode... 122
43	asa... 122	vi... 122
44	at... 122	who... 122
45	batch... 122	write... 122
46	bg... 122	yacc... 121
47	c99... 121	<i>utime()</i> function... 7
48	command... 122	<i>utimes()</i> function... 120
49	crontab... 122	uudecode utility... 122
	csplit... 122	uencode utility... 122
	ctags... 122	
	df... 122	

1 **V**

2
 3 *va_arg()* function... 6
 4 *va_copy()* function... 6
 5 *va_end()* function... 6
 6 *va_start()* function... 6
 7 Value subprogram... 10, 43, 61, 80, 98
 8 *vfork()* function... 9
 9 *vfprintf()* function... 6, 44
 10 *vfscanf()* function... 6, 44
 11 *vfuprntf()* function... 9
 12 *vfuscanf()* function... 9
 13 vi utility... 122
 14 *vprintf()* function... 6, 44
 15 *vscanf()* function... 6, 44
 16 *vsprintf()* function... 6
 17 *vsscanf()* function... 6
 18 *vsuprintf()* function... 6
 19 *vsuscanf()* function... 6
 20 *vwprintf()* function... 9
 21 *wscanf()* function... 9

22
 23 **W**

24
 25 *wait()* function... 7
 26 *waitid()* function... 9
 27 *waitpid()* function... 7
 28 *wcrtomb()* function... 6
 29 *wcscat()* function... 6
 30 *wcschr()* function... 6
 31 *wcscmp()* function... 6
 32 *wcscoll()* function... 6
 33 *wcscpy()* function... 6
 34 *wcscspn()* function... 6
 35 *wcsftime()* function... 6
 36 *wcslen()* function... 6
 37 *wcsncat()* function... 6
 38 *wcsncmp()* function... 6
 39 *wcsncpy()* function... 6
 40 *wcspbrk()* function... 6
 41 *wcsrchr()* function... 6
 42 *wcsrtombs()* function... 6
 43 *wcsspn()* function... 6
 44 *wcsstr()* function... 6
 45 *wcstod()* function... 6
 46 *wcstof()* function... 6
 47 *wcstoimax()* function... 6
 48 *wcstok()* function... 6
 49 *wcstol()* function... 6
 50 *wcstold()* function... 6

wcstoll() function... 6
wcstombs() function... 6
wcstoul() function... 6
wcstoull() function... 6
wcstoumax() function... 6
weswces() function... 120
weswidth() function... 10
wesxfrm() function... 6
wctob() function... 6
wctomb() function... 6
wctrans() function... 6
wctype() function... 6
wcwidth() function... 10
 who utility... 122
wmemchr() function... 6
wmemcmp() function... 6
wmemcpy() function... 6
wmemmove() function... 6
wmemset() function... 6
wordexp() function... 8
wordfree() function... 8
wprintf() function... 9
 Write subprogram... 11, 46
 write utility... 122
write() function... xv, xvii, 6, 44
writev() function... 9
wscanf() function... 9

X

XSI_C_LANG_SUPPORT unit of
 functionality... 9, 17, 54, 71, 90, 108, 121
 XSI_DBM unit of functionality... 9, 17, 54, 71,
 90, 108, 121
 XSI_DEVICE_IO unit of functionality... 9, 18,
 54, 71, 90, 108, 121
 XSI_DEVICE_SPECIFIC unit of
 functionality... 9, 18, 54, 71, 90, 108, 121
 XSI_DYNAMIC_LINKING unit of
 functionality... 9, 18, 54, 71, 90, 95, 108, 121
 XSI_FD_MGMT unit of functionality... 9, 18, 54,
 71, 90, 108, 121
 XSI_FILE_SYSTEM unit of functionality... 9,
 18, 54, 71, 90, 108, 121
 XSI_I18N unit of functionality... 9, 18, 54, 71,
 90, 108, 121
 XSI_IPC unit of functionality... 9, 18, 54, 71,
 90, 108, 120, 121
 XSI_JOB_CONTROL unit of functionality... 9,
 18, 54, 71, 90, 108, 121
 XSI_JUMP unit of functionality... 9, 18, 54, 71,

1 90, 108, 121
2 XSI_MATH unit of functionality... 9, 18, 54, 71,
3 90, 108, 121
4 XSI_MULTI_PROCESS unit of functionality...
5 9, 18, 54, 71, 90, 108, 121
6 XSI_SIGNALS unit of functionality... 9, 18, 54,
7 71, 90, 108, 121
8 XSI_SINGLE_PROCESS unit of
9 functionality... 9, 18, 54, 71, 90, 108, 121
10 XSI_SYSTEM_DATABASE unit of
11 functionality... 9, 18, 54, 71, 90, 108, 121
12 XSI_SYSTEM_LOGGING unit of
13 functionality... 9, 18, 54, 71, 90, 95, 109, 121
14 XSI_THREAD_MUTEX_EXT unit of
15 functionality... 10, 18, 40, 54, 58, 72, 76, 90,
16 95, 109, 121
17 XSI_THREADS_EXT unit of functionality... 10,
18 18, 40, 54, 58, 72, 77, 90, 95, 109, 121
19 XSI_TIMERS unit of functionality... 10, 18, 54,
20 71, 90, 108, 121
21 XSI_USER_GROUPS unit of functionality... 10,
22 18, 54, 71, 90, 108, 121
23 XSI_WIDE_CHAR unit of functionality... 10,
24 18, 54, 71, 90, 108, 121
25 XTI Detailed Network Interface option...
26 126

27 **Y**

28
29 *y0()* function... 9
30 *yI()* function... 9
31 *yacc* utility... 121
32 *yn()* function... 9
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49