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
Editor's Notes

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

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

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

Please report typographical errors to:

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

The copying and distribution of IEEE balloting drafts is accomplished by the Standards Office. To report problems with reproduction of your copy, or to request additional copies of this draft, contact:

Tracy Woods
IEEE Computer Society,
1730 Massachusetts Avenue, NW,
Washington DC 20036-1992, USA.
Phone: +1-202-371-1013
Fax: +1-202-728-0884
E-mail: twoods@computer.org
Web page: http://www.computer.org/standard/draftstd.htm

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
  - 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
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)
6.4 Shell and Utility Requirements ...................................................... 46
6.5 Development Platform Requirements .......................................... 46
  6.5.1 C Language Development Option ........................................... 46
  6.5.2 Ada Language Development Option ......................................... 47
6.6 Rationale for Operating System Requirements (informative) .............. 47
  6.6.1 Operating System Interface Requirements ............................... 47
  6.6.2 Shell and Utility Requirements ............................................. 55
  6.6.3 Development Platform Requirements ....................................... 55

Section 7: Realtime Controller System Profile (PSE52) .......................... 57
  7.1 Introduction ............................................................................. 57
    7.1.1 Identification ................................................................. 57
    7.1.2 Conformance ................................................................. 57
    7.1.3 Options .......................................................................... 58
  7.2 Operating System Interface Requirements .................................... 58
    7.2.1 POSIX.1 Requirements (C language Option) .......................... 58
    7.2.2 POSIX.5c Requirements (Ada Language Option) .................... 60
  7.3 Application Constraints .......................................................... 61
    7.3.1 Constraints related to POSIX.1 Interfaces (C Language Option) . 61
    7.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option) ................................................................. 62
  7.4 Shell and Utility Requirements ................................................... 63
  7.5 Development Platform Requirements .......................................... 63
    7.5.1 C Language Development Option ........................................... 63
    7.5.2 Ada Language Development Option ......................................... 64
  7.6 Rationale for Operating System Requirements (informative) .............. 64
    7.6.1 Operating System Interface Requirements ............................... 64
    7.6.2 Shell and Utility Requirements ............................................. 72
    7.6.3 Development Platform Requirements ....................................... 73

Section 8: Dedicated Realtime System Profile (PSE53) .............................. 75
  8.1 Introduction ............................................................................. 75
    8.1.1 Identification ................................................................. 75
    8.1.2 Conformance ................................................................. 75
    8.1.3 Options .......................................................................... 76
  8.2 Operating System Interface Requirements .................................... 76
    8.2.1 POSIX.1 Requirements (C language Option) .......................... 76
    8.2.2 POSIX.5c Requirements (Ada Language Option) .................... 78
  8.3 Application Constraints .......................................................... 80
    8.3.1 Constraints related to POSIX.1 Interfaces (C Language Option) . 80
    8.3.2 Constraints related to POSIX.5c Interfaces (Ada Language Option) ................................................................. 80
  8.4 Shell and Utility Requirements ................................................... 81
  8.5 Development Platform Requirements .......................................... 81
    8.5.1 C Language Development Option ........................................... 81
    8.5.2 Ada Language Development Option ......................................... 82
  8.6 Rationale for Operating System Requirements (informative) .............. 82
    8.6.1 Operating System Interface Requirements ............................... 82

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

Figure I.1. Main Building Blocks of the Profiles .............................. xix
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.
Table 9-4: POSIX.5c Option Requirements ........................................... 97
Table 9-5: Shell and Utilities Option Requirements
   (C Language Option) ............................................................. 99
Table 9-6: Shell and Utilities Option Requirements
   (Ada Language Option) .........................................................100
Table B-1: Functions under each POSIX.1
   System Interface Option .....................................................113
Table B-2: Utilities under each POSIX.1 Shell and Utilities Option ..........121
Table B-3: Packages and Subprograms under
   each POSIX.5c Option. ..........................................................123
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:


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

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

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

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

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

Editor’s note: 200x will be changed to match the year the 1003.13 revision is approved as a standard.

Organization of This Standard

This Standard is divided into eight elements:

(1) General (Section 1)
(2) Normative references (Section 2)
(3) Definitions (Section 3)
(4) Conventions and abbreviations (Section 4)
(5) Conformance (Section 5)
(6) The various realtime profiles (Sections 6 through 9)
(7) ISPICS requirements (C) (Annex A)
(8) ISPICS requirements (Ada) (Annex B)

References are provided to direct the reader to other related sections.
Informative annexes are not normative parts of the standard and are provided for information only. They are provided for guidance and to help understanding.

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

**Base Documents**

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

**Scenario**

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

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

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

**Audience**

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

1. UNIX is a registered trademark of The Open Group in the United States of America and other countries.
(1) Persons buying hardware and software systems.

(2) Persons managing companies that are deciding on future corporate computing directions.

(3) Persons implementing realtime operating systems.

(4) Persons developing realtime applications where portability is a primary objective.

Rationale on Background

This subclause contains rationale common to all four realtime profiles.

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

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

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

Features of several commercial realtime kernels were considered during the development of the 1998 version of POSIX.13. These included pSOSTM1, VRTX32TM2, and VxWorkSTM3. Since these products were commercially successful, they must have addressed a significant market segment. In addition, the uniprocessor subset of VITA's ORKID specification, NGCR's "Tiny Real Time" (TRT), and the uITRON specification were examined. These were all proposed standard interfaces for small realtime embedded systems.

Features of other commercial realtime kernels such as RT-Linux4 and QNX5, as well as free software products such as RTEMS6 were considered during the development of the current revision of POSIX.13.

The following is a list of features that are representative of current realtime systems and highlights the range of system requirements. While some concepts are

1. pSOS is now a registered trademark of Wind River Systems, Inc.
2. VRTX32 is now a registered trademark of Mentor Graphics.
3. VxWorks is a registered trademark of Wind River Systems, Inc.
4. RT-Linux is... FSM Labs
5. QNX is...
6. RTEMS is...
common to virtually all implementations (e.g., preemptive, priority-based scheduling), some only apply to smaller systems (e.g., a single address space), and some only to more full-featured systems (e.g., network support, self-hosting).

**Basic Realtime Multitasking and Synchronization**

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

**I/O**

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

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

Local File System

Realtime systems supporting a file system generally provide the following features:
— Named files
— Hierarchical filesystem (directories)
— Contiguous preallocation of disk space
— May provide media compatibility with another filesystem (e.g. MSDOS™, or RT-11™)
— No user IDs or file protection

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

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

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

1. MS-DOS is a registered trademark of Microsoft Corporation.
2. RT-11 is now a registered trademark of Compaq.
3. RSX-11M is now a registered trademark of Compaq.
Traditional implementations of POSIX.1 filesystems employ a disk buffer cache to improve average performance by reducing the number of physical media accesses, and by reordering the accesses to take advantage of the characteristics of rotating media. These implementations have not made a distinction between the buffering of data transfers [read() and write()], and directory operations [creat(), link(), unlink(), mkdir(), rmdir(), rename()]. A result of this is that a system crash at an unexpected moment can leave the filesystem in a corrupted state. This situation is usually corrected at the next system reboot by a filesystem checker and recovery program, such as fsck. The checking and correcting of a corrupted filesystem may take a long and variable amount of time to perform, may require a human operator to monitor and control its progress, and may nonetheless fail to repair the filesystem. Any one of these characteristics would make a filesystem check unacceptable for some embedded realtime applications. It was therefore suggested that such applications limit their use of directory operations to safe times, and that implementations maintain the filesystem in such a way that a filesystem check during reboot is avoided. This was considered, but rejected on the grounds that not all applications would require the capability, and that it was neither specifiable nor testable.

Network Communication

Realtime systems supporting networking generally provide the following features:

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

Distributed File System

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

- Remote access to a filesystem
- Performance not realtime

Memory Protection

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

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

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

— **network**
  Non-transparent access to remote objects, remote procedure calls

— **distributed**
  Transparent access to objects, no load-balancing

— **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).
Related Standards Activities

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

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

1. 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/).

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
If you have interest in participating in the Portable Application Standards Committee (PASC) working groups addressing these issues, please send your name, address, and phone number to

Secretary, IEEE Standards Board
Institute of Electrical and Electronics Engineers, Inc.
P.O. Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331
USA

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

If you have interest in participating in this work at the international level, contact your International Organization for Standardization/International Electrotechnical Committee (ISO/IEC) national body.
IEEE Std 1003.13-1998 was prepared by the System Services Working Group—Realtime, sponsored by the Portable Application Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the System Services Working Group—Realtime was as follows:

**Portable Application Standards Committee**

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

**IEEE System Services Working Group—Realtime**

Chair: Joseph M. Gwinn, Susan Corwin (to 1995)  
Technical Editor: Bob Luken  
Ballot Coordinators: Andrew E. Wheeler, Jr., James T. Oblinger  
Technical Reviewers: Andrew E. Wheeler, Jr., Joseph M. Gwinn, Karen D. Gordon

**Working Group**

Ray Alderman, Michael Feustel, Dave Luenger  
Larry Anderson, Bill Gallmeister, Bill Maes  
Pierre-Jean Arcos, Michael González, James T. Oblinger  
Charles R. Arnold, Karen D. Gordon, Offer Pazy  
V. Raj Avula, Randy Greene, Carolyn Petersen  
Theodore P. Baker, Rick Greer, Dave Plauger  
Todd Bargorek, Joseph M. Gwinn, Arlan Pool  
Robert Barned, Steven A. Haaser, Franklin C. Prindle  
Richard M. Bergman, Barbara Haleen, François Riche  
Nawaf Bitar, Geoffrey R. Hall, Robert Rose
Ballot Group

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

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

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

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Satish K. Aggarwal | James H. Gurney | L. Bruce McClung
---|---|---
Clyde R. Camp | Jim D. Issak | Louis-François Pau
James T. Carlo | Lowell G. Johnson | Ronald C. Petersen
Gary R. Engmann | Robert Kennelly | Gerald H. Peterson
Harold E. Epstein | E.G. “Al” Kiener | John B. Posey
Jay Forster\(^a\) | Joseph L. Koepfinger\(^a\) | Gary S. Robinson
Thomas F. Garrity | Stephen R. Lambert | Hans E. Weinrich
Ruben D. Garzon | Jim Logothetis | Donald W. Zispe

\(^a\) Member emeritus

Noelle Humenick

*IEEE Standards Project Editor*
IEEE P1003.13 was prepared by the System Services Working Group—Realtime, sponsored by the Portable Application Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the System Services Working Group—Realtime was as follows:

**Portable Application Standards Committee**

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

**IEEE System Services Working Group—Realtime**

- Chair: Joseph M. Gwinn
- Secretary: Karen D. Gordon
- Technical Editor: Michael González
- Ballot Coordinator: Jim Oblinger
- Technical Reviewers: Michael González

**Working Group**

<to be added later>

**Ballot Group**

<to be added later>
Section 1: Overview

1.1 Scope


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

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

1.3.2 Realtime Controller System Profile (PSE52)

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

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

1.3.3 Dedicated Realtime System Profile (PSE53)

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

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

1.3.4 Multi-Purpose Realtime System Profile (PSE54)

These systems include all the functionality of the other three profiles. They provide comprehensive functionality and run a mix of differing realtime and non-realtime tasks. This functionality includes most of POSIX.1 and/or POSIX.5c. Since users may conduct interactive sessions on those systems, all the mandatory elements of the Shell and Utilities volume of POSIX.1 are also included. Support for multiple multi-threaded processes is required so that multi-tasking may be done by threads (POSIX.1 threads or Ada tasks), processes, or both.
The hardware model for this profile assumes one or more processors with memory management units, high-speed storage devices, special interfaces, network support, and display devices. The system supports a mix of realtime and non-realtime tasks, some being interactive user tasks.

### 1.4 Units of Functionality

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

Table 1-1 shows the Units of Functionality defined for POSIX.1; each of these units represents a Subprofiling Option Group (See the Base Definitions Volume of POSIX.1 [3], Section 2.1.5.1, “Subprofiling Considerations”), and is a set of functions that represents a separately implementable element of POSIX.1. Table 1-2 through Table 1-18 show the Units of Functionality defined for POSIX.5c.

#### Table 1-1: POSIX.1 Units of Functionality

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>Included Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_C_LANG_JUMP</td>
<td>longjmp(), setjmp()</td>
</tr>
<tr>
<td>Unit of Functionality</td>
<td>Included Functions</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>POSIX_C_LANG_MATH</td>
<td>acos(), acosf(), acosh(), acoshf(), acosl(), asin(), asinf(), asinh(), asinhf(), asinl(), atan(), atan2(), atan2f(), atan2l(), atanh(), atanhf(), atanhl(), cabs(), cabsf(), cabsl(), cacos(), cacosf(), cacosh(), cacoshf(), cacoshl(), carg(), cargf(), cargl(), casin(), casinf(), casinh(), casinhf(), casinhl(), catan(), catanh(), catanhf(), catanhl(), cbrt(), cbrtf(), cbrtl(), cos(), cosf(), cosl(), cosh(), coshf(), coshl(), cosl(), ceil(), ceilf(), ceil(), cexp(), cexpf(), cexpl(), cimag(), cimagf(), cimagl(), clog(), clogf(), clogl(), conj(), conjf(), conjl(), copysign(), copysignf(), copysignl(), cos(), cosf(), cosl(), cosl(), cpow(), cpowf(), cpowl(), cproj(), cprojf(), cprojl(), creal(), crealf(), creall(), csin(), csinf(), csinh(), csinhf(), csinhl(), csqrt(), csqrtf(), csqrtl(), ctan(), ctanf(), ctanh(), ctanhf(), ctanhl(), erf(), erff(), erfl(), erf(), erfl(), exp(), exp2(), exp2f(), exp2l(), expf(), expl(), expm1(), expm1f(), expm1l(), fabs(), fabsf(), fabsl(), fdim(), fdimf(), floor(), floorf(), floorl(), fma(), fmaf(), fmall(), fmax(), fmaxf(), fmaxl(), fmin(), fminf(), fminl(), fmod(), fmodf(), fmodl(), fclassify(), frexp(), frexpf(), frexpl(), hypot(), hypotf(), hypotl(), ilogb(), ilogbf(), ilogbl(), isnan(), isnanf(), isnanl(), isunordered(), ldexp(), ldexpf(), ldexpl(), lgamma(), lgammaf(), lgammal(), llrint(), llrintf(), llrintl(), lround(), lroundf(), lroundl(), log(), log10(), log10f(), log10l(), log1p(), log1pf(), log1pl(), log2(), log2f(), log2l(), logbf(), logbl(), logf(), logl(), lrint(), lrintf(), lrintl(), lround(), lroundf(), lroundl(), modf(), modff(), modfl(), nan(), nanf(), nanl(), nearbyint(), nearbyintf(), nearbyintl(), nextafter(), nextafterf(), nextafterl(), nexttoward(), nexttowardf(), nexttowardl(), pow(), powf(), powl(), remainder(), remainderf(), remainderl(), remquo(), remquof(), remquol(), rint(), rintf(), rintl(), round(), roundf(), roundl(), scalbln(), scalblnf(), scalblnl(), scalbn(), scalbnf(), scalbnl(), signbit(), sin(), sinf(), sinh(), sinhf(), sinhl(), sqrt(), sqrtf(), sqrtl(), tan(), tanf(), tanhl(), tanhf(), tgamma(), tgammaf(), tgamma(), trunc(), truncf(), truncl()</td>
</tr>
</tbody>
</table>
### Table 1-1: POSIX.1 Units of Functionality (Continued)

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>Included Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>POSIX_C_LANG_SUPPORT</strong></td>
<td><code>abs()</code>, <code>asctime()</code>, <code>asctime_r()</code>, <code>atof()</code>, <code>atoi()</code>, <code>atol()</code>, <code>atoll()</code>, <code>bsearch()</code>, <code>calloc()</code>, <code>ctime()</code>, <code>ctime_r()</code>, <code>difftime()</code>, <code>div()</code>, <code>fecl Except()</code>, <code>feof()</code>, <code>ftime()</code>, <code>ftimeExcept()</code>, <code>fgetenv()</code>, <code>fgetexceptflag()</code>, <code>fgetround()</code>, <code>fholdexcept()</code>, <code>fouaiseexcept()</code>, <code>fesetenv()</code>, <code>fesetexceptflag()</code>, <code>fesetround()</code>, <code>fetestexcept()</code>, <code>freeenv()</code>, <code>gtime()</code>, <code>gtime_r()</code>, <code>imaxabs()</code>, <code>imaxdiv()</code>, <code>isalnum()</code>, <code>isalpha()</code>, <code>isblank()</code>, <code>iscntrl()</code>, <code>isdigit()</code>, <code>isgraph()</code>, <code>islower()</code>, <code>isprint()</code>, <code>ispunct()</code>, <code>isspace()</code>, <code>isupper()</code>, <code>isxdigit()</code>, <code>labs()</code>, <code>ldiv()</code>, <code>labs()</code>, <code>ldiv()</code>, <code>localeconv()</code>, <code>localtime()</code>, <code>localtime_r()</code>, <code>malloc()</code>, <code>memchr()</code>, <code>memcmp()</code>, <code>memcpy()</code>, <code>memmove()</code>, <code>memset()</code>, <code>mktime()</code>, <code>qsort()</code>, <code>rand()</code>, <code>rand_r()</code>, <code>realloc()</code>, <code>setlocale()</code>, <code>snprintf()</code>, <code>sprint()</code>, <code>srand()</code>, <code>sscanf()</code>, <code>strcat()</code>, <code>strchr()</code>, <code>strncpy()</code>, <code>strcoll()</code>, <code>strcpy()</code>, <code>strcspn()</code>, <code>strerror()</code>, <code>strftime()</code>, <code>strlen()</code>, <code>strncat()</code>, <code>strncpy()</code>, <code>strpbrk()</code>, <code>strrchr()</code>, <code>strspn()</code>, <code>strstr()</code>, <code>strtok()</code>, <code>strtok_r()</code>, <code>strtol()</code>, <code>strtoimax()</code>, <code>strtotoll()</code>, <code>strtoall()</code>, <code>strtoull()</code>, <code>strtoumax()</code>, <code>strtoul()</code>, <code>strtoumax()</code>, <code>time()</code>, <code>tolower()</code>, <code>toupper()</code>, <code>tzname()</code>, <code>tzset()</code>, <code>va_arg()</code>, <code>va_copy()</code>, <code>va_end()</code>, <code>va_start()</code>, <code>vsnprintf()</code>, <code>vprintf()</code>, <code>vsprintf()</code>, <code>wscanf()</code></td>
</tr>
<tr>
<td><strong>POSIX_C_LANG_WIDE_CHAR</strong></td>
<td><code>btoe()</code>, <code>isalnum()</code>, <code>isalpha()</code>, <code>isblank()</code>, <code>iswctype()</code>, <code>iswdigit()</code>, <code>iswgraph()</code>, <code>iswlower()</code>, <code>iswprint()</code>, <code>iswucode()</code>, <code>iswspace()</code>, <code>iswupper()</code>, <code>iswxdigit()</code>, <code>mblen()</code>, <code>mbrlen()</code>, <code>mbtowc()</code>, <code>mbstowcs()</code>, <code>mbstowcs()</code>, <code>mbtowlc()</code>, <code>mbtowc()</code>, <code>mbsinit()</code>, <code>mbsrtowcs()</code>, <code>mbswchar()</code>, <code>mbsproxy()</code>, <code>mbsproxy()</code>, <code>mbsproxy()</code></td>
</tr>
<tr>
<td></td>
<td><code>sprintf()</code>, <code>sscanf()</code>, <code>swscanf()</code>, <code>towctrans()</code>, <code>towlower()</code>, <code>toupper()</code>, <code>usprintf()</code>, <code>vsprintf()</code>, <code>uswscanf()</code>, <code>vtowmb()</code>, <code>wcset()</code>, <code>weschr()</code>, <code>wescmp()</code>, <code>wesctype()</code>, <code>wesctomb()</code>, <code>wesctowcs()</code>, <code>wesctowcs()</code>, <code>wesctowcs()</code></td>
</tr>
<tr>
<td><strong>POSIX_DEVICE_IO</strong></td>
<td><code>clearerr()</code>, <code>fclose()</code>, <code>fdopen()</code>, <code>feof()</code>, <code>ferror()</code>, <code>fflush()</code>, <code>fgets()</code>, <code>fgetenv()</code>, <code>fileno()</code>, <code>fopen()</code>, <code>fprintf()</code>, <code>fputc()</code>, <code>fputs()</code>, <code>freopen()</code>, <code>fscanf()</code>, <code>fwrite()</code>, <code>getc()</code>, <code>getchar()</code>, <code>gets()</code>, <code>open()</code>, <code>perror()</code>, <code>print()</code>, <code>putc()</code>, <code>putchar()</code>, <code>puts()</code>, <code>read()</code>, <code>scanf()</code>, <code>setbuf()</code></td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>Included Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Device_Specific</td>
<td>cfgetispeed(), cfgetispeed(), cfsetispeed(), cfsetispeed(), ctermid(), isatty(), tcdrain(), tcflow(), tcflush(), tcsetattr(), tcsetattrs(), ttyname(), ttyname_r()</td>
</tr>
<tr>
<td>POSIX_Event_Mgmt</td>
<td>FD_CLR(), FD_ISSET(), FD_SET(), FD_ZERO(), pselect(), select()</td>
</tr>
<tr>
<td>POSIX_Fd_Mgmt</td>
<td>dup(), dup2(), fcntl(), fgetpos(), fseekpos(), fseeko(), fsetpos(), ftell(), ftello(), ftuntancate(), lseek(), rewind()</td>
</tr>
<tr>
<td>POSIX_Fifo</td>
<td>mkfifo()</td>
</tr>
<tr>
<td>POSIX_File_Attributes</td>
<td>chmod(), chown(), fchmod(), fchown(), umask()</td>
</tr>
<tr>
<td>POSIX_File_Locking</td>
<td>flockfile(), ftrylockfile(), funlockfile(), gete_unlocked(), pute_unlocked(), putchar_unlocked()</td>
</tr>
<tr>
<td>POSIX_File_System</td>
<td>access(), chmod(), closedir(), creat(), fpathconf(), fstat(), getcwd(), link(), mkdir(), opendir(), pathconf(), readdir(), readdir_r(), remove(), rename(), rewinddir(), rmdir(), stat(), tmpfile(), tmpnam(), unlink(), utime()</td>
</tr>
<tr>
<td>POSIX_File_System_Ext</td>
<td>glob(), globfree()</td>
</tr>
<tr>
<td>POSIX_Job_Controla</td>
<td>setpgid(), tegetpgrp(), tcsetpgrp()</td>
</tr>
<tr>
<td>POSIX_Mult_Processa</td>
<td>_Exit(), _exit(), assert(), atexit(), clock(), execl(), execle(), execle(), execvp(), execv(), exit(), fork(), getegrp(), getpid(), getpid(), getsid(), sleep(), times(), wait(), waitpid()</td>
</tr>
<tr>
<td>POSIX_Networking</td>
<td>accept(), bind(), connect(), endhostent(), endnetent(), endprotoent(), endservent(), freeaddrinfo(), gai_strerror(), getaddrinfo(), gethostbyaddr(), gethostbyname(), gethostent(), gethostname(), getnameinfo(), getnetbyaddr(), getnetbyname(), getnetent(), getpeername(), getprotobyname(), getprotoent(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport(), getservent(), getservname(), getservbyname(), getservbyport()</td>
</tr>
<tr>
<td>POSIX_Pipe</td>
<td>pipe()</td>
</tr>
<tr>
<td>POSIX_Priority_Ranges</td>
<td>sched_get_priority_max(), sched_get_priority_min(), sched_rr_get_interval()</td>
</tr>
<tr>
<td>POSIX_Regexpb</td>
<td>regcomp(), regerror(), regexec(), regfree()</td>
</tr>
<tr>
<td>Unit of Functionality</td>
<td>Included Functions</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>POSIX_RW_LOCKS</td>
<td>pthread_rwlock_destroy(), pthread_rwlock_init(), pthread_rwlock_rdlock(),</td>
</tr>
<tr>
<td></td>
<td>pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock(),</td>
</tr>
<tr>
<td></td>
<td>pthread_rwlock_tryrdlock(), pthread_rwlock_trywrlock(),</td>
</tr>
<tr>
<td></td>
<td>pthread_rwlock_unlock(), pthread_rwlock_wrlock(),</td>
</tr>
<tr>
<td></td>
<td>pthread_rwlockattr_destroy(), pthread_rwlockattr_getpshared(),</td>
</tr>
<tr>
<td></td>
<td>pthread_rwlockattr_init(), pthread_rwlockattr_setpshared()</td>
</tr>
<tr>
<td>POSIX_SHELL_FUNC</td>
<td>pclose(), popen(), system(), wordexp(), wordfree()</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
<td>abort(), alarm(), kill(), pause(), raise(), sigaction(),</td>
</tr>
<tr>
<td></td>
<td>sigaddset(), sigdelset(), sigemptyset(), sigfillset(),</td>
</tr>
<tr>
<td></td>
<td>sigismember(), signal(), sigpending(), sigprocmask(), sigsuspend(), sigwait()</td>
</tr>
<tr>
<td>POSIX_SIGNAL_JUMP</td>
<td>siglongjmp(), sigsetjmp()</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
<td>confstr(), getenv(), setenv(), sysconf(), uname(), unsetenv()</td>
</tr>
<tr>
<td>POSIX_STRING_MATCHING</td>
<td>fnmatch(), getopt()</td>
</tr>
<tr>
<td>POSIX_SYMBOLIC_LINKS</td>
<td>lstat(), readlink(), symlink()</td>
</tr>
<tr>
<td>POSIX_SYSTEM_DATABASE</td>
<td>getgrgid(), getgrgid_r(), getgrnam(), getgrnam_r(),</td>
</tr>
<tr>
<td></td>
<td>getpwnam(), getpwnam_r(), getpwuid(), getpwuid_r()</td>
</tr>
<tr>
<td>POSIX_THREADS_BASE</td>
<td>pthread_atfork(), pthread_attr_destroy(),</td>
</tr>
<tr>
<td></td>
<td>pthread_attr_getdetachstate(), pthread_attr_getschedparam(),</td>
</tr>
<tr>
<td></td>
<td>pthread_attr_getsetstate(), pthread_attr_getinit(),</td>
</tr>
<tr>
<td></td>
<td>pthread_attr_setdetachstate(), pthread_attr_setinit(),</td>
</tr>
<tr>
<td></td>
<td>pthread_atfork(), pthread_cancel(), pthread_cleanup_pop(),</td>
</tr>
<tr>
<td></td>
<td>pthread_cleanup_push(), pthread_cond_broadcast(),</td>
</tr>
<tr>
<td></td>
<td>pthread_cond_destroy(), pthread_cond_init(),</td>
</tr>
<tr>
<td></td>
<td>pthread_cond_timedwait(), pthread_cond_wait(),</td>
</tr>
<tr>
<td></td>
<td>pthread_condattr_destroy(), pthread_condattr_init(),</td>
</tr>
<tr>
<td></td>
<td>pthread_create(), pthread_detach(), pthread_equal(),</td>
</tr>
<tr>
<td></td>
<td>pthread_exit(), pthread_getspecific(), pthread_join(),</td>
</tr>
<tr>
<td></td>
<td>pthread_key_create(), pthread_key_delete(),</td>
</tr>
<tr>
<td></td>
<td>pthread_kill(), pthread_mutex_destroy(),</td>
</tr>
<tr>
<td></td>
<td>pthread_mutex_init(), pthread_mutex_lock(),</td>
</tr>
<tr>
<td></td>
<td>pthread_mutex_trylock(), pthread_mutex_unlock(),</td>
</tr>
<tr>
<td></td>
<td>pthread_mutexattr_destroy(),</td>
</tr>
<tr>
<td></td>
<td>pthread_mutexattr_init(), pthread_once(),</td>
</tr>
<tr>
<td></td>
<td>pthread_self(), pthread_setcancelstate(),</td>
</tr>
<tr>
<td></td>
<td>pthread_setcanceltype(), pthread_setspecific(),</td>
</tr>
<tr>
<td></td>
<td>pthread_sigmask(), pthread_testcancel()</td>
</tr>
</tbody>
</table>
Table 1-1: POSIX.1 Units of Functionality (Continued)

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>Included Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_USER_GROUPS</td>
<td>getegid(), geteuid(), getgid(), getgroups(), getlogin(), getlogin_r(), getuid(), getgid(), setegid(), setgid(), setuid(), seteuid()</td>
</tr>
<tr>
<td>POSIX_WIDE_CHAR_IO</td>
<td>fgetwc(), fgetws(), fputwc(), fputws(), fwscanf(), fprintf(), fтруд, getwc(), getwchar(), getwc(), putwc(), ungetwc(), vfprintf(), vfscanf(), vfprintf(), vuscanf(), wscanf()</td>
</tr>
<tr>
<td>XSI_C_LANG_SUPPORT</td>
<td>_tolower(), _toupper(), a64l(), daylight(), drand48(), erand48(), ffs(), getcontext(), getdate(), getsubopt(), hcreate(), hdestroy(), hsearch(), iconv(), iconv_close(), iconv_open(), initstate(), insque(), isascii(), jrand48(), jrand48(), lcong48(), lfind(), lrand48(), lsearch(), makecontext(), memccpy(), mrand48(), nrand48(), random(), remquo(), seed48(), setcontext(), setstate(), srand48(), srand48(), strcasecmp(), strncasecmp(), strndup(), strfmon(), strncasecmp(), strptime(), swab(), swapcontext(), tdelete(), tfind(), timezone(), toascii(), tsearch(), twalk()</td>
</tr>
<tr>
<td>XSI_DBM</td>
<td>dbm_clearr(), dbm_close(), dbm_delete(), dbm_error(), dbm_fetch(), dbm_firstkey(), dbm_nextkey(), dbm_open(), dbm_store()</td>
</tr>
<tr>
<td>XSI_DEVICE_IO</td>
<td>fntmsg(), poll(), pread(), pwrite(), readv(), writev()</td>
</tr>
<tr>
<td>XSI_DEVICE_SPECIFIC</td>
<td>granpt(), posix_openpt(), ptsname(), unlcpt()</td>
</tr>
<tr>
<td>XSI_DYNAMIC_LINKING</td>
<td>dlclose(), dlerror(), dlopen(), dlset()</td>
</tr>
<tr>
<td>XSI_FD_MGMT</td>
<td>truncate()</td>
</tr>
<tr>
<td>XSI_FILE_SYSTEM</td>
<td>basename(), dirname(), fchdir(), fstatvfs(), ftrw(), lchown(), lockf(), mkod(), mkstemp(), nftw(), realpath(), seekdir(), statefs(), sync(), telldir(), temnam()</td>
</tr>
<tr>
<td>XSI_A8N</td>
<td>catclose(), catgets(), catopen(), nl_langinfo()</td>
</tr>
<tr>
<td>XSI_IPC</td>
<td>ftdl(), msgctl(), msget(), msgrec(), msgsnd(), semctl(), semget(), semop(), shmct(), shmct(), shmget()</td>
</tr>
<tr>
<td>XSI_JOB_CONTROL</td>
<td>tgeetsid()</td>
</tr>
<tr>
<td>XSI_JUMP</td>
<td>_longjmp(), _set jmp()</td>
</tr>
<tr>
<td>XSI_MATH</td>
<td>f0(), ft(), jn(), scalb(), y0(), yf(), yn()</td>
</tr>
<tr>
<td>XSI_MULTI_PROCESS</td>
<td>getpgid(), getpriority(), getrusage(), getsid(), nice(), setpgid(), setpriority(), setrlimit(), uselwait(), ufork(), waitid()</td>
</tr>
<tr>
<td>XSI_SIGNALS</td>
<td>bsd_signal(), killpg(), sigaltstack(), sighold(), sigignore(), sigint(), sigpause(), sigset(), sigact(), sigset(), unalarm()</td>
</tr>
<tr>
<td>XSI_SINGLE_PROCESS</td>
<td>gethostid(), gettimeofday(), putenv()</td>
</tr>
<tr>
<td>XSI_SYSTEM_DATABASE</td>
<td>endpwent(), getpwent(), setpwent()</td>
</tr>
<tr>
<td>XSI_SYSTEM_LOGGING</td>
<td>closelog(), openlog(), setlogmask(), syslog()</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>Included Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSI_THREAD_MUTEX_EXT</td>
<td>pthread_mutexattr_gettype(), pthread_mutexattr_settype()</td>
</tr>
<tr>
<td>XSI_THREADS_EXT</td>
<td>pthread_attr_getguardsize(), pthread_attr_getstack(),</td>
</tr>
<tr>
<td></td>
<td>pthread_attr_setguardsize(), pthread_attr_setstack(), pthread_getconcurrency(),</td>
</tr>
<tr>
<td></td>
<td>pthread_setconcurrency()</td>
</tr>
<tr>
<td>XSI_TIMERS</td>
<td>getitimer(), setitimer()</td>
</tr>
<tr>
<td>XSI_USER_GROUPS</td>
<td>endgrent(), endutxent(), getgrent(), getutxent(),</td>
</tr>
<tr>
<td></td>
<td>getutxid(), getutxline(), setgrent(), setreuid(), setutxent(),</td>
</tr>
<tr>
<td>XSI_WIDE_CHAR</td>
<td>wcswidth(), wcwidth()</td>
</tr>
</tbody>
</table>

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_REGEXP, 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)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_ADA_LANG_SUPPORT</td>
<td>Extra requirements specified in POSIX.5c, Section 2.8</td>
</tr>
<tr>
<td>System_Storage_Elements</td>
<td>All(^a)</td>
</tr>
<tr>
<td>POSIX_Page_Alignment</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Supplement_To_Ada_IO</td>
<td>All</td>
</tr>
<tr>
<td>Ada_Task_Identification</td>
<td>All</td>
</tr>
<tr>
<td>Ada_Streams</td>
<td>All</td>
</tr>
</tbody>
</table>

\(^a\): 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.
Table 1-3: POSIX.5 Units of Functionality (Device IO)

<table>
<thead>
<tr>
<th>POSIX_DEVICE_IO Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_IO</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Close</td>
</tr>
<tr>
<td></td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Write</td>
</tr>
<tr>
<td></td>
<td>Generic_Read</td>
</tr>
<tr>
<td></td>
<td>Generic_Write</td>
</tr>
<tr>
<td></td>
<td>Is_Open</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>POSIX_DEVICE_SPECIFIC Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Terminal_Functions</td>
<td>Get_Terminal_Characteristics</td>
</tr>
<tr>
<td></td>
<td>Get_Controlling_Terminal_Name</td>
</tr>
<tr>
<td></td>
<td>Set_Terminal_Characteristics</td>
</tr>
<tr>
<td></td>
<td>Terminal_Modes_Of</td>
</tr>
<tr>
<td></td>
<td>Define_Terminal_Modes</td>
</tr>
<tr>
<td></td>
<td>Bits_Per_Character_Of</td>
</tr>
<tr>
<td></td>
<td>Define_Bits_Per_Character</td>
</tr>
<tr>
<td></td>
<td>Special_Control_Character_Of</td>
</tr>
<tr>
<td></td>
<td>Define_Special_Control_Character</td>
</tr>
<tr>
<td></td>
<td>Disable_Control_Character</td>
</tr>
<tr>
<td></td>
<td>Input_Time_Of</td>
</tr>
<tr>
<td></td>
<td>Define_Input_Time</td>
</tr>
<tr>
<td></td>
<td>Minimum_Input_Count_Of</td>
</tr>
<tr>
<td></td>
<td>Define_Minimum_Input_Count</td>
</tr>
<tr>
<td></td>
<td>Input_Baud_Rate_Of</td>
</tr>
<tr>
<td></td>
<td>Output_Baud_Rate_Of</td>
</tr>
<tr>
<td></td>
<td>Define_Input_Baud_Rate</td>
</tr>
<tr>
<td></td>
<td>Define_Output_Baud_Rate</td>
</tr>
<tr>
<td></td>
<td>Send_Break</td>
</tr>
<tr>
<td></td>
<td>Drain</td>
</tr>
<tr>
<td></td>
<td>Discard_Data</td>
</tr>
<tr>
<td></td>
<td>Flow</td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Is_A_Terminal</td>
</tr>
<tr>
<td></td>
<td>Get_Terminal_Name</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>POSIX_EVENT_MGMT Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Event_Management</td>
<td>Make_Empty</td>
</tr>
<tr>
<td></td>
<td>Add</td>
</tr>
<tr>
<td></td>
<td>Remove</td>
</tr>
<tr>
<td></td>
<td>In_Set</td>
</tr>
<tr>
<td></td>
<td>Select_File</td>
</tr>
<tr>
<td></td>
<td>For_Every_File_In</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Signal</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_File_Locking</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Duplicate</td>
</tr>
<tr>
<td></td>
<td>Duplicate_And_Close</td>
</tr>
<tr>
<td></td>
<td>Get_File_Control</td>
</tr>
<tr>
<td></td>
<td>Set_File_Control</td>
</tr>
<tr>
<td></td>
<td>Get_Close_On_Exec</td>
</tr>
<tr>
<td></td>
<td>Set_Close_On_Exec</td>
</tr>
<tr>
<td></td>
<td>Seek</td>
</tr>
<tr>
<td></td>
<td>File_Size</td>
</tr>
<tr>
<td></td>
<td>File_Position</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Files</td>
<td>Create_FIFO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Permissions</td>
<td>Set_Allowed_Process_Permissions</td>
</tr>
<tr>
<td></td>
<td>Get_Allowed_Process_Permissions</td>
</tr>
<tr>
<td>POSIX_Files</td>
<td>Change_Owner_And_Group</td>
</tr>
<tr>
<td></td>
<td>Change_Permissions</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Configurable_File_Limits</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_File_Status</td>
<td>All</td>
</tr>
</tbody>
</table>
Table 1-9: POSIX.5 Units of Functionality (File System) (Continued)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Files</td>
<td>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</td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Open_Or_Create</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_JOB_CONTROL\a</td>
<td></td>
</tr>
<tr>
<td>POSIX_Process_Identification</td>
<td>Set_Process_Group_Id Create_Process_Group</td>
</tr>
<tr>
<td>POSIX_Terminal_Functions</td>
<td>Get_Process_Group_Id Set_Process_Group_Id</td>
</tr>
<tr>
<td>POSIX_Signals</td>
<td>Set_Stopped_Child_Signal Stopped_Child_Signal_Enabled</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_PROCESS_Primitives</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Unsafe_Process_Primitives</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Process_Times</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Process_Identification</td>
<td>Get_Process_Id Get_Parent_Process_Id</td>
</tr>
</tbody>
</table>
### Table 1-12: POSIX.5 Units of Functionality (Networking)

<table>
<thead>
<tr>
<th>POSIX_NETWORKING</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package</strong></td>
<td><strong>Subprograms</strong></td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Get_Owner</td>
</tr>
<tr>
<td></td>
<td>Set_Socket_Process_Owner</td>
</tr>
<tr>
<td></td>
<td>Set_Socket_Group_Owner</td>
</tr>
<tr>
<td></td>
<td>Set_Buffer</td>
</tr>
<tr>
<td></td>
<td>Get_Buffer</td>
</tr>
<tr>
<td>POSIX_Sockets</td>
<td>All\textsuperscript{a}</td>
</tr>
<tr>
<td>POSIX_Sockets_Local</td>
<td>All\textsuperscript{a}</td>
</tr>
<tr>
<td>POSIX_Sockets_Internet</td>
<td>All\textsuperscript{b}</td>
</tr>
</tbody>
</table>

\textsuperscript{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.

\textsuperscript{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)

<table>
<thead>
<tr>
<th>POSIX_PIPES</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package</strong></td>
<td><strong>Subprograms</strong></td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Create_Pipe</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>POSIX_PRIORITY_RANGES</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package</strong></td>
<td><strong>Subprograms</strong></td>
</tr>
<tr>
<td>POSIX_Process_Scheduling</td>
<td>Get_Maximum_Priority</td>
</tr>
<tr>
<td></td>
<td>Get_Minimum_Priority</td>
</tr>
<tr>
<td></td>
<td>Get_Round_Robin Interval</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Table 1-15: POSIX.5 Units of Functionality (Signals)

<table>
<thead>
<tr>
<th>POSIX_SIGNALS</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_Signals</td>
<td>Add_Signal</td>
</tr>
<tr>
<td></td>
<td>Add_All_Signals</td>
</tr>
<tr>
<td></td>
<td>Delete_Signal</td>
</tr>
<tr>
<td></td>
<td>Delete_All_Signals</td>
</tr>
<tr>
<td></td>
<td>Is_Member</td>
</tr>
<tr>
<td></td>
<td>Send_Signal</td>
</tr>
<tr>
<td></td>
<td>Set_Block_Signals</td>
</tr>
<tr>
<td></td>
<td>Block_Signals</td>
</tr>
<tr>
<td></td>
<td>Unblock_Signals</td>
</tr>
<tr>
<td></td>
<td>Blocked_Signals</td>
</tr>
<tr>
<td></td>
<td>Ignore_Signal</td>
</tr>
<tr>
<td></td>
<td>Unignore_Signal</td>
</tr>
<tr>
<td></td>
<td>Is_Ignored</td>
</tr>
<tr>
<td></td>
<td>Install_Empty_Handler</td>
</tr>
<tr>
<td></td>
<td>Pending_Signals</td>
</tr>
<tr>
<td></td>
<td>Await_Signal(^a)</td>
</tr>
<tr>
<td></td>
<td>Await_Signal_Or_Timeout(^a)</td>
</tr>
<tr>
<td></td>
<td>Interrupt_Task</td>
</tr>
<tr>
<td></td>
<td>Get_Signal(^b)</td>
</tr>
<tr>
<td></td>
<td>Set_Signal(^b)</td>
</tr>
<tr>
<td></td>
<td>Get_Notification</td>
</tr>
<tr>
<td></td>
<td>Set_Notification</td>
</tr>
<tr>
<td></td>
<td>Get_Data(^b)</td>
</tr>
<tr>
<td></td>
<td>Set_Data(^b)</td>
</tr>
</tbody>
</table>

a. Return type Signal
b. Operation on type Signal_Event

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

<table>
<thead>
<tr>
<th>POSIX_SINGLE_PROCESS</th>
<th>Subprograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Limits</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Options</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Profiles</td>
<td>All(^a)</td>
</tr>
<tr>
<td>POSIX_Configurable_System_Limits</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Calendar</td>
<td>All</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Table 1-18: POSIX.5 Units of Functionality (User Groups)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>POSIX_Process_Identification</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>PSE51</th>
<th>PSE52</th>
<th>PSE53</th>
<th>PSE54</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_ADA_LANG_SUPPORTa</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_C_LANG_JUMPb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_C_LANG_MATHb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_C_LANG_SUPPORTb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_DEVICE_SPECIFIC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_EVENT_MGMT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_FILE_ATTRIBUTES</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_FILE_LOCKINGb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM_EXTb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_JOB_CONTROL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_MULTI_PROCESS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_NETWORKING</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_PIPE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_PRIORITY_RANGES</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_REGEXPb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_RW_LOCKSb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_SHELL_FUNCb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_SIGNAL_JUMPb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_SINGULAR_PROCESS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_STRING_MATCHINGb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_SYMBOLIC_LINKSb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_SYSTEM_DATABASE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_THREADS_BASEb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_USER_GROUPS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_WIDE_CHAR_IOb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_C_LANG_SUPPORTb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_DBMb</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
<th>PSE51</th>
<th>PSE52</th>
<th>PSE53</th>
<th>PSE54</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSI_DEVICE_IO\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_DEVICE_SPECIFIC\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_DYNAMIC_LINKING\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>XSI_FD_MGMT\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_FILE_SYSTEM\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_I18N\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_IPC\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_JOB_CONTROL\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_JUMP\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_MATH\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_MULTI_PROCESS\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_SIGNALS\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_SINGLE_PROCESS\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_SYSTEM_DATABASE\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_SYSTEM_LOGGING\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>XSI_THREAD_MUTEX_EXT\textsuperscript{b}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XSI_THREADS_EXT\textsuperscript{b}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XSI_TIMERS\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_USER_GROUPS\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XSI_WIDE_CHAR\textsuperscript{b}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

a. Required only for the Ada-Language option
b. Required only for the C-Language option

Table 1-20: POSIX.1 Option Requirements

<table>
<thead>
<tr>
<th>Option</th>
<th>PSE51</th>
<th>PSE52</th>
<th>PSE53</th>
<th>PSE54</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_ADVISORY_INFO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_ASYNCHRONOUS_IO</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_BARRIERS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_CHOWN_RESTRICTED</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_CLOCK_SELECTION</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_CPUTIME</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_FSYNC</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_IPv6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSIX_JOB_CONTROL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_MAPPED_FILES</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_MEMLOCK</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_MEMLOCK_RANGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_MEMORY_PROTECTION</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POSIX_MESSAGE_PASSING</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
<table>
<thead>
<tr>
<th>Option</th>
<th>PSE51</th>
<th>PSE52</th>
<th>PSE53</th>
<th>PSE54</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_MONOTONIC_CLOCK</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_NO_TRUNC</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>_POSIX_PRIORITIZED_IO</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING</td>
<td>-</td>
<td>-</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_RAW_SOCKETS</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>_POSIX_READER_WRITER_LOCKS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>_POSIX_REALTIME_SIGNALS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_REGEXP</td>
<td></td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SAVED_IDS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SEMAPHORES</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SHELL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SPAWN</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SPIN_LOCKS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_SPORADIC_SERVER</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>_POSIX_SYNCHRONIZED_IO</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADDR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_CPUTIME</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_SCHEDULING</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_INHERIT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_PROTECT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESSION_SHARED</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREAD_SAFE_FUNCTIONS</td>
<td>See units of functionality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_SPOORADIC_SERVER</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_THREADS</td>
<td>See units of functionality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_TIMERS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_TRACE</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_TRACE_EVENT_FILTER</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_TRACE_INHERIT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>_POSIX_TRACE_LOG</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX_TYPED_MEMORY_OBJECTS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>_POSIX_VDISABLE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX2_C_BIND</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX2_C_DEV</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X&lt;sup&gt;b&lt;/sup&gt;</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX2_CHAR_TERM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX2_FORT_DEV</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>_POSIX2_FORT_RUN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX2_LOCALEDEF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>_POSIX2_PBS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>_POSIX2_PBS_ACCOUNTING</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Table 1-20: POSIX.1 Option Requirements (Continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>PSE51</th>
<th>PSE52</th>
<th>PSE53</th>
<th>PSE54</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX2_PBS_CHECKPOINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_POSIX2_PBS_LOCATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_POSIX2_PBS_MESSAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_POSIX2_PBS_TRACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_POSIX2_SW_DEV</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>_POSIX2_UPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_CRYPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_ENH_I18N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_LEGACY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_REALTIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_REALTIME_THREADS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_SHM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_STREAMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_XOPEN_UNIX</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>POSIX.1 Option</th>
<th>POSIX.5c Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_ADVISE_INFO</td>
<td>none</td>
</tr>
<tr>
<td>_POSIXASYNCHRONOUS_IO</td>
<td>Asynchronous I/O</td>
</tr>
<tr>
<td>_POSIX_BARRIERS</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_CHOWN_RESTRICTED</td>
<td>Change Owner Restriction</td>
</tr>
<tr>
<td>_POSIX_CLOCK_SELECTION</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_CPUTIME</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_FSYNC</td>
<td>File Synchronization</td>
</tr>
<tr>
<td>_POSIX_IPV6</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_MAPPED_FILES</td>
<td>Memory Mapped Files</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK</td>
<td>Memory Locking</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK_RANGE</td>
<td>Memory Range Locking</td>
</tr>
<tr>
<td>_POSIX_MEMORY_PROTECTION</td>
<td>Memory Protection</td>
</tr>
<tr>
<td>_POSIX_MESSAGE_PASSING</td>
<td>Message Queues</td>
</tr>
<tr>
<td>_POSIX_MONOTONIC_CLOCK</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_NO_TRUNC</td>
<td>Filename Truncation\a</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING</td>
<td>Priority Process Scheduling</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING</td>
<td>Priority Process Scheduling</td>
</tr>
<tr>
<td>_POSIX_RAW_SOCKETS</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_READER_WRITER_LOCKS</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_REALTIME_SIGNALS</td>
<td>Realtime Signals</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
<table>
<thead>
<tr>
<th>POSIX.1 Option</th>
<th>POSIX.5c Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_REXEXP</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_SAVED_IDS</td>
<td>Saved IDs Support</td>
</tr>
<tr>
<td>_POSIX_SEMAPHORES</td>
<td>Semaphores</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS</td>
<td>Shared Memory Objects</td>
</tr>
<tr>
<td>_POSIX_SHELL</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX_SPAWN</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX_SPIN_LOCKS</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_SPOREADIC_SERVER</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_SYNCHRONIZED_IO</td>
<td>Synchronized I/O</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADD</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX_THREAD_CPUTIME</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_THREAD_Prio_INHERIT</td>
<td>Mutex Priority Inheritance</td>
</tr>
<tr>
<td>_POSIX_THREAD_Prio_PROTECT</td>
<td>Mutex Priority Ceiling</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_SCHEDU</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESS_SHARED</td>
<td>Process Shared</td>
</tr>
<tr>
<td>_POSIX_THREAD_SAFE_FUNCTIONS</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX_THREAD_SPORADIC_SERVER</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_THREADS</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_TIMERS</td>
<td>Timers</td>
</tr>
<tr>
<td>_POSIX_TRACE</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_TRACE_EVENT_FILTER</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_TRACE_INHERIT</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_TRACE_LOG</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_TYPED_MEMORY_OBJECTS</td>
<td>none</td>
</tr>
<tr>
<td>_POSIX_VDISABLE</td>
<td>C-language specific</td>
</tr>
<tr>
<td>_POSIX2_C_BIND</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_C_DEV</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_CHAR_TERM</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_FORT_DEV</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_FORT_RUN</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_LOCALEDEF</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_PBS</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_PBS_ACCOUNTING</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_PBS_CHECKPOINT</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_PBS_LOCATE</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_PBS_MESSAGE</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_PBS_TRACK</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_SW_DEV</td>
<td>not applicable</td>
</tr>
<tr>
<td>_POSIX2_UPE</td>
<td>not applicable</td>
</tr>
<tr>
<td>_XOPEN_CRYPT</td>
<td>none</td>
</tr>
<tr>
<td>_XOPEN_ENH_I18N</td>
<td>none</td>
</tr>
</tbody>
</table>
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.)

---

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

<table>
<thead>
<tr>
<th>POSIX.1 Option</th>
<th>POSIX.5c Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOPEN_LEGACY</td>
<td>none</td>
</tr>
<tr>
<td>XOPEN_REALTIME</td>
<td>none</td>
</tr>
<tr>
<td>XOPEN_REALTIME_THREADS</td>
<td>none</td>
</tr>
<tr>
<td>XOPEN_SHM</td>
<td>none</td>
</tr>
<tr>
<td>XOPEN_STREAMS</td>
<td>none</td>
</tr>
<tr>
<td>XOPEN_UNIX</td>
<td>none</td>
</tr>
</tbody>
</table>

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

---

Copyright © 2003 IEEE. All rights reserved. This is an unapproved IEEE Standards Draft, subject to change.
Section 2: Normative References

2.1 Normative References

The following standards contain provisions which, through references in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this 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. 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/).

POSIX Ada Language Interfaces—Part 1: Binding for System Application
Program Interface (API)—Amendment 2: Protocol Independent Interfaces.

omy of International Standardized Profiles -- Part 1: General principles and
documentation framework.

omy of International Standardized Profiles -- Part 3: Principles and Taxonomy
for Open System Environment Profiles.
Section 3: Terms and Definitions

3.1 Terminology

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

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

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

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

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

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

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

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

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

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

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

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

### 3.2 Definitions

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

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

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

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

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

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
that it uses, and the relationships between these interfaces. [ISO/IEC TR 10000-3:1998 {8}]

3.2.5 Conformance Document: A document provided by an implementor that contains implementation details as described in 5.1.1.2.

3.2.6 Development Platform: A system used to prepare an application for execution. Such a system is possibly distinct from the system on which the application will execute.

3.2.7 Embedded Computer System: A computer (and its software) is considered embedded if it is an integral component of a larger system and is used to control and/or directly monitor that system, using special hardware devices.

3.2.8 Generic Application Environment Profile: An Application Environment Profile which is not specific to a particular community of use [ISO/IEC TR 10000-3:1998 {8}]

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

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

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

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

3.2.13 International Standardized Profile (ISP): An internationally agreed-to, harmonized document which identifies a standard or group of standards, together with options and parameters, necessary to accomplish a function or set of functions. [ISO/IEC TR 10000-1:1998 {8}]

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
3.2.14 Open System Environment (OSE): The comprehensive set of interfaces, services, and supporting formats for interoperability and/or for portability of applications, data or people, as specified by information technology standards and profiles. [ISO/IEC TR 10000-3:1998 {8}]

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

See also 3.2.21, “Unbounded Priority Inversion”.

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

3.2.17 Realtime Environment Profile: A profile designed to support applications requiring bounded response.

3.2.18 System Documentation: All documentation provided with an implementation, except the conformance document.

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

3.2.19 Subprofiling Option Group: A unit of functionality (See 3.2.22).

3.2.20 System Profile: An Application Environment Profile that specifies a set of functions necessary to support a 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. [ISO/IEC TR 10000-3:1998 {8}]

NOTE: A system profile is defined in terms of component profiles that specify units of functionality that can be combined to realize the application platform.

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

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
ple, this can happen when a lower priority thread is holding a lock also requested by the high priority thread, and then one or more medium priority threads request execution, thus preempting the lower priority thread.

See also 3.2.15, “Priority Inversion”.

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

### 3.3 Rationale for definitions

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

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

- Are programs that understand physics and/or hardware embedded? For example one that uses finite-element methods to predict fluid flow over airplane wings? No. These programs are never considered to be embedded because they are not an integral component of a larger system.

- Is the internal microprocessor controlling a disk drive an example of an embedded system? Yes, regardless of what the disk drive is used for. The software (firmware, actually) within the disk drive controls the HDA (head disk assembly) hardware, and is hard realtime as well.

- I/O drivers control hardware, so does presence of an I/O driver imply that the computer executing the driver is embedded? No, because that computer may be a general-purpose computer that is not part of a larger system.

- Is a PDA (Personal Digital Assistant) an embedded system? No. People often say that PDAs are embedded because they are very small and constrained, and because PDA OS and application software is kept in non-volatile memory, but PDAs parallel the desktop systems used to run office productivity applications, and no special hardware is being controlled.

- Is the microprocessor controlling a cellphone an embedded system? Yes. The firmware in the cellphone is controlling the radio hardware.

- Are the computers in a big phased-array radar considered embedded? These radars are ten-story buildings with one to three 100-foot diameter radiating patches on the sloped sides of the building. Yes. These computers are generally some of the most powerful computers available when the system was built, live in a large computer room occupying almost one whole floor of a building, may be hundreds of meters away from the radar hard-
ware, but these computers are still an integral component of a larger system.

— Is a traditional Flight Management System (FMS) built into an airplane cockpit considered embedded? If the FMS is not connected to the avionics, and is used only for logistics computations, a function readily performed on a laptop, then the FMS is clearly not embedded.

— Are the computers in a hardware-in-the-loop (HIL) simulator embedded? Yes, both in the simulator, and in the thing being tested in the HIL simulator. Hardware is being controlled on both sides.

— Is the computer controlling a pacemaker in a person’s chest an embedded computer? Yes. In this case the “system” is the combination of the pacemaker and the person’s heart.

— Is the computer controlling fuel injection in an automobile engine embedded? Yes. It is part of a larger system, the engine, and it is directly monitoring and controlling the engine through special hardware.
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 typesetting and there is no normative difference between these two cases.
The conventions listed previously are for ease of reading only. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in this standard.

Notes provided as parts of labeled tables and figures are integral parts of this Standard (normative). Footnotes and notes within the body of the text are for information only (informative).

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

4.2 Abbreviations

For the purposes of this document the following abbreviations apply:


4.2.3 MMU: Memory management unit.


4.2.6 POSIX.13: This standard.

4.2.7 AEP: Application Environment Profile.

4.2.8 ISP: International Standardized Profile.

4.2.9 OSE: Open System Environment.

4.2.10 PSE: Generic Environment Profile.

4.2.11 PSE51: The Minimal Realtime System Profile defined herein.

4.2.12 PSE52: The Realtime Controller System Profile defined herein.

4.2.13 PSE53: The Dedicated Realtime System Profile defined herein.

4.2.14 PSE54: The Multi-Purpose Realtime System Profile defined herein.

4.2.15 PSE5X: Any one of the PSE51, PSE52, PSE53 or PSE54 profiles.
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-
documents that document the base standard conformance. This will primarily apply to implementations that support the PSE53 or PSE54 Profiles.

(2) If the implementation does not fully conform to one or more of the referenced base standards, or if separate base standard conformance documents are not cited, the implementation shall document the specific extent of conformance to each such base standard. This specification shall include:

— A complete list of interfaces from the base standard that are present in the implementation.

— Limit values whose specification is normally required in a conformance document for the base standard (e.g. the limit values found in the <limits.h> and <unistd.h> headers), stating values, the conditions under which those values may change, and the limits of such variations, if any.

— A description of the behavior of the implementation for all implementation-defined features specified by those portions of the base standard that the implementation provides. This requirement shall be met by listing these features and providing either a specific reference to the system documentation or providing full syntax and semantics of these features. The conformance document may specify the behavior of the implementation for those features where the referenced standards state that the implementations may vary or where features are identified as undefined or unspecified.

Regardless of whether separate base standard conformance documents are cited, the conformance document for these profile(s) shall contain a statement that indicates the full name, number, and date of the standard (i.e. the profile standard) that applies. The conformance document may also list international standards that are available for use by a Conforming POSIX.13 Application. Applicable characteristics where documentation is required by one of these standards or by standards of government bodies, may also be included.

5.1.2 Application Conformance

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

Any application that conforms to one or more of these profiles under the C-Language option also conforms to POSIX.1. Any application that conforms to one or more of these profiles under the Ada-Language option also conforms to POSIX.5c.
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)
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.
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>`:
  
  ```c
  _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>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_C_LANG_JUMP</td>
</tr>
<tr>
<td>POSIX_C_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_C_LIB_EXT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_FILE_LOCKING</td>
</tr>
<tr>
<td>POSIX_PRIORITY_RANGES</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
<tr>
<td>POSIX_THREADS_BASE</td>
</tr>
<tr>
<td>XSI_THREAD_MUTEX_EXT</td>
</tr>
<tr>
<td>XSI_THREADS_EXT</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
The Minimal 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 6-2: POSIX.1 Option Requirements

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_CLOCK_SELECTION</td>
</tr>
<tr>
<td>POSIX_FSYNC</td>
</tr>
<tr>
<td>POSIX_MEMLOCK</td>
</tr>
<tr>
<td>POSIX_MEMLOCK_RANGE</td>
</tr>
<tr>
<td>POSIX_MONOTONIC_CLOCK</td>
</tr>
<tr>
<td>POSIX_NO_TRUNC</td>
</tr>
<tr>
<td>POSIX_REALTIME_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SEMAPHORES</td>
</tr>
<tr>
<td>POSIX_SHARED_MEMORY_OBJECTS</td>
</tr>
<tr>
<td>POSIX_SYNCHRONIZED_IO</td>
</tr>
<tr>
<td>POSIX_THREAD_ATTR_STACKADDR</td>
</tr>
<tr>
<td>POSIX_THREAD_ATTR_STACKSIZE</td>
</tr>
<tr>
<td>POSIX_THREAD_CPUTIME</td>
</tr>
<tr>
<td>POSIX_THREAD_PRIO_INHERIT</td>
</tr>
<tr>
<td>POSIX_THREAD_PRIO_PROTECT</td>
</tr>
<tr>
<td>POSIX_THREAD_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>POSIX_THREAD_SPORADIC_SERVER</td>
</tr>
<tr>
<td>POSIX_TIMEOUTS</td>
</tr>
<tr>
<td>POSIX_TIMERS</td>
</tr>
</tbody>
</table>

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 PSE51 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 PSE51 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.
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 6-2 through Table 6-18):

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

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_ADA_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_FILE_LOCKING</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Synchronization</td>
</tr>
<tr>
<td>Memory Locking</td>
</tr>
<tr>
<td>Memory Range Locking</td>
</tr>
<tr>
<td>Filename Truncation</td>
</tr>
<tr>
<td>Realtime Signals</td>
</tr>
<tr>
<td>Semaphores</td>
</tr>
<tr>
<td>Shared Memory Objects</td>
</tr>
<tr>
<td>Synchronized I/O</td>
</tr>
<tr>
<td>Mutexes Support</td>
</tr>
<tr>
<td>Mutex Priority Inheritance</td>
</tr>
<tr>
<td>Mutex Priority Ceiling</td>
</tr>
<tr>
<td>Timers</td>
</tr>
</tbody>
</table>

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.

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.)

Implementations of the PSE51 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.

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.

6.3 Application Constraints

The Minimal Realtime System profile defined in this standard requires only specific Units of Functionality of the required standards. The absence of particular elements of these standards introduces constraints on the use of some of the features of particular operations. This clause defines the constraints that an application strictly conforming to one of the profiles shall observe when using each of the operations required by that profile.

6.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm()</td>
<td>sigaddset()</td>
<td>sigpending()</td>
<td>timer_getoverrun()</td>
</tr>
<tr>
<td>clock_gettime()</td>
<td>sigdelset()</td>
<td>sigprocmask()</td>
<td>timer_gettime()</td>
</tr>
<tr>
<td>kill()</td>
<td>sigemptyset()</td>
<td>sigqueue()</td>
<td>timer_setitimer()</td>
</tr>
<tr>
<td>raise()</td>
<td>sigfillset()</td>
<td>sigset()</td>
<td>times()</td>
</tr>
<tr>
<td>sem_post()</td>
<td>sigismember()</td>
<td>sysconf()</td>
<td>uname()</td>
</tr>
<tr>
<td>sigaction()</td>
<td>signal()</td>
<td>time()</td>
<td></td>
</tr>
</tbody>
</table>

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 profile 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()`, `getc()`, `gets()`, `open()`, `perror()`, `printf()`, `putc()`, `putchar()`, `puts()`, `read()`, `scanf()`, `vfprintf()`, `vfscanf()`, `vfprintf()`, `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.

---

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
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 be-
cause 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 sys-
tem 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) An application strictly conforming to PSE51 shall not require any of the Input/Output operations (Read, Write, Get, Put, etc.) contained in the predefined Ada I/O packages or their instantiations to 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.

(2) An application strictly conforming to PSE51 shall use the Name of the Open operations contained in the predefined Ada I/O packages or their instantiations only to specify the name of the object without any file system semantics implied, since this profile does not require general file system capabilities.

(3) An application strictly conforming to PSE51 shall not call any of the Create or Delete operations contained in the predefined Ada I/O packages or their instantiations, since this profile does not require general file system capabilities.

6.4 Shell and Utility Requirements

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

6.5 Development Platform Requirements

One or more of the development options in 6.5.1 and 6.5.2 shall be implemented.

6.5.1 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.
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-
eas. Instead of full file system support, basic device I/O (read, write, open, close, control) is considered sufficient for kernels of this size. Systems of this size frequently do not include process isolation hardware or software; therefore, multiple processes (as opposed to threads) may not be supportable.

System options that allow an application to be upwards compatible without modifying application source code have been chosen. For example, although the assumed hardware model implies fixed address space without a Memory Management Unit (MMU), the symbol _POSIX_MEMLOCK is still defined. This increases portability of the application code to higher level systems that do not necessarily have the same restrictions.

6.6.1.1 Process Primitives

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

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

6.6.1.2 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.
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.
### 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.
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(max_FIFO_prio-31,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-
gets not using a MMU for virtual memory, the locking functions do nothing and always report success, while in the larger profiles there really is memory to be locked. In summary, by requiring this service in the PSE51 profile, it is possible to write portable application code that runs correctly in all the profiles.

6.6.1.12 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.

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.

6.6.1.13 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. Bounding the execution times of the different threads in the application increases predictability and reliability.

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

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

### 6.6.1.14 Message Passing

In the PSE51 profile of IEEE Std 1003.13-1998, message queues were required because commercial realtime kernels available at that time with similar functionality to the Minimal Realtime System Profile typically included some form of message queueing mechanism for communication between threads.

However, many embedded realtime applications for small systems do not require message queues and this feature makes the implementation larger. Because message queues can be easily implemented by the application using mutexes and condition variables, this version of the standard has dropped the requirement to support message queues.

### 6.6.1.15 Threads

The basic assumption in this profile is that the system will consist of a single (implicit) process, with multiple threads. Therefore, all basic thread services are required, except for those related to multiple processes. The POSIX_THREADS_BASE unit of functionality was specified in this document instead of the _POSIX_THREADS option, because this option requires reader/writer locks, but this profile does not.

### 6.6.1.16 Tracing

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

### 6.6.1.17 Networking

Although some small embedded systems require networking services, most don’t, so to keep the implementation small, this unit of functionality is not required.
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_REGEXP 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_USERGROUPS, 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.
The _XOPEN_CRYPT option provides cryptography facilities that are not required in PSE51 environments.

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

The _XOPEN_STREAMS option provides facilities that are mainly related to networking, and thus are not required for PSE51 environments, as discussed above.

6.6.1.21 Language-Specific Services for the C Programming Language

Support for the C Language is required in the C Language option, with the exceptions of the POSIX_C_LANG_MATH and POSIX_C_LANG_WIDE_CHAR units of functionality. The reasons for these exceptions are that these are very large libraries that are not necessary for many of the PSE51 applications.

6.6.1.22 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.

6.6.2 Shell and Utility Requirements

Because the Minimal Realtime System Profile is intended for small embedded systems which usually have no terminal or 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.

6.6.3 Development Platform Requirements

The embedded nature of the PSE51 execution platform makes it difficult to use as a development platform. Therefore, the implementation is required to define a development environment in which a PSE51 application can be prepared for execution on the target platform. The development platform depends on the language option chosen by the implementation.
Minimal Realtime System Profile

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
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.
### 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>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_C_LANG_JUMP</td>
</tr>
<tr>
<td>POSIX_C_LANG_MATH</td>
</tr>
<tr>
<td>POSIX_C_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
</tr>
<tr>
<td>POSIX_FILE_LOCKING</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
</tr>
<tr>
<td>POSIX_PRIORITY_RANGES</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
<tr>
<td>POSIX_THREADS_BASE</td>
</tr>
<tr>
<td>XSI_THREAD_MUTEX_EXT</td>
</tr>
<tr>
<td>XSI_THREADS_EXT</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved. This is an unapproved IEEE Standards Draft, subject to change.
The Realtime Controller System Profile implementation shall support the following options defined in POSIX.1, by defining the associated symbol with a value greater than zero:

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 PSE52 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 PSE52 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.

Operating System Interface Requirements
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>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_ADA_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Synchronization</td>
</tr>
<tr>
<td>Memory Mapped Files</td>
</tr>
<tr>
<td>Memory Locking</td>
</tr>
<tr>
<td>Memory Range Locking</td>
</tr>
<tr>
<td>Message Queues</td>
</tr>
<tr>
<td>Filename Truncation</td>
</tr>
<tr>
<td>Realtime Signals</td>
</tr>
<tr>
<td>Semaphores</td>
</tr>
<tr>
<td>Shared Memory Objects</td>
</tr>
<tr>
<td>Synchronized I/O</td>
</tr>
<tr>
<td>Mutexes Support</td>
</tr>
<tr>
<td>Mutex Priority Inheritance</td>
</tr>
<tr>
<td>Mutex Priority Ceiling</td>
</tr>
<tr>
<td>Timers</td>
</tr>
</tbody>
</table>

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.

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.)

Implementations of the PSE52 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.

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.

7.3 Application Constraints

The Realtime Controller System profile defined in this standard requires only specific Units of Functionality of the required standards. The absence of particular elements of these standards introduces constraints on the use of some of the features of particular operations. This clause defines the constraints that an application strictly conforming to one of the profiles shall observe when using each of the operations required by that profile.

7.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm()</td>
<td>sigaddset()</td>
<td>sigpending()</td>
<td>timer_getoverrun()</td>
<td></td>
</tr>
<tr>
<td>clock_gettime()</td>
<td>sigdelset()</td>
<td>sigprocmask()</td>
<td>timer_gettime()</td>
<td></td>
</tr>
<tr>
<td>kill()</td>
<td>sigemptyset()</td>
<td>sigqueue()</td>
<td>timer_settime()</td>
<td></td>
</tr>
<tr>
<td>raise()</td>
<td>sigfillset()</td>
<td>sigset()</td>
<td>times()</td>
<td></td>
</tr>
<tr>
<td>sem_post()</td>
<td>sigismember()</td>
<td>sysconf()</td>
<td>uname()</td>
<td></td>
</tr>
<tr>
<td>sigaction()</td>
<td>signal()</td>
<td>time()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

Implementations of PSE52 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.

7.4 Shell and Utility Requirements

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

7.5 Development Platform Requirements

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

7.5.1 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.

7.5.1.1 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.

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

7.6.1.1 Process Primitives

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

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

7.6.1.2 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.

7.6.1.3 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.
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.
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
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 pri-

ority under the round-robin policy (SCHED_RR) to fairly share the available por-
tion 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 mak-
ing 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(max_FIFO_prio-31,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 spec-
ifies 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 re-
quirement just by specifying in the conformance document that the scheduling al-
location domain is of size one and that static binding of threads to allocation
domains is the default behavior.

7.6.1.11 Process Memory Locking

Process memory locking is inherent in systems following this profile because most
PSE52 targets have no MMU and thus swapping is not supported; code and data
stays 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 PSE52 target and unit test that code on a much larger and
more capable platform, perhaps a PSE54, with minimal modification. In those tar-
gets not using an MMU for virtual memory, the locking functions do nothing and
always report success, while in the larger profiles there really is memory to be
locked. In summary, by requiring this service in the PSE52 profile, it is possible to
write portable application code that runs correctly in all the profiles.
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 filesystem 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.
The Timeouts option is a general requirement for realtime applications and thus is required in this profile.

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

### 7.6.1.14 Message Passing

Currently available commercial realtime kernels with similar functionality to the Realtime Controller System Profile typically include some form of message queueing mechanism for communication between threads. The POSIX.1 Message Passing offers an appropriate level of performance to provide this functionality.

### 7.6.1.15 Threads

The basic assumption in this profile is that the system will consist of a single (implicit) process, with multiple threads. Therefore, all basic thread services are required, except for those related to multiple processes. The POSIX_THREADS_BASE unit of functionality was specified in this document instead of the _POSIX_THREADS option, because this option requires reader/writer locks, but this profile does not.

### 7.6.1.16 Tracing

Tracing is required for the PSE52 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.

The presence of a file system in the PSE52 profile facilitates the recording of the trace events, through the Trace Log option, which is required for this profile.
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_REGEXP and POSIX_SHELLFUNC 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, XSIDEVICE_SPECIFIC, XSI_FD_MGMT, XSI_FILE_SYSTEM, XSI_IPC, XSIJOB_CONTROL, XSI_JUMP, XSI_MATH, XSIMULTI_PROCESS, XSI_SIGNALS, XSI_SINGLE_PROCESS, XSI_SYSTEM_DATABASE, XSI_TIMERS, XSI_USERGROUPS, 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.
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 control 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 PSE52 environments.

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

The _XOPEN_STREAMS option provides facilities that are mainly related to networking, and thus are not required for PSE52 environments, as discussed above.

7.6.1.21 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 PSE52 applications.

7.6.1.22 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.

7.6.2 Shell and Utility Requirements

Because the Realtime Controller System Profile is intended for control systems which usually have no terminal or 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.
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.
Realtime Controller System Profile

74

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
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>
<thead>
<tr>
<th>Table 8-1: POSIX.1 Units of Functionality Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of Functionality</td>
</tr>
<tr>
<td>POSIX_C_LANG_JUMP</td>
</tr>
<tr>
<td>POSIX_C_LANG_MATH</td>
</tr>
<tr>
<td>POSIX_C_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_EVENT_MGMT</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
</tr>
<tr>
<td>POSIX_FILE_LOCKING</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
</tr>
<tr>
<td>POSIX_MULTI_PROCESS</td>
</tr>
<tr>
<td>POSIX_NETWORKING</td>
</tr>
<tr>
<td>POSIX_PIPE</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SIGNAL_JUMP</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
<tr>
<td>POSIX_THREADS_BASE</td>
</tr>
<tr>
<td>XSI_THREAD_MUTEX_EXT</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSI_THREADS_EXT</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_ASYNCHRONOUS_IO</td>
</tr>
<tr>
<td>_POSIX_CLOCK_SELECTION</td>
</tr>
<tr>
<td>_POSIX_CPUTIME</td>
</tr>
<tr>
<td>_POSIX_FSYNC</td>
</tr>
<tr>
<td>_POSIX_MAPPED_FILES</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK_RANGE</td>
</tr>
<tr>
<td>_POSIX_MEMORY_PROTECTION</td>
</tr>
<tr>
<td>_POSIX_MESSAGE_PASSING</td>
</tr>
<tr>
<td>_POSIX_MONOTONIC_CLOCK</td>
</tr>
<tr>
<td>_POSIX_NO_TRUNC</td>
</tr>
<tr>
<td>_POSIX_PRIORITIZED_IO</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>_POSIX_RAW_SOCKETS</td>
</tr>
<tr>
<td>_POSIX_REALTIME_SIGNALS</td>
</tr>
<tr>
<td>_POSIX_SEMAPHORES</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS</td>
</tr>
<tr>
<td>_POSIX_SPAWN</td>
</tr>
<tr>
<td>_POSIX_SPORADIC_SERVER</td>
</tr>
<tr>
<td>_POSIX_SYNCHRONIZED_IO</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADDR</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE</td>
</tr>
<tr>
<td>_POSIX_THREAD_CPUTIME</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIO_INHERIT</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIO_PROTECT</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESS_SHARED</td>
</tr>
<tr>
<td>_POSIX_THREAD_SPORADIC_SERVER</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS</td>
</tr>
<tr>
<td>_POSIX_TIMERS</td>
</tr>
<tr>
<td>_POSIX_TRACE</td>
</tr>
<tr>
<td>_POSIX_TRACE_EVENT_FILTER</td>
</tr>
<tr>
<td>_POSIX_TRACE_LOG</td>
</tr>
</tbody>
</table>

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 PSE53 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 PSE53 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 PSE53 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”.

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

The Dedicated 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>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_ADA_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_EVENT_MGMT</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
</tr>
<tr>
<td>POSIX_MULTI_PROCESS(^a)</td>
</tr>
<tr>
<td>POSIX_NETWORKING</td>
</tr>
<tr>
<td>POSIX_PIPE</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
</tbody>
</table>

\(^a\) The POSIX_MULTI_PROCESS unit of functionality shall be supported, with the provision that the package POSIX_Unsafe_Process_Primitives is not required.

The Dedicated Realtime System Profile implementation shall support the following options defined in POSIX.5c, by defining the associated option subtypes to have

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

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

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.
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.

8.3 Application Constraints

The Dedicated Realtime System profile defined in this standard requires only specific units of functionality of the required standards. The absence of particular elements of these standards introduces constraints on the use of some of the features of particular operations. This clause defines the constraints that an application strictly conforming to one of the profiles shall observe when using each of the operations required by that profile.

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

The sysconf() function has the following constraints:

(1) An application strictly conforming to the PSE53 profile shall not call the sysconf() function with the parameter _POSIX_VERSION since a meaningful value cannot be returned.¹

An application strictly conforming to PSE53 shall not call the kill() function with a negative argument because this profile does not require process group functionality.

An application strictly conforming to PSE53, 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.

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

An application strictly conforming to PSE53 shall not call the functions POSIX_Configurable_System_Limits.System_POSIX_Version or

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

Copyright © 2003 IEEE. All rights reserved. This is an unapproved IEEE Standards Draft, subject to change.
POSIX_Configurable_System_Limits.System_POSIX_Ada_Version, since
a meaningful value cannot be returned.¹

An application strictly conforming to PSE53 shall not call the subprograms con-

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

Implementations of PSE53 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.

8.4 Shell and Utility Requirements

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

8.5 Development Platform Requirements

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

8.5.1 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.

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 \texttt{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 \texttt{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,
and event management services. These applications require support for a simplified file system.

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

8.6.1.1 Process Primitives

Applications that correspond to the Dedicated Realtime System Environment are usually large embedded systems that require multiple processes for handling multiple, concurrent activities with independent address spaces. 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 in these systems.

8.6.1.2 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.

8.6.1.3 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 these systems require multiple processes, but not users or groups, the functions defined by the POSIX_MULTI_PROCESS unit of functionality are required.

### 8.6.1.4 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 PSE53 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.

The File Descriptor Management unit of functionality is included to aid the handling of file descriptors across the process creation and program execution operations.

### 8.6.1.5 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.

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

### 8.6.1.6 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.
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
priority ceiling options to reader/writer locks, which would eliminate the unbound-
ed priority inversion.

### 8.6.1.10 Priority Scheduling

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

A common requirement of realtime systems is that they be able to run threads or processes with real-time requirements together with threads with no real-time re-
quirements. 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 avail-
able 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 require-
ments 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
\([\text{max}_\text{FIFO}_\text{prio} \text{,} \text{max}_\text{FIFO}_\text{prio}-30]\) with SCHED_FIFO for real-time threads
(with a total of 31 priority levels), and then use the priority value
\(\min(\text{max}_\text{FIFO}_\text{prio}-31, \text{max}_\text{RR}_\text{prio})\) 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.

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

The current POSIX.1 specification allows implementations to support either sys-
tem-wide or process-wide contention scope, or both. This represents a compromise
that tries to address the requirements of both realtime and non-realtime applica-
tions, but introduces a potential source for non portability. Because the realtime
profiles are specifically targeted at realtime systems, the system-wide contention
scope option is required in the profiles that support multiple processes. Process-
wide contention scope may also be provided, perhaps for the non realtime threads of the application.

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.

8.6.1.11 Process Memory Locking

Realtime processes must be able to guarantee memory residency to reduce the latency for instruction fetches, data access, I/O operations, etc. The mechanism described in the POSIX.1 Process Memory Locking extension will satisfy this requirement.

8.6.1.12 Shared Memory

The Shared Memory Objects option provides the capability for more than one execution entity to share memory, without incurring the overhead of the shared memory object on permanent media. MemoryMapped I/O may be implemented using the Shared Memory facility. An implementation must provide facilities for creating a block of physical memory in which the application may place devices and facilities for binding to a user-provided pathname through which a device may subsequently be opened as a Shared Memory special file, and mapped into the process address space for the purpose of performing I/O or other functions from applications programs.

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.

8.6.1.13 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.

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

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

### 8.6.1.14 Message Passing

These realtime systems typically include some form of message queuing mechanism for communication among processes or threads. The POSIX.1 message passing offers an appropriate level of performance to provide this functionality.

### 8.6.1.15 Threads

The basic assumption in this profile is that the system will consist of one or more processes with multiple threads. Therefore, all thread services are required. The `_POSIX_THREADS_BASE` unit of functionality was specified in this document instead of the `_POSIX_THREADS` option, because this option requires reader/writer locks, but this profile does not.
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_REGEXP 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.
8.6.1.20 X/Open Units of Functionality and Options

Some XSI Units of Functionality (XSI_C_LANG_SUPPORT, XSIDEVICE_IO, XSIDEVICE_SPECIFIC, XSI_FDMGMT, XSI_FILESYSTEM, XSI_IPC, XSIJOB_CONTROL, XSI_JUMP, XSIMATH, XSI_MULTI_PROCESS, XSI_SIGNALS, XSI_SINGLE_PROCESS, XISYSTEM_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 interfaces 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.
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.
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>`:

```c
_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:

```c
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>
<thead>
<tr>
<th>Unit of Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX_C_LANG_JUMP</td>
</tr>
<tr>
<td>POSIX_C_LANG_MATH</td>
</tr>
<tr>
<td>POSIX_C_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_C_LANG_WIDE_CHAR</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_DEVICE_SPECIFIC</td>
</tr>
<tr>
<td>POSIX_EVENT_MGMT</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
</tr>
<tr>
<td>POSIX_FIFO</td>
</tr>
<tr>
<td>POSIX_FILE_ATTRIBUTES</td>
</tr>
<tr>
<td>POSIX_FILE_LOCKING</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM_EXT</td>
</tr>
<tr>
<td>POSIX_JOB_CONTROL</td>
</tr>
<tr>
<td>POSIX_MULTI_PROCESS</td>
</tr>
<tr>
<td>POSIX_NETWORKING</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_ADVISORY_INFO</td>
</tr>
<tr>
<td>_POSIX_ASYNCHRONOUS_IO</td>
</tr>
<tr>
<td>_POSIX_CHOWN_RESTRICTED</td>
</tr>
<tr>
<td>_POSIX_CLOCK_SELECTION</td>
</tr>
<tr>
<td>_POSIX_CPUTIME</td>
</tr>
<tr>
<td>_POSIX_FSYNC</td>
</tr>
<tr>
<td>_POSIX_JOB_CONTROL</td>
</tr>
<tr>
<td>_POSIX_MAPPED_FILES</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK_RANGE</td>
</tr>
<tr>
<td>_POSIX_MEMORY_PROTECTION</td>
</tr>
<tr>
<td>_POSIX_MESSAGE_PASSING</td>
</tr>
<tr>
<td>_POSIX_MONOTONIC_CLOCK</td>
</tr>
<tr>
<td>_POSIX_NO_TRUNC</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>_POSIX_RAW_SOCKETS</td>
</tr>
<tr>
<td>_POSIX_REALTIME_SIGNALS</td>
</tr>
<tr>
<td>_POSIX_REGEXP</td>
</tr>
<tr>
<td>_POSIX_SAVED_IDS</td>
</tr>
<tr>
<td>_POSIX_SEMAPHORES</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_SHELL</td>
</tr>
<tr>
<td>_POSIX_SPAWN</td>
</tr>
<tr>
<td>_POSIX_SPORADIC_SERVER</td>
</tr>
<tr>
<td>_POSIX_SYNCHRONIZED_IO</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADDRT</td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE</td>
</tr>
<tr>
<td>_POSIX_THREAD_CPUTIME</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIO_INHERIT</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIO_PROTECT</td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESS_SHARED</td>
</tr>
<tr>
<td>_POSIX_THREAD_SAFE_FUNCTIONS</td>
</tr>
<tr>
<td>_POSIX_THREAD_SPORADIC_SERVER</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS</td>
</tr>
<tr>
<td>_POSIX_TIMERS</td>
</tr>
<tr>
<td>_POSIX_TRACE</td>
</tr>
<tr>
<td>_POSIX_TRACE_EVENT_FILTER</td>
</tr>
<tr>
<td>_POSIX_TRACE_LOG</td>
</tr>
<tr>
<td>_POSIX_VDISABLE</td>
</tr>
</tbody>
</table>
9.2.2 POSIX.5c Requirements (Ada Language Option)

The Multi-Purpose 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>
<thead>
<tr>
<th>Table 9-3: POSIX.1 Units of Functionality Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of Functionality</td>
</tr>
<tr>
<td>POSIX_ADA_LANG_SUPPORT</td>
</tr>
<tr>
<td>POSIX_DEVICE_IO</td>
</tr>
<tr>
<td>POSIX_DEVICE_SPECIFIC</td>
</tr>
<tr>
<td>POSIX_EVENT_MGMT</td>
</tr>
<tr>
<td>POSIX_FD_MGMT</td>
</tr>
<tr>
<td>POSIX_FIFO</td>
</tr>
<tr>
<td>POSIX_FILE_ATTRIBUTES</td>
</tr>
<tr>
<td>POSIX_FILE_SYSTEM</td>
</tr>
<tr>
<td>POSIX_JOB_CONTROL</td>
</tr>
<tr>
<td>POSIX_MULTI_PROCESS</td>
</tr>
<tr>
<td>POSIX_NETWORKING</td>
</tr>
<tr>
<td>POSIX_PIPE</td>
</tr>
<tr>
<td>POSIX_SIGNALS</td>
</tr>
<tr>
<td>POSIX_SINGLE_PROCESS</td>
</tr>
<tr>
<td>POSIX_SYSTEM_DATABASE</td>
</tr>
<tr>
<td>POSIX_USER_GROUPS</td>
</tr>
</tbody>
</table>

The Multi-Purpose 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>
<thead>
<tr>
<th>Table 9-4: POSIX.5c Option Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX.5c Option</td>
</tr>
<tr>
<td>Asynchronous I/O</td>
</tr>
<tr>
<td>Change Owner Restriction</td>
</tr>
<tr>
<td>File Synchronization</td>
</tr>
<tr>
<td>Memory Mapped Files</td>
</tr>
<tr>
<td>Memory Locking</td>
</tr>
<tr>
<td>Memory Range Locking</td>
</tr>
<tr>
<td>Memory Protection</td>
</tr>
<tr>
<td>Message Queues</td>
</tr>
<tr>
<td>Filename Truncation</td>
</tr>
<tr>
<td>Prioritized I/O</td>
</tr>
<tr>
<td>Priority Process Scheduling</td>
</tr>
<tr>
<td>Realtime Signals</td>
</tr>
<tr>
<td>Saved IDs Support</td>
</tr>
<tr>
<td>Semaphores</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved. This is an unapproved IEEE Standards Draft, subject to change.
Table 9-4: POSIX.5c Option Requirements (Continued)

<table>
<thead>
<tr>
<th>POSIX.5c Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Memory Objects</td>
</tr>
<tr>
<td>Synchronized I/O</td>
</tr>
<tr>
<td>Mutexes Supported</td>
</tr>
<tr>
<td>Mutex Priority Inheritance</td>
</tr>
<tr>
<td>Mutex Priority Ceiling</td>
</tr>
<tr>
<td>Process Shared</td>
</tr>
<tr>
<td>Timers</td>
</tr>
</tbody>
</table>

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.
9.3 Application Constraints

The Multi-Purpose Realtime System profile defined in this standard requires only specific units of functionality of the required standards. The absence of particular elements of these standards introduces constraints on the use of some of the features of particular operations. This clause defines the constraints that an application strictly conforming to one of the profiles shall observe when using each of the operations required by that profile.

9.3.1 Constraints related to POSIX.1 Interfaces (C Language Option)

This profile has no constraints on the application related to POSIX.1 interfaces, because it requires the implementation to be POSIX.1 conforming.

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

An application strictly conforming to PSE54 shall not attempt to bind a signal to a task entry.

9.4 Shell and Utility Requirements

An implementation of the Multi-Purpose Realtime System Profile shall provide all the mandatory utilities in the Shell and Utilities volume of POSIX.1 with all the functional behavior described therein. The system shall support the Large File capabilities described in the Shell and Utilities volume of POSIX.1.

If the C Language Option is supported, the following options of the Shell and Utilities volume of POSIX.1 shall be supported:

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX2_C_BIND</td>
</tr>
<tr>
<td>POSIX2_CDEV</td>
</tr>
<tr>
<td>POSIX2_CHAR_TERM</td>
</tr>
<tr>
<td>POSIX2_FORT_RUN</td>
</tr>
<tr>
<td>POSIX2_SW_DEV</td>
</tr>
<tr>
<td>POSIX2_UPE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIX2_CHAR_TERM</td>
</tr>
<tr>
<td>POSIX2_FORT_RUN</td>
</tr>
<tr>
<td>POSIX2_SW_DEV</td>
</tr>
<tr>
<td>POSIX2_UPE</td>
</tr>
</tbody>
</table>

#### 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:
9.5.2.1 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>`.

9.6 Rationale for Operating System Requirements (informative)

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

9.6.1 Operating System Interface Requirements

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

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.
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.
The File Descriptor Management unit of functionality is included to aid the handling of file descriptors across the process creation and program execution operations.

### 9.6.1.5 Input and Output Primitives

The functions contained in the Device I/O unit of functionality are required to support I/O on devices, files, and special files.

Although asynchronous I/O can be easily implemented using threads dedicated to I/O, it is required in the PSE54 profile to support portability of applications that may have been developed before POSIX threads implementations were widely available.

### 9.6.1.6 Synchronized Input and Output

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

### 9.6.1.7 Device- and Class-Specific Functions

The terminal control functions are required for systems to support COTS applications and for the standard terminal devices that may be attached to the computer system. To support non-standard terminal devices, additional functions may be necessary.

### 9.6.1.8 System Databases, Users and Groups

The group and user database access functions are required for COTS database applications that may require them.

### 9.6.1.9 Synchronization

Mutexes and Condition Variables are required as part of threads model of concurrency.
Semaphores are required to synchronize a signal handler with some other process or thread. Semaphores are also 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 priority ceiling options to reader/writer locks, which would eliminate the unbounded priority inversion.

9.6.1.10 Priority Scheduling

This realtime environment requires the ability to do scheduling of concurrent processes and threads with a preemptive priority-based scheduler to ensure that hard deadlines are met. Thread and process priority scheduling are required for realtime applications. The Sporadic Server Scheduling option is also required for processes and threads, to enhance support of applications with aperiodic timing requirements. The POSIX_PRIORITY_RANGES unit of functionality is not required because its functions are already included in the required_POSIX_PRIORITY_SCHEDULING option.

A common requirement of realtime systems is that they be able to run threads or processes 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 avail-
able 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 $[\text{max_FIFO}_\text{prio}, \text{max_FIFO}_\text{prio}-30]$ with SCHED_FIFO for real-time threads (with a total of 31 priority levels), and then use the priority value $\min(\text{max_FIFO}_\text{prio}-31, \text{max_RR}_\text{prio})$ 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.

The implementation is required to support the PTHREAD_SCOPE_SYSTEM threading-scheduling contention scope. The contention scope of a thread defines the set of threads with which the thread competes for use of the processing resources. A thread created with PTHREAD_SCOPE_SYSTEM scheduling contention scope contends for resources with all other threads in the system that have the same scheduling allocation domain. This allows a consistent scheduling of threads across the system and therefore a predictable timing behavior. As a consequence, this is the preferred method for realtime systems.

The current POSIX.1 specification allows implementations to support either system-wide or process-wide contention scope, or both. This represents a compromise that tries to address the requirements of both realtime and non-realtime applications, but introduces a potential source for non portability. Because the realtime profiles are specifically targeted at realtime systems, the system-wide contention scope option is required in the profiles that support multiple processes. Process-wide contention scope may also be provided, perhaps for the non realtime threads of the application.

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.

9.6.1.11 Process Memory Locking

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

9.6.1.12 Shared Memory

The ability to share large volumes of data among many cooperating execution streams is required. The POSIX.1 Shared Memory extension provides this capability. Memory Mapped I/O may be implemented using the Shared Memory facility. An implementation must provide facilities for creating a block of physical memory in which the application may place devices and facilities for binding to a user-provided pathname through which a device may subsequently be opened as a Shared Memory special file, and mapped into the process address space for the purpose of performing I/O or other functions from applications programs.

The Memory Mapped Files is required 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, data can be manipulated as memory, and I/O data movement can be significantly reduced. 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 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.

9.6.1.13 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 \texttt{sleep()} and \texttt{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.

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

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

9.6.1.14 Message Passing

These realtime systems typically include some form of message queuing mechanism for communication among processes or threads. The POSIX.1 message passing offers an appropriate level of performance to provide this functionality.

9.6.1.15 Threads

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

9.6.1.16 Tracing

Tracing is required for the PSE54 environment because it 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 PSE54 profile requires general file system capabilities, the Trace Log option is required for this profile.
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_REGEXP 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.
The XSI_SYSTEM_LOGGING unit of functionality provides facilities for logging system activities, which are usually required in PSE54 environments. Therefore, this unit of functionality is required.

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 all PSE54 environments. It remains as an optional feature.

The _XOPEN_LEGACY option provides facilities for backwards compatibility that are not required in most PSE54 environments.

The _XOPEN_STREAMS option provides facilities that are not required in most PSE54 environments.

### 9.6.1.21 Language-Specific Services for the C Programming Language

Full support for the C Language standard is required in the C language option.

### 9.6.1.22 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.

### 9.6.2 Shell and Utility Requirements

The utilities and facilities described in the Shell and Utilities Volume of POSIX.1 are required in PSE54 environments.

### 9.6.3 Development Platform Requirements

The implementation is required to define a development environment in which a PSE54 application can be prepared for execution on the target platform. For this
profile, in most cases the development and the target platform roles will be combined in the same system.
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;
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>
<thead>
<tr>
<th>Table B-1: Functions under each POSIX.1 System Interface Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_ADVISORY_INFO</td>
</tr>
<tr>
<td><em>posix_fadvise</em>, <em>posix_fallocate</em>, <em>posix_memalign</em></td>
</tr>
<tr>
<td>_POSIX_ADVISORY_INFO and either _POSIX_MAPPED_FILES or</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS</td>
</tr>
<tr>
<td><em>posix_madvise</em></td>
</tr>
<tr>
<td>_POSIX_ASYNCHRONOUS_IO</td>
</tr>
<tr>
<td><em>aio_cancel</em>, <em>aio_error</em>, <em>aio_fsync</em>, <em>aio_read</em>, <em>aio_return</em>, <em>aio_suspend</em>,</td>
</tr>
<tr>
<td><em>aio_write</em>, <em>lio_listio</em></td>
</tr>
<tr>
<td>_POSIX_BARRIERS and _POSIX_THREADS</td>
</tr>
<tr>
<td><em>pthread_barrier_destroy</em>, <em>pthread_barrier_init</em>, <em>pthread_barrier_wait</em>,</td>
</tr>
<tr>
<td><em>pthread_barrierattr_destroy</em>, <em>pthread_barrierattr_init</em>,</td>
</tr>
<tr>
<td>_POSIX_BARRIERS, _POSIX_THREADS and _POSIX_THREAD_PROCESS_SHARED</td>
</tr>
<tr>
<td><em>pthread_barrierattr_getpshared</em>, <em>pthread_barrierattr_setpshared</em></td>
</tr>
<tr>
<td>_POSIX_CHOWN_RESTRICTED</td>
</tr>
<tr>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_CLOCK_SELECTION</td>
</tr>
<tr>
<td><em>clock_nanosleep</em></td>
</tr>
<tr>
<td>_POSIX_CLOCK_SELECTION and _POSIX_THREADS</td>
</tr>
<tr>
<td><em>pthread_condattr_getclock</em>, <em>pthread_condattr_setclock</em></td>
</tr>
<tr>
<td>_POSIX_CPUTIME</td>
</tr>
<tr>
<td><em>clock_getcpuclkid</em></td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>_POSIX_FSYNC</th>
<th>fsync()</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_IPV6</td>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_JOB_CONTROL</td>
<td>See the POSIX_JOB_CONTROL unit of functionality</td>
</tr>
<tr>
<td>_POSIX_MAPPED_FILES or _POSIX_SHARED_MEMORY_OBJECTS</td>
<td>mmap(), munmap()</td>
</tr>
<tr>
<td>_POSIX_MAPPED_FILES and _POSIX_SYNCHRONIZED_IO</td>
<td>msync()</td>
</tr>
<tr>
<td>_POSIX_MAPPED_FILES and _POSIX_ADVISORY_INFO</td>
<td>posix_madvise()</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK</td>
<td>mlockall(), munlockall()</td>
</tr>
<tr>
<td>_POSIX_MEMLOCK_RANGE</td>
<td>mlock(), munlock()</td>
</tr>
<tr>
<td>_POSIX_MEMORY_PROTECTION</td>
<td>mprotect()</td>
</tr>
<tr>
<td>_POSIX_MESSAGE_PASSING</td>
<td>mq_close(), mq_getattr(), mq_notify(), mq_open(), mq_receive(), mq_send(), mq_setattr(), mq_unlink(),</td>
</tr>
<tr>
<td>_POSIX_MESSAGE_PASSING and _POSIX_TIMEOUTS</td>
<td>mq_timedreceive(), mq_timedsend()</td>
</tr>
<tr>
<td>_POSIX_MONOTONIC_CLOCK</td>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_NO_TRUNC</td>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_PRIORITIZED_IO</td>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING</td>
<td>sched_get_priority_max(), sched_get_priority_min(), sched_getparam(), sched_getscheduler(), sched_rr_get_interval(), sched_setparam(), sched_setscheduler()</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING or _POSIX_THREADS</td>
<td>sched_yield()</td>
</tr>
<tr>
<td>_POSIX_PRIORITY_SCHEDULING and _POSIX_SPAWN</td>
<td>posix_spawnattr_getschedparam(), posix_spawnattr_setschedparam(), posix_spawnattr_getschedpolicy(), posix_spawnattr_setschedpolicy()</td>
</tr>
</tbody>
</table>
Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

<table>
<thead>
<tr>
<th>Function Option</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_RAW_SOCKETS</td>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_READER_WRITER_LOCKS</td>
<td>See the POSIX_RW_LOCKS unit of functionality</td>
</tr>
<tr>
<td>_POSIX_REALTIME_SIGNALS</td>
<td>sigqueue(), sigtimedwait(), sigwaitinfo()</td>
</tr>
<tr>
<td>_POSIX_REGEXP</td>
<td>See POSIX_REGEXP unit of functionality.</td>
</tr>
<tr>
<td>_POSIX_SAVED_IDS</td>
<td>No functions under this option</td>
</tr>
<tr>
<td>_POSIX_SEMAPHORES</td>
<td>sem_close(), sem_destroy(), sem_getvalue(), sem_init(), sem_open(), sem_post(), sem_trywait(), sem_wait(), sem_unlink()</td>
</tr>
<tr>
<td>_POSIX_SEMAPHORES and _POSIX_TIMEOUTS</td>
<td>sem_timedwait()</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS</td>
<td>shm_open(), shm_unlink()</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS and _POSIX_ADVISORY_INFO</td>
<td>posix_madvise()</td>
</tr>
<tr>
<td>_POSIX_SHARED_MEMORY_OBJECTS or _POSIX_MAPPED_FILES</td>
<td>mmap(), munmap()</td>
</tr>
<tr>
<td>_POSIX_SPAWN</td>
<td>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_spawnp()</td>
</tr>
<tr>
<td>_POSIX_SPAWN and _POSIX_PRIORITY_SCHEDULING</td>
<td>posix_spawnpattr_getschedparam(), posix_spawnpattr_setschedparam(), posix_spawnpattr_setschedpolicy(), posix_spawnpattr_setschedpolicy()</td>
</tr>
<tr>
<td>_POSIX_SPIN_LOCKS and _POSIX_THREADS</td>
<td>pthread_spin_destroy(), pthread_spin_init(), pthread_spin_lock(), pthread_spin_trylock(), pthread_spin_unlock()</td>
</tr>
<tr>
<td>_POSIX_SPORADIC_SERVER</td>
<td>No functions under this option</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
<table>
<thead>
<tr>
<th>Function</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_SYNCHRONIZED_IO</td>
<td></td>
</tr>
<tr>
<td>fdatasync()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_SYNCHRONIZED_IO and _POSIX_MAPPED_FILES</td>
<td></td>
</tr>
<tr>
<td>msync()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADDR and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstackaddr(), pthread_attr_setstackaddr()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADDR, _POSIX_THREADS and</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstack(), pthread_attr_setstack()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstacksize(), pthread_attr_setstacksize()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKSIZE, _POSIX_THREADS and</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_ATTR_STACKADDR</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstack(), pthread_attr_setstack()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_CPUTIME and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_getcpuunlockid()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIO_INHERIT and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_getprotocol(), pthread_mutexattr_setprotocol()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIO_PROTECT and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_getprioceiling(), pthread_mutex_setprioceiling(),</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_getprioceiling(), pthread_mutexattr_setprioceiling(),</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_getprioceiling(), pthread_mutexattr_setprioceiling(),</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_PRIORITY_SCHEDULING and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getinheritsched(), pthread_attr_getschedpolicy(),</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getscope(), pthread_attr_setscheduler(),</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setschedpolicy(), pthread_attr_setscope(),</td>
<td></td>
</tr>
<tr>
<td>pthread_getschedparam(), pthread_setschedparam(), pthread_setschedprio()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESS_SHARED and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_condattr_getpshared(), pthread_condattr_setpshared(),</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_getpshared(), pthread_mutexattr_setpshared()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESS_SHARED, _POSIX_BARRIERS and _POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_barrierattr_getpshared(), pthread_barrierattr_setpshared()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_PROCESS_SHARED, _POSIX_READER_WRITER_LOCKS and</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREADS</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_getpshared(), pthread_rwlockattr_setpshared()</td>
<td></td>
</tr>
<tr>
<td>_POSIX_THREAD_SAFE_FUNCTIONS</td>
<td></td>
</tr>
<tr>
<td>asctime_r(), ctime_r(), flockfile(), ftrylockfile(), funlockfile(), getc_unlocked(),</td>
<td></td>
</tr>
<tr>
<td>getchar_unlocked(), getgrgid_r(), getgrnam_r(), getlogin_r(), getpwuid_r(),</td>
<td></td>
</tr>
<tr>
<td>getpwnam_r(), gmtime_r(), localtime_r(), putc_unlocked(), putchar_unlocked(),</td>
<td></td>
</tr>
<tr>
<td>rand_r(), readdir_r(), strerror_r(), strtok_r(), ttyname_r()</td>
<td></td>
</tr>
</tbody>
</table>
Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

<table>
<thead>
<tr>
<th>_POSIX_THREAD_SPORADIC_SERVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>No functions under this option</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_atfork(),</td>
</tr>
<tr>
<td>pthread_attr_destroy(),</td>
</tr>
<tr>
<td>pthread_attr_getdetachstate(),</td>
</tr>
<tr>
<td>pthread_attr_getschedparam(),</td>
</tr>
<tr>
<td>pthread_attr_init(),</td>
</tr>
<tr>
<td>pthread_attr_setdetachstate(),</td>
</tr>
<tr>
<td>pthread_attr_setschedparam(),</td>
</tr>
<tr>
<td>pthread_cancel(),</td>
</tr>
<tr>
<td>pthread_cleanup_pop(),</td>
</tr>
<tr>
<td>pthread_cleanup_push(),</td>
</tr>
<tr>
<td>pthread_cond_broadcast(),</td>
</tr>
<tr>
<td>pthread_cond_destroy(),</td>
</tr>
<tr>
<td>pthread_cond_init(),</td>
</tr>
<tr>
<td>pthread_cond_signal(),</td>
</tr>
<tr>
<td>pthread_cond_timedwait(),</td>
</tr>
<tr>
<td>pthread_cond_wait(),</td>
</tr>
<tr>
<td>pthread_condattr_destroy(),</td>
</tr>
<tr>
<td>pthread_condattr_init(),</td>
</tr>
<tr>
<td>pthread_create(),</td>
</tr>
<tr>
<td>pthread_detach(),</td>
</tr>
<tr>
<td>pthread_equal(),</td>
</tr>
<tr>
<td>pthread_exit(),</td>
</tr>
<tr>
<td>pthread_getspecific(),</td>
</tr>
<tr>
<td>pthread_join(),</td>
</tr>
<tr>
<td>pthread_key_create(),</td>
</tr>
<tr>
<td>pthread_key_delete(),</td>
</tr>
<tr>
<td>pthread_kill(),</td>
</tr>
<tr>
<td>pthread_mutex_destroy(),</td>
</tr>
<tr>
<td>pthread_mutex_init(),</td>
</tr>
<tr>
<td>pthread_mutex_lock(),</td>
</tr>
<tr>
<td>pthread_mutex_trylock(),</td>
</tr>
<tr>
<td>pthread_mutexUnlock(),</td>
</tr>
<tr>
<td>pthread_mutexattr_destroy(),</td>
</tr>
<tr>
<td>pthread_mutexattr_init(),</td>
</tr>
<tr>
<td>pthread_once(),</td>
</tr>
<tr>
<td>pthread_self(),</td>
</tr>
<tr>
<td>pthread_setspecific(),</td>
</tr>
<tr>
<td>pthread_setcancelstate(),</td>
</tr>
<tr>
<td>pthread_setcanceltype(),</td>
</tr>
<tr>
<td>pthread_setspecific(),</td>
</tr>
<tr>
<td>pthread_sigmask(),</td>
</tr>
<tr>
<td>pthread_etcancel()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and _POSIX_CLOCK_SELECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_condattr_setclock()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and _POSIX_BARRIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_barrier_destroy(),</td>
</tr>
<tr>
<td>pthread_barrier_init(),</td>
</tr>
<tr>
<td>pthread_barrier_wait(),</td>
</tr>
<tr>
<td>pthread_barrierattr_destroy(),</td>
</tr>
<tr>
<td>pthread_barrierattr_init(),</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS, _POSIX_BARRIERS and _POSIX_THREAD_PROCESS_SHARED</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_barrierattr_getpshared(), pthread_barrierattr_setpshared()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and _POSIX_SPIN_LOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_spin_destroy(),</td>
</tr>
<tr>
<td>pthread_spin_init(),</td>
</tr>
<tr>
<td>pthread_spin_lock(),</td>
</tr>
<tr>
<td>pthread_spin_trylock(),</td>
</tr>
<tr>
<td>pthread_spin_unlock()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and _POSIX_THREAD_ATTR_STACKADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getstackaddr(),</td>
</tr>
<tr>
<td>pthread_attr_setstackaddr()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS, _POSIX_THREAD_ATTR_STACKADDR and _POSIX_THREAD_ATTR_STACKSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getstack(), pthread_attr_getstacksize()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and _POSIX_THREAD_ATTR_STACKSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getstacksize(), pthread_attr_setstacksize()^a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and _POSIX_THREAD_CPUTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_getcpuclockid()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>_POSIX_THREADS and either _POSIX_THREAD_PRIO_INHERIT or _POSIX_THREAD_PRIO_PROTECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_mutexattr_getprotocol(), pthread_mutexattr_setprotocol()</td>
</tr>
</tbody>
</table>

This table row continued on next page...
Table B-1: Functions under each POSIX.1 System Interface Option (Continued)

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_THREAD_PRIO_PROTECT and _POSIX_THREADS</td>
</tr>
<tr>
<td>pthread_mutex_getprioceiling(), pthread_mutex_setprioceiling(),</td>
</tr>
<tr>
<td>pthread_mutexattr_getprioceiling(), pthread_mutexattr_setprioceiling()</td>
</tr>
<tr>
<td>_POSIX_THREADS and _POSIX_THREAD_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>pthread_attr_getinheritsched(), pthread_attr_getschedpolicy(),</td>
</tr>
<tr>
<td>pthread_attr_getscope(), pthread_attr_setinheritsched(),</td>
</tr>
<tr>
<td>pthread_attr_setschedpolicy(), pthread_attr_setscope(),</td>
</tr>
<tr>
<td>pthread_getschedparam(), pthread_setschedparam(), pthread_setschedprio()</td>
</tr>
<tr>
<td>_POSIX_THREADS and _POSIX_THREAD_PROCESS_SHARED</td>
</tr>
<tr>
<td>pthread_condattr_getpshared(), pthread_condattr_setpshared(),</td>
</tr>
<tr>
<td>pthread_mutexattr_getpshared(), pthread_mutexattr_setpshared()</td>
</tr>
<tr>
<td>_POSIX_THREADS, _POSIX_THREAD_PROCESS_SHARED and _POSIX_READER_WRITER_LOCKS</td>
</tr>
<tr>
<td>pthread_rwlockattr_getpshared(), pthread_rwlockattr_setpshared()</td>
</tr>
<tr>
<td>_POSIX_THREADS and _POSIX_THREADS_PROCESS_SHARED and _POSIX_TIMEOUTS</td>
</tr>
<tr>
<td>pthread_mutex_timedlock()</td>
</tr>
<tr>
<td>_POSIX_THREADS, _POSIX_TIMEOUTS and _POSIX_READER_WRITER_LOCKS</td>
</tr>
<tr>
<td>pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock()</td>
</tr>
<tr>
<td>_POSIX_THREADS and _POSIX_READER_WRITER_LOCKS</td>
</tr>
<tr>
<td>pthread_rwlock_destroy(), pthread_rwlock_init(), pthread_rwlock_rdlock(),</td>
</tr>
<tr>
<td>pthread_rwlock_trdlock(), pthread_rwlock_trwlock(),</td>
</tr>
<tr>
<td>pthread_rwlock_unlock(), pthread_rwlock_unrlock(),</td>
</tr>
<tr>
<td>pthread_rwlockattr_init(), pthread_rwlockattr_destroy()</td>
</tr>
<tr>
<td>_POSIX_THREADS or _POSIX_PRIORITY_SCHEDULING</td>
</tr>
<tr>
<td>sched_yield()</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS and _POSIX_MESSAGE_PASSING</td>
</tr>
<tr>
<td>mq_timedreceive(), mq_timedsend()</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS, _POSIX_THREADS, and _POSIX_READER_WRITER_LOCKS</td>
</tr>
<tr>
<td>pthread_rwlock_timedrdlock(), pthread_rwlock_timedwrlock()</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS and _POSIX_SEMAPHORES</td>
</tr>
<tr>
<td>sem_timedwait()</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS and _POSIX_THREADS</td>
</tr>
<tr>
<td>pthread_mutex_timedlock()</td>
</tr>
<tr>
<td>_POSIX_TIMEOUTS and _POSIX_TRACE</td>
</tr>
<tr>
<td>posix_trace_timedgetnext_event()</td>
</tr>
<tr>
<td>_POSIX_TIMERS</td>
</tr>
</tbody>
</table>
| clock_getres(), clock_gettime(), clock_settime(), nanosleep(), timer_create(),
| timer_delete(), timer_gettime(), timer_getoverrun(), timer_settime()     |
Table B-1: Functions under each POSIX.1
System Interface Option (Continued)

<table>
<thead>
<tr>
<th>Function and Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>_POSIX_TRACE</strong></td>
</tr>
<tr>
<td>posix_trace_attr_destroy(),</td>
</tr>
<tr>
<td>posix_trace_attr_getclockres(),</td>
</tr>
<tr>
<td>posix_trace_attr_getcreateetime(),</td>
</tr>
<tr>
<td>posix_trace_attr_getgenversion(),</td>
</tr>
<tr>
<td>posix_trace_attr_getstreamfullpolicy(),</td>
</tr>
<tr>
<td>posix_trace_attr_getmaxdatasize(),</td>
</tr>
<tr>
<td>posix_trace_attr_getmaxsystemevents(),</td>
</tr>
<tr>
<td>posix_trace_attr_getmaxuserevents(),</td>
</tr>
<tr>
<td>posix_trace_attr_getstreamsize(),</td>
</tr>
<tr>
<td>posix_trace_attr_init(),</td>
</tr>
<tr>
<td>posix_trace_attr_setname(),</td>
</tr>
<tr>
<td>posix_trace_attr_setstreamfullpolicy(),</td>
</tr>
<tr>
<td>posix_trace_attr_setmaxdatasize(),</td>
</tr>
<tr>
<td>posix_trace_attr_setmaxstreamsize(),</td>
</tr>
<tr>
<td>posix_trace_attr_setlogfullpolicy(),</td>
</tr>
<tr>
<td>posix_trace_attr_setlogsize(),</td>
</tr>
<tr>
<td>posix_trace_close(),</td>
</tr>
<tr>
<td>posix_trace_open(),</td>
</tr>
<tr>
<td>posix_trace_rewind(),</td>
</tr>
<tr>
<td>posix_trace_create_withlog(),</td>
</tr>
<tr>
<td>posix_trace_flush()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE and _POSIX_TIMEOUTS</strong></td>
</tr>
<tr>
<td>posix_trace_timedgetnext_event()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE and _POSIX_TRACE_INHERIT</strong></td>
</tr>
<tr>
<td>posix_trace_attr_getinherited()</td>
</tr>
<tr>
<td>posix_trace_attr_setinherited()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE and _POSIX_TRACE_LOG</strong></td>
</tr>
<tr>
<td>posix_trace_getlogfullpolicy()</td>
</tr>
<tr>
<td>posix_trace_getlogsize()</td>
</tr>
<tr>
<td>posix_trace_setlogfullpolicy()</td>
</tr>
<tr>
<td>posix_trace_setlogsize()</td>
</tr>
<tr>
<td>posix_trace_close()</td>
</tr>
<tr>
<td>posix_trace_open()</td>
</tr>
<tr>
<td>posix_trace_rewind()</td>
</tr>
<tr>
<td>posix_trace_create_withlog()</td>
</tr>
<tr>
<td>posix_trace_flush()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE and _POSIX_TRACE_EVENT_FILTER</strong></td>
</tr>
<tr>
<td>posix_trace_eventset_add()</td>
</tr>
<tr>
<td>posix_trace_eventset_del()</td>
</tr>
<tr>
<td>posix_trace_eventset_empty()</td>
</tr>
<tr>
<td>posix_trace_eventset_fill()</td>
</tr>
<tr>
<td>posix_trace_eventset_ismember()</td>
</tr>
<tr>
<td>posix_trace_get_filter()</td>
</tr>
<tr>
<td>posix_trace_set_filter()</td>
</tr>
<tr>
<td>posix_trace_get_graphid()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE_EVENT_FILTER and _POSIX_TRACE</strong></td>
</tr>
<tr>
<td>posix_trace_eventset_add()</td>
</tr>
<tr>
<td>posix_trace_eventset_del()</td>
</tr>
<tr>
<td>posix_trace_eventset_empty()</td>
</tr>
<tr>
<td>posix_trace_eventset_fill()</td>
</tr>
<tr>
<td>posix_trace_eventset_ismember()</td>
</tr>
<tr>
<td>posix_trace_get_filter()</td>
</tr>
<tr>
<td>posix_trace_set_filter()</td>
</tr>
<tr>
<td>posix_trace_get_graphid()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE_INHERIT and _POSIX_TRACE</strong></td>
</tr>
<tr>
<td>posix_trace_attr_getinherited()</td>
</tr>
<tr>
<td>posix_trace_attr_setinherited()</td>
</tr>
<tr>
<td><strong>_POSIX_TRACE_LOG and _POSIX_TRACE</strong></td>
</tr>
<tr>
<td>posix_trace_getlogfullpolicy()</td>
</tr>
<tr>
<td>posix_trace_getlogsize()</td>
</tr>
<tr>
<td>posix_trace_setlogfullpolicy()</td>
</tr>
<tr>
<td>posix_trace_setlogsize()</td>
</tr>
<tr>
<td>posix_trace_close()</td>
</tr>
<tr>
<td>posix_trace_open()</td>
</tr>
<tr>
<td>posix_trace_rewind()</td>
</tr>
<tr>
<td>posix_trace_create_withlog()</td>
</tr>
<tr>
<td>posix_trace_flush()</td>
</tr>
<tr>
<td><strong>_POSIX_TYPED_MEMORY_OBJECTS</strong></td>
</tr>
<tr>
<td>posix_mem_offset()</td>
</tr>
<tr>
<td>posix_typed_mem_get_info()</td>
</tr>
<tr>
<td>posix_typed_mem_open()</td>
</tr>
<tr>
<td><strong>_POSIX_VDISABLE</strong></td>
</tr>
<tr>
<td>No functions under this option</td>
</tr>
</tbody>
</table>

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)

| **_XOPEN_CRYPT** | crypt(), encrypt(), setkey() |
| **_XOPEN_ENH_I18N** | No functions under this option |
| **_XOPEN_LEGACY** | bcmp(), bcopy(), bzero(), ecvt(), fcvt(), ftime(), gevt(), getwd(), index(), mktemp(), rindex(), utimes(), wcsves() |
| **_XOPEN_REALTIME** | This Option Group consists of the set of the following options from within POSIX.1: _POSIX_ASYNCHRONOUS_IO, _POSIX_FSYNC, _POSIX_MAPPED_FILES, _POSIX_MEMLOCK, _POSIX_MEMLOCK_RANGE, _POSIX_MEMORY_PROTECTION, _POSIX_MESSAGE_PASSING, _POSIX_PRIORITIZED_IO, _POSIX_PRIORITY_SCHEDULING, _POSIX_REALTIME_SIGNALS, _POSIX_SEMAPHORES, _POSIX_SHARED_MEMORY_OBJECTS, _POSIX_SYNCHRONIZED_IO, _POSIX_TIMERS |
| **_XOPEN_REALTIME_THREADS** | This Option Group consists of the set of the following options from within POSIX.1: _POSIX_THREAD_PRIO_INHERIT, _POSIX_THREAD_PRIO_PROTECT, _POSIX_THREAD_PRIORITY_SCHEDULING |
| **_XOPEN_SHM** | This option is included in the XSI_IPC unit of functionality |
| **_XOPEN_STREAMS** | fattach(), fdetach(), getmsg(), getpmsg(), ioctl(), isastream(), putmsg(), putpmsg() |
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

<table>
<thead>
<tr>
<th>Option</th>
<th>Utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX_SHELL</td>
<td>sh</td>
</tr>
<tr>
<td>_POSIX_C_BIND</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_C_DEV</td>
<td>c99, lex, yacc</td>
</tr>
<tr>
<td>_POSIX2_CHAR_TERM</td>
<td>No utilities under this option</td>
</tr>
</tbody>
</table>

---

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
### Table B-2: Utilities under each POSIX.1 Shell and Utilities Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>_POSIX2_FORT_DEV</td>
<td>fort77</td>
</tr>
<tr>
<td>_POSIX2_FORT_RUN</td>
<td>asa</td>
</tr>
<tr>
<td>_POSIX2_LOCALEDEF</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_PBS</td>
<td>qalter, qdel, qhold, qmove, qmsg, qrerun, qrls, qselect, qsig, qstat, qsub</td>
</tr>
<tr>
<td>_POSIX2_PBS_ACCOUNTING</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_PBS_CHECKPOINT</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_PBS_LOCATE</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_PBS_MESSAGE</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_PBS_TRACK</td>
<td>No utilities under this option</td>
</tr>
<tr>
<td>_POSIX2_SW_DEV</td>
<td>ar, make, strip</td>
</tr>
<tr>
<td>_POSIX2_SW_DEV and _POSIX2_UPE</td>
<td>nm</td>
</tr>
<tr>
<td>_POSIX2_UPE</td>
<td>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</td>
</tr>
<tr>
<td>_POSIX2_UPE and _POSIX2_SW_DEV</td>
<td>nm</td>
</tr>
</tbody>
</table>
B.2 POSIX.5c Options

The following table shows the subprograms included under each of the options specified in POSIX.5c:

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprogram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous I/O</td>
<td>All except the two subprograms below</td>
</tr>
<tr>
<td>POSIX_Asynchonous_IO</td>
<td></td>
</tr>
<tr>
<td>Asynchronous I/O and Synchronized I/O</td>
<td>Synchronize_Data</td>
</tr>
<tr>
<td>POSIX_Asynchonous_IO</td>
<td>Synchronize_File</td>
</tr>
<tr>
<td>Change Owner Restriction</td>
<td>None</td>
</tr>
<tr>
<td>File Synchronization</td>
<td>Synchronize_File</td>
</tr>
<tr>
<td>POSIX_IO</td>
<td></td>
</tr>
<tr>
<td>Filename Truncation</td>
<td>None</td>
</tr>
<tr>
<td>Memory Mapped Files or Shared Memory Objects</td>
<td></td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Change_Permissions</td>
</tr>
<tr>
<td>POSIX_Memory_Mapping</td>
<td>Truncate_File</td>
</tr>
<tr>
<td></td>
<td>Map_Memory^a</td>
</tr>
<tr>
<td></td>
<td>Unmap_Memory</td>
</tr>
<tr>
<td>Memory Mapped Files and Synchronized I/O</td>
<td>Synchronize_Memory</td>
</tr>
<tr>
<td>POSIX_Memory_Mapping</td>
<td></td>
</tr>
<tr>
<td>Memory Locking</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Memory_Locking</td>
<td></td>
</tr>
<tr>
<td>Memory Protection</td>
<td>Change_Protection</td>
</tr>
<tr>
<td>POSIX_Memory_Mapping</td>
<td></td>
</tr>
<tr>
<td>Memory Range Locking</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Memory_Range_Locking</td>
<td></td>
</tr>
<tr>
<td>Message Queues</td>
<td>All</td>
</tr>
<tr>
<td>POSIX_Message_ Queues</td>
<td></td>
</tr>
</tbody>
</table>
### Table B-3: Packages and Subprograms under each POSIX.5c Option (Continued)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprogram</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mutexes</strong></td>
<td>All except the subprograms below</td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td></td>
</tr>
<tr>
<td>POSIX_Condition_Variables</td>
<td></td>
</tr>
<tr>
<td><strong>Mutexes and Process Shared</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td>Get_Process_Shared</td>
</tr>
<tr>
<td></td>
<td>Set_Process_Shared</td>
</tr>
<tr>
<td>POSIX_Condition_Variables</td>
<td>Get_Process_Shared</td>
</tr>
<tr>
<td></td>
<td>Set_Process_Shared</td>
</tr>
<tr>
<td><strong>Mutexes and MutexPriority Ceiling</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td>Set_Ceiling_Priority&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Get_Ceiling_Priority&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>Mutexes and either Mutex Priority Inheritance or MutexPriority Ceiling</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td>Set_Locking_Policy</td>
</tr>
<tr>
<td></td>
<td>Get_Locking_Policy</td>
</tr>
<tr>
<td><strong>Mutex Priority Ceiling and Mutexes</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td>Set_Ceiling_Priority&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Get_Ceiling_Priority&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Set_Locking_Policy</td>
</tr>
<tr>
<td></td>
<td>Get_Locking_Policy</td>
</tr>
<tr>
<td><strong>Mutex Priority Inheritance and Mutexes</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td>Set_Locking_Policy</td>
</tr>
<tr>
<td></td>
<td>Get_Locking_Policy</td>
</tr>
<tr>
<td><strong>Network Management and Sockets Detailed Network Interface</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Sockets</td>
<td>Set_Flags</td>
</tr>
<tr>
<td></td>
<td>Get_Flags</td>
</tr>
<tr>
<td></td>
<td>Set_Family</td>
</tr>
<tr>
<td></td>
<td>Get_Family</td>
</tr>
<tr>
<td></td>
<td>Set_Socket_Type</td>
</tr>
<tr>
<td></td>
<td>Get_Socket_Type</td>
</tr>
<tr>
<td></td>
<td>Set_Protocol_Number</td>
</tr>
<tr>
<td></td>
<td>Get_Protocol_Number</td>
</tr>
<tr>
<td></td>
<td>Get_Canonical_Name</td>
</tr>
<tr>
<td></td>
<td>Get_Socket_Address_Info</td>
</tr>
<tr>
<td></td>
<td>Get_Socket_Address_Info</td>
</tr>
<tr>
<td></td>
<td>For_Every_Item</td>
</tr>
<tr>
<td><strong>Poll</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Event_Management</td>
<td>Get_File</td>
</tr>
<tr>
<td></td>
<td>Set_File</td>
</tr>
<tr>
<td></td>
<td>Get_Events</td>
</tr>
<tr>
<td></td>
<td>Set_Events</td>
</tr>
<tr>
<td></td>
<td>Get_Returned_Events</td>
</tr>
<tr>
<td></td>
<td>Set_Returned_Events</td>
</tr>
<tr>
<td></td>
<td>Poll</td>
</tr>
<tr>
<td><strong>Prioritized I/O</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved. This is an unapproved IEEE Standards Draft, subject to change.
### Table B-3: Packages and Subprograms under each POSIX.5c Option (Continued)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprogram</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Priority Process Scheduling</strong></td>
<td><strong>All</strong></td>
</tr>
<tr>
<td>POSIX.Process_Scheduling</td>
<td></td>
</tr>
<tr>
<td><strong>Process Shared and Mutexes</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Mutexes</td>
<td>Get_Process_Shared</td>
</tr>
<tr>
<td></td>
<td>Set_Process_Shared</td>
</tr>
<tr>
<td>POSIX.Condition_Variables</td>
<td>Get_Process_Shared</td>
</tr>
<tr>
<td></td>
<td>Set_Process_Shared</td>
</tr>
<tr>
<td><strong>Realtime Signals</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Signals</td>
<td>Enable_Queueing</td>
</tr>
<tr>
<td></td>
<td>Disable_Queueing</td>
</tr>
<tr>
<td></td>
<td>Await_Signal&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Await_Signal_Or_Timeout&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>Queue_Signal</td>
</tr>
<tr>
<td><strong>Saved IDs Support</strong></td>
<td><strong>None</strong></td>
</tr>
<tr>
<td><strong>Select</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Event_Management</td>
<td>Add</td>
</tr>
<tr>
<td></td>
<td>Remove</td>
</tr>
<tr>
<td></td>
<td>In_Set</td>
</tr>
<tr>
<td></td>
<td>Select_File&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>Semaphores</strong></td>
<td></td>
</tr>
<tr>
<td>POSIX_Semaphores</td>
<td><strong>All</strong></td>
</tr>
<tr>
<td><strong>Shared Memory Objects</strong></td>
<td><strong>All</strong></td>
</tr>
<tr>
<td>POSIX_Shared_Memory_Objects</td>
<td></td>
</tr>
<tr>
<td>POSIX_Generic_Shared_Memory</td>
<td></td>
</tr>
<tr>
<td><strong>Shared Memory Objects and Memory Range Locking</strong></td>
<td><strong>Lock_Shared_Memory</strong></td>
</tr>
<tr>
<td>POSIX_Generic_Shared_Memory</td>
<td>Unlock_Shared_Memory</td>
</tr>
<tr>
<td><strong>Shared Memory Objects or Memory Mapped Files</strong></td>
<td><strong>Truncate_File</strong></td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Truncate_File</td>
</tr>
</tbody>
</table>
Table B-3: Packages and Subprograms under each POSIX.5c Option (Continued)

<table>
<thead>
<tr>
<th>Package</th>
<th>Subprogram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sockets Detailed Network Interface</td>
<td></td>
</tr>
<tr>
<td>POSIX_Sockets</td>
<td>All except the subprograms below</td>
</tr>
<tr>
<td>Sockets Detailed Network Interface and Network Management</td>
<td></td>
</tr>
<tr>
<td>POSIX_Sockets</td>
<td>Set_Flags</td>
</tr>
<tr>
<td></td>
<td>Get_Flags</td>
</tr>
<tr>
<td></td>
<td>Set_Family</td>
</tr>
<tr>
<td></td>
<td>Get_Family</td>
</tr>
<tr>
<td></td>
<td>Set_Socket_Type</td>
</tr>
<tr>
<td></td>
<td>Get_Socket_Type</td>
</tr>
<tr>
<td></td>
<td>Set_Protocol_Number</td>
</tr>
<tr>
<td></td>
<td>Get_Protocol_Number</td>
</tr>
<tr>
<td></td>
<td>Get_Canonical_Name</td>
</tr>
<tr>
<td></td>
<td>Get_Socket_Address_Info</td>
</tr>
<tr>
<td></td>
<td>Get_Socket_Address_Info</td>
</tr>
<tr>
<td></td>
<td>For_Every_Item</td>
</tr>
<tr>
<td>Synchronized I/O</td>
<td></td>
</tr>
<tr>
<td>POSIX_IO</td>
<td>Synchronize_Data</td>
</tr>
<tr>
<td>Synchronized I/O and Memory Mapped Files</td>
<td></td>
</tr>
<tr>
<td>POSIX_Memory_Mapping</td>
<td>Synchronize_Memory</td>
</tr>
<tr>
<td>Timers</td>
<td></td>
</tr>
<tr>
<td>POSIX_Timers</td>
<td>All</td>
</tr>
<tr>
<td>XTI Detailed Network Interface</td>
<td></td>
</tr>
<tr>
<td>POSIX_XTI</td>
<td>All</td>
</tr>
</tbody>
</table>

a. All versions
b. Return type Signal_Info
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


{B2} ISO/IEC 10646:..., Information processing—Multiple octet coded character set.

{B3} IEEE Std 100-1988, IEEE Standard Dictionary of Electrical and Electronics Terms.


C.2 Other Documents

## Alphabetic Topical Index

### Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;limits.h&gt;</code> header</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td><code>&lt;unistd.h&gt;</code> header</td>
<td>36, 39, 40, 47, 57, 58, 63, 64, 75, 76, 82, 93, 94, 100, 101</td>
<td></td>
</tr>
<tr>
<td><code>_Exit()</code> function</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td><code>_exit()</code> function</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td><code>_longjmp()</code> function</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_ADVISORY_INFO</code> option</td>
<td>18, 20, 49, 66, 84, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_format</code> option</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_option</code></td>
<td>39, 57, 75, 93</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_CONTROLLER</code> option</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIMEDEDICATED</code> option</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_LANG_Ada95</code> option</td>
<td>40, 47, 58, 64, 76, 82, 94, 101</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_LANG_C99</code> option</td>
<td>40, 47, 58, 63, 76, 82, 94, 100</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_MINIMAL</code> option</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_AEP_REALTIME_MULTI</code> option</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td><code>_POSIXASYNCIO</code> option</td>
<td>18, 20, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_BARRIERS</code> option</td>
<td>18, 20</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_CHOWN_RESTRICTED</code> option</td>
<td>18, 20, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_CLOCK_SELECTION</code> option</td>
<td>18, 20, 41, 59, 77, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_CPU_TIME</code> option</td>
<td>18, 20, 77, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_FSYNC</code> option</td>
<td>18, 20, 41, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_IPV6</code> option</td>
<td>18, 20</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_JOB_CONTROL</code> option</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_JOB_CONTROL</code> option</td>
<td>18, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_MAPPED_FILES</code> option</td>
<td>18, 20, 49, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_MEMLOCK</code> option</td>
<td>18, 20, 41, 48, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_MEMLOCK_RANGE</code> option</td>
<td>18, 20, 41, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_MEMORY_PROTECTION</code> option</td>
<td>18, 20, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_MESSAGE_PASSING</code> option</td>
<td>18, 20, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_MONOTONIC_CLOCK</code> option</td>
<td>19, 20, 41, 59, 77, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_NGROUPS_MAX</code> limit</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_NO_TRUNC</code> option</td>
<td>19, 20, 22, 41, 49, 59, 77, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_PRIORITIZED_IO</code> option</td>
<td>19, 20, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_PRIORITY_SCHEDULING</code> option</td>
<td>19, 20, 77, 86, 95, 104, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_RAW_SOCKETS</code> option</td>
<td>19, 20, 77, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_READER_WRITER_LOCKS</code> option</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_READER_WRITER_LOCKS</code> option</td>
<td>19, 20</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_REALTIME_SIGNALS</code> option</td>
<td>19, 20, 41, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_REGEXP</code> option</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_REGEXP</code> option</td>
<td>19, 21, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_RT_SIG_MAX</code> limit</td>
<td>41, 59, 77, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SAVED_IDS</code> option</td>
<td>19, 21, 95</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SEMAPHORES</code> option</td>
<td>19, 21, 41, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SHARED_MEMORY_OBJECTS</code> option</td>
<td>19, 21, 41, 59, 77, 95, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SHELL</code> option</td>
<td>19, 21, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SPAWN</code> option</td>
<td>19, 21, 77, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SPIN_LOCKS</code> option</td>
<td>19, 21</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SPOREDIC_SERVER</code> option</td>
<td>19, 21, 77, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_SYNCHRONIZED_IO</code> option</td>
<td>19, 21, 41, 59, 77, 96, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_THREAD_ATTR_STACKADDR</code> option</td>
<td>19, 21, 41, 59, 77, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_THREAD_ATTR_STACKSIZE</code> option</td>
<td>19, 21, 41, 59, 77, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_THREAD_CPU_TIME</code> option</td>
<td>19, 21, 41, 59, 77, 96</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_THREAD_PRIO_INHERIT</code> option</td>
<td>19, 21, 41, 59, 77, 96, 120</td>
<td></td>
</tr>
<tr>
<td><code>_POSIX_THREAD_PRIO_PROTECT</code> option</td>
<td>19, 21, 41, 59, 77, 96, 120</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Application Platform... 26
ar utility... 122
Argument_List subprogram... 16
asa utility... 122
asctime() function... 6
asctime_r() function... 6, 116
asin() function... 5
asinf() function... 5
asinh() function... 5
asinhf() function... 5
asinl() function... 5
assert() function... 7
Asyncronous I/O option... 20, 79, 97, 123
at utility... 122
atan() function... 5
atan2() function... 5
atan2f() function... 5
atanf() function... 5
atanh() function... 5
atanhf() function... 5
atanhl() function... 5
atanl() function... 5
atexit() function... 7
atof() function... 6
atoi() function... 6
 atol() function... 6
atoll() function... 6
Await_Signal subprogram... 15, 125
Await_Signal_Or_Timeout subprogram... 15, 125

B

Base Standard... 26
base standard... 26
basename() function... 9
batch utility... 122
bcmp() function... 120
bcopy() function... 120
bg utility... 122
Bibliography... 127
bind() function... 7
Bits_Per_Character_Of subprogram... 11
Block_Signals subprogram... 15
Blocked_Signals subprogram... 15
Blocking_Behavior constant... 22
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
cacoshl() function... 5
cacosf() function... 5
calloc() function... 6
carg() function... 5
cargf() function... 5
cargl() function... 5
casin() function... 5
casin() function... 5
casinl() function... 5
cat() function... 5
catan() function... 5
catan() function... 5
catanf() function... 5
catanh() function... 5
catanh() function... 5
catanhf() function... 5
catanhf() function... 5
catanhl() function... 5
catanhl() function... 5
cat() function... 5
cbrt() function... 5
cbrtf() function... 5
cbrtl() function... 5
cos() function... 5
cosf() function... 5
cosh() function... 5
coshf() function... 5
coshl() function... 5
cosl() function... 5
ceil() function... 5

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
ceilf() function... 5
ceil() function... 5
cexp() function... 5
cexpf() function... 5
cexpl() function... 5
cfgetispeed() function... 7
cfgetospeed() function... 7
cfsetispeed() function... 7
cfsetospeed() function... 7

Change Owner Restriction option... 20, 97, 123
Change_Owner_And_Group subprogram... 12
Change_Permissions subprogram... 12, 123
Change_Protection subprogram... 123
Change_Working_Directory subprogram... 13

chdir() function... 7
CHILD_MAX limit... 96
CHILD_MAX option... 44, 62
echmod() function... 7
echown() function... 7
cimag() function... 5
cimagf() function... 5
cimagl() function... 5
CONSTANT

Clear_Environment subprogram... 16
Clearerr() function... 6
Clock Selection option... 52, 69, 88, 106
clock() function... 7
clock_gettime() function... 118
clock_nanosleep() function... 113
clock_settime() function... 118
clog() function... 5
clogf() function... 5
clogl() function... 5
Close subprogram... 11
close() function... xv, 6
closedir() function... 7
closelog() function... 9
command utility... 122
Component Profile... 26
component profile... 26
Conformance... 35
Conformance Document... 27
conformance document... 27, 28, 35
Conformant Application... 37
Conformant Application Using Extensions... 37

conforming application
strictly... 35
conforming implementation... 35
confstr() function... 8
conj() function... 5
conjf() function... 5
conjl() function... 5
connect() function... 7
constant

Blocking_Behavior... 22
False... 22
False..False... 42, 60, 79, 97, 111
File_Structure... 45, 63, 81
Group... 45
Operation_Not_Implemented... 22, 98
Operation_Not_Supported... 43, 61, 80, 98
Other... 45
Owner... 45
POSIX_Limits.Child_Processes_Maxima'Last... 45, 62
POSIX_Limits.Groups_Maxima'First... 22
POSIX_Limits.Groups_Maxima'First... 43, 61, 79
PTHREAD_SCOPE_PROCESS... 78, 96
PTHREAD_SCOPE_SYSTEM... 78, 86, 96, 105
Read_Write_Execute... 45
SCHED_FIFO... 41, 51, 59, 68, 78, 86, 96, 104, 105
SCHED_RR... 41, 51, 59, 68, 78, 86, 96, 104, 105
True...True... 39, 40, 42, 47, 57, 58, 60, 63, 64, 75, 76, 79, 82, 93, 94, 97, 100, 101, 111
constant-width format... 31
Conventions... 31
Copy_Environment subprogram... 16
Copy_From_Current_Environment subprogram... 16
Copy_To_Current_Environment subprogram... 16
copysign() function... 5
copysignf() function... 5
copysignl() function... 5
cos() function... 5
cosf() function... 5
cosh() function... 5
coshf() function... 5
coshl() function... 5

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
cosl() function... 5
cpowl() function... 5
cpow() function... 5
cpowf() function... 5
cproj() function... 5
cproject() function... 5
creal() function... 5
crealf() function... 5
creall() function... 5
creat() function... xvii, 7
Create subprogram... 46
Create_Directory subprogram... 13
Create_FIFO subprogram... 12
Create_Pipe subprogram... 14
Create_Process_Group subprogram... 13
Create_Session subprogram... 16
crontab utility... 122
crypt() function... 120
csin() function... 5
csinf() function... 5
csinh() function... 5
csinhf() function... 5
csinhl() function... 5
csin() function... 5
csplit utility... 122
csqrt() function... 5
csqrtf() function... 5
csqrtl() function... 5
ctags utility... 122
ctan() function... 5
ctanf() function... 5
ctanh() function... 5
ctanhf() function... 5
ctanhl() function... 5
ctermid() function... 7
ctime() function... 6
ctime_r() function... 6, 116
dbm_store() function... 9
Dedicated Realtime System Profile... 3, 75
Define_Bits_Per_Character subprogram... 11
Define_Input_Baud_Rate subprogram...
11
Define_Input_Time subprogram... 11
Define_Minimum_Input_Count subprogram... 11
Define_Output_Baud_Rate subprogram... 11
Define_Special_Control_Character subprogram... 11
Define_Terminal_Modes subprogram...
11
Definitions... 26
Delete subprogram... 46
Delete_All_Signals subprogram... 15
Delete_Environment_Variable subprogram... 16
Delete_Signal subprogram... 15
Development Environment... 16
Development Platform... 27
development platform... 27
df utility... 122
diffmtime() function... 6
dirname() function... 9
Disable_Control_Character subprogram... 11
Disable_Queueing subprogram... 125
Discard_Data subprogram... 11
div() function... 6
dlclose() function... 9
dlerror() function... 9
dlerror() function... 9
dlsym() function... 9
document conformance... 27
Documentation... 35
documentation system... 28
Drain subprogram... 11
drain48() function... 9
du utility... 122
dup() function... 7
dup2() function... 7
Duplicate subprogram... 12
Duplicate_And_Close subprogram... 12
1 File Structure constant... 45, 63, 81
2 Filename Truncation option... 42, 60, 79, 97
3 Filename Truncation option... 20, 22, 42, 60, 79, 97, 123
4 Filename_Of subprogram... 13
5 fileno() function... 6
6 flockfile() function... 7, 116
7 floor() function... 5
8 floor() function... 5
9 floor() function... 5
10 Flow subprogram... 11
11 fma() function... 5
12 fmaf() function... 5
13 fmal() function... 5
14 fmax() function... 5
15 fmaxf() function... 5
16 fmaxl() function... 5
17 fmin() function... 5
18 fminf() function... 5
19 fminl() function... 5
20 fmod() function... 5
21 fmodf() function... 5
22 fmodl() function... 5
23 fmatch() function... 8
24 fopen() function... 6, 44
25 For_Every_Current_Environment_Variable subprogram... 16
26 For_Every_Directory_Entry subprogram... 13
27 For_Every_Environment_Variable subprogram... 16
28 For_Every_File_In subprogram... 11
29 For_Every_Item subprogram... 124, 126
30 for() function... 7
31 format
32 format function family... 31
33 for77 utility... 122
34 fpathconf() function... 5
35 fpclassify() function... 5
36 fprintf() function... 6, 44
37 fwrite() function... 6, 44
38 fwrite() function... 6, 44
39 freeaddrinfo() function... 7
40 freopen() function... 6, 44
41 frexp() function... 5
42 frexp() function... 5
43 frexp() function... 5
44 fscanf() function... 6, 44
45 fseek() function... 7
46 fseeko() function... 7
47 fsetpos() function... 7
48 fstat() function... 7
49 ftell() function... 7
50 ftello() function... 7
51 ftell() function... 9
52 ftell() function... 9
53 fprintf() function... 7, 116
54 fwrite() function... 9
55 function
56 _Exit()... 7
57 _exit()... 7
58 _longjmp()... 9
59 _setjmp()... 9
60 _tolower()... 9
61 _toupper()... 9
62 a64l()... 9
63 abort()... 8
64 abs()... 6
65 accept()... 7
66 access()... 7
67 acos()... 5
68 acosf()... 5
69 acosh()... 5
70 acoshf()... 5
71 acosl()... 5
72 acosl()... 5
73 aio_cancel()... 113
74 aio_error()... 113
75 aio Fsyc()... 113
76 aio_read()... 113
77 aio_return()... 113
78 aio_suspend()... 113
79 aio_write()... 113
80 alarm()... 8, 44, 52, 62, 69, 88, 106
81 asctime()... 6
82 asctime_r()... 6, 116
83 asin()... 5
84 asinf()... 5
85 asinh()... 5
86 asinhf()... 5
87 asinhl()... 5
88 asinl()... 5
assert()... 7
cbrtf()... 5
cbrtl()... 5
cos()... 5
cosf()... 5
cosh()... 5
coshf()... 5
coshl()... 5
cosl()... 5
ceil()... 5
ceilf()... 5
ceill()... 5
cexp()... 5
cexpf()... 5
cexpl()... 5
cfgetispeed()... 7
cfgetospeed()... 7
cfsetispeed()... 7
cfsetospeed()... 7
chdir()... 7
chmod()... 7
chown()... 7
cimag()... 5
cimagf()... 5
cimagl()... 5
clearerr()... 6
clock()... 7
clock_getcpu()... 113
clock_getres()... 118
clock_gettime()... 44, 62, 118
clock_getpoint()... 113
clock_settime()... 118
clog()... 5
clogf()... 5
clog()... 5
close()... xv, 6
closedir()... 7
closelog()... 9
confstr()... 8
conj()... 5
conff()... 5
conjl()... 5
connect()... 7
copiesign()... 5
copiesignf()... 5
copiesignl()... 5

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

136
This is an unapproved IEEE Standards Draft, subject to change.
1 getpeername()... 7
2 getpgid()... 9
3 getppgrp()... 7
4 getpid()... 7
5 getpmsg()... 120
6 getppid()... 7
7 getpriority()... 9
8 getprotobyname()... 7
9 getprotobynumber()... 7
10 getprotoent()... 7
11 getpwent()... 9
12 getpwnam()... 8
13 getpwnam_r()... 8, 116
14 getpwuid()... 8
15 getpwuid_r()... 8, 116
16 getrlimit()... 9
17 getrusage()... 9
18 gets()... 6, 44
19 getservbyname()... 7
20 getservbyport()... 7
21 getservent()... 7
22 getsid()... 9
23 getsockname()... 7
24 getsockopt()... 7
25 getsockopt()... 7
26 getsubopt()... 9
27 gettimeofday()... 9
28 getuid()... 9
29 getutxent()... 10
30 getutxid()... 10
31 getutxline()... 10
32 getutctz()... 9
33 getwchar()... 9
34 getwd()... 120
35 glob()... 7
36 globfree()... 7
37 gmtime()... 6
38 gmtime_r()... 6, 116
39 grantpt()... 9
40 hcreate()... 9
41 hdestroy()... 9
42 hsearch()... 9
43 htonl()... 7
44 htons()... 7
45 hypot()... 5
46 hypotf()... 5
47 hypotl()... 5
48 iconv()... 9
49 iconv_close()... 9
50 iconv_open()... 9
51 if_freenameindex()... 7
52 if_indextoname()... 7
53 if_nameindex()... 7
54 if_nametoindex()... 7
55 ilogb()... 5
56 ilogbf()... 5
57 ilogbl()... 5
58 imaxabs()... 6
59 imaxdiv()... 6
60 index()... 120
61 inet_addr()... 7
62 inet_ntoa()... 7
63 inet_ntop()... 7
64 intstate()... 9
65 insue()... 9
66 ioctl()... xv, 120
67 isalnum()... 6
68 isalpha()... 6
69 isascii()... 9
70 isastream()... 120
71 isatty()... 7
72 isblank()... 6
73 iscntrl()... 6
74 isdigit()... 6
75 isfinite()... 5
76 isgraph()... 6
77 isgreater()... 5
78 isgreaterequal()... 5
79 isinf()... 5
80 isless()... 5
81 islessequal()... 5
82 islessgreater()... 5
83 islower()... 6
84 isnormal()... 5
85 isnan()... 5
86 isprint()... 6
87 ispunct()... 6
88 isspace()... 6
89 isnunordered()... 5
90 isspace()... 6
91 iswalnum()... 6
92 iswalpha()... 6
93 iswblank()... 6
94 iswctype()... 6
95 iswdigit()... 6
96 iswgraph()... 6
97 iswlower()... 6
98 iswprint()... 6
99 iswpunct()... 6
100 iswspace()... 6
101 iswupper()... 6
102 iswxdigit()... 6
103 isxdigit()... 6

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

139
j0()... 9
j1()... 9
jn()... 9
jrand48()... 9
kill()... 8, 44, 62, 80
killpg()... 9
l64a()... 9
labs()... 6
lchown()... 9
lelong48()... 9
ldexp()... 5
ldexpf()... 5
ldexpl()... 5
ldiv()... 6
lfind()... 9
lgamma()... 5
lgammaf()... 5
lgamma()... 5
link()... xvii, 7
lio_listio()... 113
listen()... 7
llabs()... 6
lldiv()... 6
llrint()... 5
llrintf()... 5
llrand48()... 9
llround()... 5
llroundf()... 5
llroundl()... 5
lround()... 5
lroundf()... 5
lroundl()... 5
lstat()... 8
main()... 48, 65
makecontext()... 9
malloc()... 6
mblen()... 6
mbrel()... 6
mbtowc()... 6
memccpy()... 9
memchr()... 6
memcpy()... 6
memmove()... 6
memset()... 6
mkdtr()... xvii, 7
mkfinfo()... 7
mkstemp()... 120
mktemp()... 9
mktime()... 6
mlock()... 114
mlockall()... 114
mmap()... 114, 115
modf()... 5
modff()... 5
mpprotect()... 114
mq_close()... 114
mq_getattr()... 114
mq_notify()... 114
mq_open()... 114
mq_receive()... 114
mq_send()... 114
mq_setattr()... 114
mq_timedreceive()... 114, 118
mq_timedsend()... 114, 118
mq_unlink()... 114
mrand48()... 9
msgget()... 9
msgget()... 9
msgsend()... 9
msgsnd()... 9
msync()... 114, 116
This is an unapproved IEEE Standards Draft, subject to change.

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

114, 115
posix_spawnattr_getschedpolicy()

114, 115
posix_spawnattr_getsigdefault()
posix_spawnattr_getsigmask()
posix_spawnattr_init()
posix_spawnattr_setflags()
posix_spawnattr_setgroup()
posix_spawnattr_setschedparam()
posix_trace_eventset_add()...119  
posix_trace_eventset_del()...119  
posix_trace_eventset_empty()...119  
posix_trace_eventset_fill()...119  
posix_trace_eventset_ismember()...119  
posix_trace_eventtypelist_getnext_id()...
..119  
posix_trace_eventtypelist_rewind()...
   119  
posix_trace_flush()...119  
posix_trace_get_attr()...119  
posix_trace_get_filter()...119  
posix_trace_get_status()...119  
posix_trace_getnext_event()...119  
posix_trace_open()...119  
posix_trace_rewind()...119  
posix_trace_set_filter()...119  
posix_trace_set_open()...119  
posix_trace_start()...119  
posix_trace_stop()...119  
posix_trace_timedgetnext_event()...
..118, 119  
posix_trace_trid_eventid_open()...119  
posix_trace_trygetnext_event()...119  
posix_typed_mem_get_info()...119  
posix_typed_mem_open()...119  
pow()...5  
powf()...5  
powl()...5  
pread()...9  
printf()...6, 44  
pselect()...7  
pthread_atfork()...8, 117  
pthread_attr_destroy()...8, 117  
pthread_attr_getdetachstate()...8, 117  
pthread_attr_getguardsize()...10  
pthread_attr_getinheritsched()...116, 118  
pthread_attr_getschedparam()...8, 117  
pthread_attr_getschedpolicy()...116, 118  
pthread_attr_getscope()...116, 118  
pthread_attr_getstack()...10, 116, 117  
pthread_attr_getstackaddr()...116, 117  
pthread_attr_getstacksize()...116, 117, 121  
pthread_attr_init()...8, 117  
pthread_attr_setdetachstate()...8, 117  
pthread_attr_setguardsize()...10  
pthread_attr_setinheritsched()...116, 118  
pthread_attr_setschedparam()...8, 117  

Copyright © 2003 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.
118

pthread_mutex_timedlock() ... 118

pthread_mutex_trylock() ... 8, 117

pthread_mutex_unlock() ... 8, 117

pthread_mutexattr_destroy() ... 8, 117

pthread_mutexattr_getprioceiling() ... 116, 118

pthread_mutexattr_getprotocol() ... 116, 117

pthread_mutexattr_getpshared() ... 116, 118

pthread_mutexattr_gettype() ... 10

pthread_mutex_init() ... 8, 117

pthread_mutex_lock() ... 8, 118

pthread_rwlock_timedrdlock() ... 8, 118

pthread_rwlock_timedwrlock() ... 8, 118

pthread_rwlock_tryrdlock() ... 8, 118

pthread_rwlock_trywrlock() ... 8, 118

pthread_rwlock_unlock() ... 8, 118

pthread_rwlockattr_destroy() ... 8, 118

pthread_rwlockattr_getpshared() ... 8, 116, 118

pthread_rwlockattr_init() ... 8, 118

pthread_rwlockattr_setpshared() ... 8, 116, 118

pthread_self() ... 8, 117

pthread_setcancelstate() ... 8, 117

pthread_setcanceltype() ... 8, 117

pthread_setscheduler() ... 116, 118

pthread_setschedparam() ... 116, 118

pthread_setspecific() ... 8, 117

pthread_setsigmask() ... 8, 117

pthread_spin_destroy() ... 115, 117

pthread_spin_init() ... 115, 117

pthread_spin_lock() ... 115, 117

pthread_spin_trylock() ... 115, 117

pthread_spin_unlock() ... 115, 117

pthread_testcancel() ... 8, 117

ptsname() ... 9

putc() ... 6, 44

putc_unlocked() ... 7, 116

putchar() ... 6, 44

putchar_unlocked() ... 7, 116

putenv() ... 9

putmsg() ... 120

putmsgs() ... 120

puts() ... 6, 44

pututxline() ... 10

putw() ... 9

putwchar() ... 9

fwrite() ... 9

qsort() ... 6

raise() ... 8, 44, 62

random() ... 6

random_r() ... 6, 116

read() ... xv, xvii, 6, 44

readdir() ... 7

readdir_r() ... 7, 116

readlink() ... 8

realloc() ... 6

realpath() ... 9

recv() ... 7

recvfrom() ... 7

recvmsg() ... 7

regcomp() ... 7

regerror() ... 7

regexec() ... 7

regfree() ... 7

remainder() ... 5

remainderf() ... 5

remainderl() ... 5

remove() ... 7

remqu() ... 9

remquo() ... 5

remquo() ... 5

remquo() ... 5

rename() ... xvii, 7

rewind() ... 7

rewinddir() ... 7

rindex() ... 120

rint() ... 5

rintf() ... 5

rintl() ... 5

rmdir() ... xvii, 7

round() ... 5

roundf() ... 5

roundl() ... 5

scalb() ... 9

scalbln() ... 5

scalbnf() ... 5

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
sem_unlink...
sem_init...
semget...
sem_wait...
sem_trywait...
sem_getvalue...
sem_close...
sched_yield...
sched_rr_get_interval...
sched_setparam...
sched_setscheduler...
sched_yield...
netsid...
seed48...
seekdir...
select...
sem_close...
sem_destroy...
sem_getvalue...
sem_init...
sem_open...
sem_post...
sem_timedwait...
sem_trywait...
sem_unlink...
sem_wait...
semcthread...
semget...
semop...
send...
sendmsg...
sendto...
sentbuf...
sentcontext...
setegid...
setenv...
seteuid...
setgid...
setgrent...
sethostent...
setitimer...
setjmp...
setkey...
setlocale...
setlogmask...
setnetent...
setpgid...
setpgrp...
setpriority...
setprotoent...
setpwnent...
setregid...

setreuid...
setrlimit...
setservent...
setsid...
setsockopt...
setsockopt...
setsockopt...
setsockopt...
sigaction...
sigaddset...
sigaltstack...
sigdelset...
sigemptyset...
sigfillset...
sighold...
sigignore...
siginterrrupt...
sigismember...
siglongjmp...
signal...
sigpause...
sigpending...
sigprocmask...
sigqueue...
sigrelse...
sigset...
sigsetjmp...
sigsuspend...
sigtimedwait...

sleep...
sprintf...
signal...
socket...
socketpair...
sprintf...
ungetc()... 6
ungetwc()... 6
unlink()... xvii, 7
unlockpt()... 9
unsetenv()... 8
usleep()... 9
utime()... 7
utimes()... 120
va_arg()... 6
va_copy()... 6
va_end()... 6
va_start()... 6
vfork()... 9
vfprintf()... 6, 44
vfscanf()... 6, 44
vfwprintf()... 9
vfwscanf()... 9
vprintf()... 6, 44
vscanf()... 6, 44
vswprintf()... 6
vswscanf()... 6
vwprintf()... 9
vwscanf()... 9
wait()... 7
waitid()... 9
waitpid()... 7
waitton()... 6
wctype()... 6
wcstoll()... 6
wcstombs()... 6
wcstoul()... 6
wsmax()... 6
wstoull()... 6
wstrmax()... 6
wstoull()... 6
wcsmax()... 6
wcsmax()... 120
wcswidth()... 10
wcssfrm()... 6
wetob()... 6
wetomb()... 6
wtrans()... 6
wctype()... 6
wcswidth()... 10
wmemchr()... 6
wmemcmp()... 6
wmemcp()... 6
wmemmove()... 6
wmemset()... 6
wordexp()... 8
wordexp()... 8
wprintf()... 9
write()... xv, xvii, 6, 44
write()... 9
wscanf()... 9
y0()... 9
y1()... 9
yn()... 9

function family
format... 31
functionality
unit of... 29
funlockfile() function... 7, 116
fwrite() function... 9
fwprintf() function... 9
fwrite() function... 6, 44
fscanf() function... 9

G

gai_strerror() function... 7
gerrorf() function... 120
Generic Application Environment Profile... 27
generic application environment profile... 27
generic environment profile... 33
generic interface profile... 27
Generic_Read subprogram... 11
Generic_Write subprogram... 11
Get subprogram... 46
Get_Allowed_Process_Permissions subprogram... 12
Get_Buffer subprogram... 14

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Get_Canonical_Name subprogram... 124, 126
Get_Ceiling_Priority subprogram... 124
Get_Close_On_Exec subprogram... 12
Get_Controlling_Terminal_Name subprogram... 11
Get_Data subprogram... 15
Get_Effective_Group_ID subprogram... 16
Get_Effective_User_ID subprogram... 16
Get_Family subprogram... 124, 126
Get_File subprogram... 124
Get_File_Control subprogram... 12
Get_Flags subprogram... 124, 126
Get_Groups subprogram... 16
Get_Locking_Policy subprogram... 124
Get_Login_Name subprogram... 16
Get_Maximum_Priority subprogram... 14
Get_Minimum_Priority subprogram... 14
Get_Notification subprogram... 15
Get.Owner subprogram... 14
Get_Parent_Process_Id subprogram... 13
Get_Process_Group_ID subprogram... 16
Get_Process_Group_Id subprogram... 13
Get_Process_Id subprogram... 13
Get_Process_Shared subprogram... 124, 125
Get_Program_Number subprogram... 124, 126
Get_Queue_Name subprogram... 16
Get_Real_Group_ID subprogram... 16
Get_Real_User_ID subprogram... 16
Get_Returned_Events subprogram... 124
Get_Round_Robin_Interval subprogram... 14
Get_Signal subprogram... 15
Get_Socket_Address_Info subprogram... 124, 126
Get_Socket_Type subprogram... 124, 126
Get_Terminal_Characteristics subprogram... 11
Get_Terminal_Name subprogram... 11
Get_Working_Directory subprogram... 13
Getaddrinfo() function... 7
getdate() function... 9
getegid() function... 9
getent() function... 8
geteuid() function... 9
getgid() function... 9
getgrent() function... 10
getgrgid() function... 8
getgrgid_r() function... 8, 116
getgrnam() function... 8
getgrent_r() function... 8, 116
getgroups() function... 9
gethostbyaddr() function... 7
gethostbyname() function... 7
gethostent() function... 7
gethostid() function... 9
gethostnam() function... 7
getitimer() function... 10
getlogin() function... 9
getlogin_r() function... 9, 116
getmsg() function... 120
getnameinfo() function... 7
getnetbyaddr() function... 7
gnetbyname() function... 7
gnetent() function... 7
getopt() function... 8
getpeername() function... 7
getpgid() function... 9
getpgrp() function... 7
getpid() function... 7
getpmsg() function... 120
getppid() function... 7
getprio() function... 10
getpriority() function... 9
getprotobyname() function... 7
getprotobynumber() function... 7
getprotoent() function... 7
getpwent() function... 9
getpwnam() function... 8
getpwunam() function... 8, 116
getpwnam_r() function... 8, 116
getpuuid() function... 8
getpuuid_r() function... 8, 116
getrlimit() function... 9
getrusage() function... 9
gets() function... 6, 44
getservbyname() function... 7
getservbyport() function... 7
getservent() function... 7
getsid() function... 9
getsid() function... 9
getsockname() function... 7
getsockopt() function... 7
gettimeofday() function... 9
```
getuid() function... 9
getutsent() function... 10
getutxid() function... 10
getutxline() function... 10
getpwd() function... 9
glob() function... 120
globfree() function... 7
gmtime() function... 6
gmtime_r() function... 6, 116
grantpt() function... 9
Group constant... 45

h
hcreate() function... 9
hdestroy() function... 9
header
    <limits.h>... 36
    <unistd.h>... 36, 39, 40, 47, 57, 58, 63,
    64, 75, 76, 82, 93, 94, 100, 101
hsearch() function... 9
htonl() function... 7
htons() function... 7
hypot() function... 5
hypotf() function... 5
hypotl() function... 5

i
iconv() function... 9
iconv_close() function... 9
iconv_open() function... 9
if_freenamelnex() function... 7
if_indextoname() function... 7
if_namelnex() function... 7
if_namelnex() function... 7
Ignore_Signal subprogram... 15
ilogb() function... 5
ilogbf() function... 5
ilogbl() function... 5
Image subprogram... 10, 43, 61, 80, 98
imaxabs() function... 6
imaxdiv() function... 6
Implementation Conformance... 35
implementation defined... 36, 37, 41, 59, 78, 96
terminology... 25
In_Set subprogram... 11, 125
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
inittate() 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
isctrl() function... 6
isdigit() function... 5
isfinite() function... 6
isgraph() function... 6
isgreater() function... 5
isgreaterequal() function... 5
isinf() function... 5
isleast() function... 5
islessequal() function... 5
```
islessgreater() function... 5
islower() function... 6
isnan() function... 5
isnormal() function... 5
ISO/IEC Conformant Application... 37
ISP... 27, 33
isprint() function... 6
isspace() function... 6
isspace() function... 6
isunordered() function... 5
isupper() function... 6
iswalnum() function... 6
iswalpha() function... 6
iswblank() function... 6
iswcntrl() function... 6
iswctype() function... 6
iswdigit() function... 6
iswgraph() function... 6
iswlower() function... 6
iswprint() function... 6
iswpunct() function... 6
iswspace() function... 6
iswupper() function... 6
iswxdigit() function... 6
isxdigit() function... 6
j0() function... 9
j1() function... 9
jn() function... 9
Job Control option... 13
jobs utility... 122
jrand48() function... 9
kill() function... 8, 44, 62, 80
killpg() function... 9
ldiv() function... 6
Length subprogram... 16
lex utility... 121
lfind() function... 9
lgamma() function... 5
lgammaf() function... 5
lgammal() function... 5
limit
POSIX_NGROUPS_MAX... 96
POSIX_RTSIG_MAX... 41, 59, 77, 96
POSIX_TIMER_MAX... 41, 59, 77, 96
CHILD_MAX... 96
Link subprogram... 13
link() function... xvii, 7
lio_listio() function... 113
listen() function... 7
llabs() function... 6
ldiv() function... 6
llrint() function... 5
llrintf() function... 5
llrintl() function... 5
llround() function... 5
llroundf() function... 5
llroundl() function... 5
localeconv() function... 6
localtime() function... 6
localtime_r() function... 6, 116
Lock_Shared_Memory subprogram... 125
lockf() function... 9
log() function... 5
log10() function... 5
log10f() function... 5
log10l() function... 5
log1p() function... 5
log1pf() function... 5
log1pl() function... 5
log2() function... 5
log2f() function... 5
log2l() function... 5
logb() function... 5
logbf() function... 5
logbl() function... 5
log() function... 5
long type... 96
longjmp() function... 4
lrand48() function... 9
lrint() function... 5
lrintf() function... 5
lrintl() function... 5
lround() function... 5
lroundf() function... 5
lroundl() function... 5

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
lroundl() function... 5
lseek() function... 9
lstat() function... 7
lstat() function... 8

M

macro
S-IRWXU... 44, 62, 80
main() function... 48, 65
make utility... 122
Make_Empty subprogram... 11
makecontext() function... 9
malloc() function... 6
Map_Memory subprogram... 123
may
terminology... 25
mbrel() function... 6
mbrel() function... 6
mbhrtowc() function... 6
mbssinit() function... 6
mbstowcs() function... 6
mbtowc() function... 6
memccpy() function... 9
memchr() function... 6
memcmp() function... 6
memcpy() function... 6
memmove() function... 6
Memory Locking option... 20, 42, 60, 79, 97, 123
Memory Mapped option... 125
Memory Mapped Files option... 20, 60, 79, 97, 123, 126
Memory Protection option... 20, 79, 97, 123
Memory Range option... 125
Memory Range Locking option... 20, 42, 60, 79, 97, 123
Memory-Mapped Files option... 69
memset() function... 6
msg utility... 122
Message Queues option... 20, 60, 79, 97, 123
Minimal Realtime System Profile... 2, 39
Minimum_Input_Count_Of subprogram... 11
mkdir() function... xvii, 7
mkfifo() function... 7
mknod() function... 9
mkstemp() function... 9
mktemp() function... 120
mktime() function... 6
mlock() function... 114
mlockall() function... 114
mmap() function... 114, 115
MMU... 32
modfl() function... 5
modff() function... 5
modfl() function... 5
Monotonic Clock option... 52, 69, 88, 106
more utility... 122
mprotect() function... 114
mq_close() function... 114
mq_getattr() function... 114
mq_notify() function... 114
mq_open() function... 114
mq_recev() function... 114
mq_send() function... 114
mq_setattr() function... 114
mq_timedrecv() function... 114, 118
mq_timedsend() function... 114, 118
mq_unlink() function... 114
mrand48() function... 9
msgctl() function... 9
msgget() function... 9
msgget() function... 9
msgsnd() function... 9
Multi-Purpose Realtime System Profile... 3, 93
munlock() function... 114
munlockall() function... 114
munmap() function... 114, 115
Mutex Priority Ceiling option... 21, 42, 60, 79, 98, 124
Mutex Priority Inherit option... 124
Mutex Priority Inheritance option... 21, 42, 60, 79, 98, 124
Mutexes option... 124
Mutexes Support option... 42, 60, 79
Mutexes Supported option... 98
MutexPriority Ceiling option... 124

N

nan() function... 5
nanf() function... 5
nanl() function... 5
nanosleep() function... 118
National Body Conformant POSIX.13 Application... 37
nearbyint() function... 5
nearbyintf() function... 5
nearbyintl() function... 5

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
off_t type... 96
Open subprogram... 11, 46
open system environment... 28, 33
open() function... xv, 6, 44, 49
open_or_Create subprogram... 13
opendir() function... 7
openlog() function... 9
Operation_Not_Implemented constant... 22, 98
Operation_Not_Supported constant... 43, 61, 80, 98
option
_POSIX_ADVISORY_INFO... 18, 20, 49, 66, 84, 95
_POSIX_AEP_REALTIME... 39, 57, 75, 93
_POSIX_AEP_REALTIME_CONTROLLE R... 57
_POSIX_AEP_REALTIME_DEDICATED ... 75
_POSIX_AEP_REALTIME_LANG_Ada9 5... 40, 47, 58, 64, 76, 82, 94, 101
_POSIX_AEP_REALTIME_LANG_C99... 40, 47, 58, 63, 76, 82, 94, 100
_POSIX_AEP_REALTIME_MINIMAL... 39
_POSIX_AEP_REALTIME_MULTI... 93
_POSIXASYNCHRONOUS_IO... 18, 20, 77, 95, 120
_POSIX_BARRIERS... 18, 20
_POSIX_CHOWN_RESTRICTED... 18, 20,

POSIX Clock selection... 18, 20, 41, 59, 77, 95
_POSIX_CPU TIME... 18, 20, 77, 95
_POSIX_FSYNC... 18, 20, 41, 59, 77, 95, 120
_POSIX_IPV6... 18, 20
_POSIX_JOB_CONTROL... 10
_POSIX_JOB_CONTROL... 18, 95
_POSIX_MAPPED_FILES... 18, 20, 49, 59, 77, 95, 120
_POSIX_MEMLOCK... 18, 20, 41, 48, 59, 77, 95, 120
_POSIX_MEMORY_RANGE... 18, 20, 41, 59, 77, 95, 120
_POSIX_MEMORY_PROTECTION... 18, 20, 77, 95, 120
_POSIX_MESSAGE_PASSING... 18, 20, 59, 77, 95, 120
_POSIX_MONOTONIC_CLOCK... 19, 20, 41, 59, 77, 95
_POSIX_NO_TRUNC... 19, 20, 22, 41, 49, 59, 77, 95
_POSIX_PRIORITY... 19, 20, 77, 95, 120
_POSIX_PRIORITY_SCHEDUL... 19, 20, 77, 86, 95, 104, 120
_POSIX_RAW_SOCKETS... 19, 20, 77, 95
_POSIX_READER_WRITER_LOCKS...
10
_POSIX_READER_WRITER_LOCKS...
19, 20
_POSIX_REALTIME_SIGNALS... 19, 20, 41, 59, 77, 95, 120
_POSIX_REGEXP... 10
_POSIX_REGEXP... 19, 21, 95
_POSIX_SAVED_IDS... 19, 21, 95
_POSIX_SEMAPHORES... 19, 21, 41, 59, 77, 95, 120
_POSIX_SHARED_MEMORY_OBJECTS...
19, 21, 41, 59, 77, 95, 120
_POSIX_SHELL... 19, 21, 96
_POSIX_SPAWN... 19, 21, 77, 96
_POSIX_SPIN_LOCKS... 19, 21
_POSIX_SPIRATIC_SERVER... 19, 21, 77, 96
_POSIX_SYNCHRONIZED_IO... 19, 21, 41, 59, 77, 96, 120
_POSIX_THREAD_ATTR_STACKADDR...
19, 21, 41, 59, 77, 96
_POSIX_THREAD_ATTR_STACKSIZE...
19, 21, 41, 59, 77, 96

Network Management option... 124, 126
newgrp utility... 122
nextafter() function... 5
nextafter() function... 5
nextafter() function... 5
nexttooward() function... 5
nexttooward() function... 5
nexttooward() function... 5
strftime() function... 9
nice utility... 122
nice() function... 9
nl_langinfo() function... 9
rm utility... 122
Normative References... 23
nrand48() function... 9
ntol() function... 7
ntohl() function... 7

O
IEEE Std P1003.13-Draft 2.1, Feb 2003
STANDARDIZED APPLICATION ENVIRONMENT PROFILE

_POSIX_THREAD_CPUTIME... 19, 21, 41, 59, 77, 96
_POSIX_THREAD_PRIO_INHERIT...
19, 21, 41, 59, 77, 96, 120
_POSIX_THREAD_PRIO_PROTECT...
19, 21, 41, 59, 77, 96, 120
_POSIX_THREAD_PRIORITY_SCHEDUL
ING... 19, 21, 41, 59, 77, 96, 120
_POSIX_THREAD_PROCESS_SHARED...
.. 10
_POSIX_THREAD_PROCESS_SHARED...
19, 21, 77, 96
_POSIX_THREAD_SAFE_FUNCTIONS...
19, 21, 96
_POSIX_THREAD_SPIRATICK_SERVER...
19, 21, 41, 59, 77, 96
_POSIX_THREAD_STACK_ADDRESS...
121
_POSIX_THREAD_STACK_SIZE... 121
_POSIX_THREADS... 10, 19, 21, 53, 70, 88, 107
_POSIX_TIMEOUTS... 10
_POSIX_TIMEOUTS... 19, 21, 41, 59, 77, 96
_POSIX_TIMERS... 19, 21, 41, 59, 77, 96, 120
_POSIX_TRACE... 19, 21, 59, 77, 96
_POSIX_TRACE_EVENT_FILTER... 19, 21, 59, 77, 96
_POSIX_TRACE_INHERIT... 19, 21
_POSIX_TRACE_LOG... 19, 21, 59, 77, 96
_POSIX_TYPED_MEMORY_OBJECTS...
19, 21
_POSIX_VDISABLE... 19, 21, 22, 96
_POSIX_VERSION... 44, 62, 80
_POSIX2_C_BIND... 19, 21
_POSIX2_C_DEV... 19, 21
_POSIX2_CHAR_TERM... 19, 21
_POSIX2_FORT_DEV... 19, 21
_POSIX2_FORT_RUN... 19, 21
_POSIX2_LOCALEDEF... 19, 21
_POSIX2_PBS... 19, 21
_POSIX2_PBS_ACCOUNTING... 19, 21
_POSIX2_PBS_CHECKPOINT... 20, 21
_POSIX2_PBS_LTOLOCATE... 20, 21
_POSIX2_PBS_MESSAGE... 20, 21
_POSIX2_PBS_TRACK... 20, 21
_POSIX2_SW_DEV... 20, 21
_POSIX2_UPE... 20, 21
_XOPEN_CRYPT... 20, 21, 55, 72, 90, 109
_XOPEN_ENH_I18N... 20, 21
_XOPEN_LEGACY... 20, 22, 55, 72, 90, 109
_XOPEN_REALTIME... 20, 22
_XOPEN_REALTIME_THREADS... 20, 22
_XOPEN_SHM... 20, 22
_XOPEN_STREAMS... 20, 22, 55, 72, 90, 109
_XOPEN_UNIX... 20, 22
Ada Language... 42, 45, 60, 62, 72, 78, 80, 97, 99, 100
Ada language... 91, 109
Ada-Language... 18, 36
Advisory Information... 102
Asynchronous I/O... 20, 79, 97, 123
C Language... 40, 43, 61, 72, 76, 80, 94, 99
C language... 58, 90, 109
Change Owner Restriction... 20, 97, 123
CHILD_MAX... 44, 62
C-Language... 18, 36
C-language... 20
Clock Selection... 52, 69, 88, 106
File Locking... 49, 66, 102
File Synchronization... 49, 66, 84
File Synchronization... 20, 42, 60, 79, 97, 123
Filename Truncation... 42, 60, 79, 97
Filename Truncation... 20, 22, 42, 60, 79, 97, 123
Internet Datagram... 14
Internet Protocol... 14
Internet Protocol Version 6... 89, 108
Internet Stream... 14
Job Control... 13
Memory Locking... 20, 42, 60, 79, 97, 123
Memory Mapped... 125
Memory Mapped Files... 20, 60, 79, 97, 123, 126
Memory Protection... 20, 79, 97, 123
Memory Range... 125
Memory Range Locking... 20, 42, 60, 79, 97, 123
Memory-Mapped Files... 69
Message Queues... 20, 60, 79, 97, 123
Monotonic Clock... 52, 69, 88, 106
Mutex Priority Ceiling... 21, 42, 60, 79, 98, 124
Mutex Priority Inherit... 124
Mutex Priority Inheritance... 21, 42, 60, 79, 98, 124
Mutexes... 124
Mutexes Support... 42, 60, 79
Mutexes Supported... 98
MutexPriority Ceiling... 124
Network Management... 124, 126

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Poll... 124
POSIX2_C_BIND... 16, 46, 63, 81, 99, 100
POSIX2_C_DEV... 16, 46, 63, 81, 100
POSIX2_CDEV... 99
POSIX2_CHAR_TERM... 99, 100
POSIX2_FORT_RUN... 99, 100
POSIX2_SW_DEV... 16, 46, 47, 63, 64, 81, 82, 99, 100, 101
POSIX2_UPE... 99, 100
Prioritized I/O... 20, 79, 97, 124
Priority Process Scheduling... 20, 79, 97, 125
Process Shared... 50, 67, 85, 104
Process Shared... 21, 79, 98, 124
Process Shared and Mutexes... 125
Raw Sockets... 89, 108
Realtime Signals... 20, 42, 60, 79, 97, 125
Required... 18
Saved IDs Support... 21, 97, 125
Select... 12
Select... 125
Semaphore... 21, 42, 60, 79, 97, 125
Server Scheduling... 67
Shared Memory... 123
Shared Memory Objects... 87
Shared Memory Objects... 21, 42, 60, 79, 98, 125
Sockets Detailed... 124
Sockets Detailed Network Interface... 14
Sockets Detailed Network Interface... 126
spawn... 65
Sporadic Server Scheduling... 51, 86, 104
Synchronized I/O... 21, 42, 60, 79, 98, 123, 126
Timeouts... 52, 70, 88, 107
Timers... 21, 42, 60, 79, 98, 126
Trace Event Filtering... 70, 89, 107
Trace Log... 70, 89, 107
XTI Detailed Network Interface... 126
OSE... 28, 33
Other constant... 45
Output_Baud_Rate_Of subprogram... 11
Owner constant... 45

P

package
Ada_Streams... 10
Ada_Task_Identification... 10
POSIX... 15

POSIX_Calendar... 15
POSIX_Condition_Variables...
124, 125
POSIX_Configurable_File_Limit
s... 12
POSIX_Configurable_System_Lim
its... 15
POSIX_Event_Mangement... 11, 124,
125
POSIX_File_Locking... 12
POSIX_File_Status... 12
POSIX_Files... 12, 13
POSIX_Generic_Shared_Memory...
125
POSIX_Group_Database... 16
POSIX_IO... 11, 12, 13, 14, 125, 126
POSIX_Limits... 15
POSIX_Memory_Mapping... 126
POSIX_Mutexes... 124, 125
POSIX_Options... 15, 39, 57, 75, 93
POSIX_Page_ALIGNMENT... 10
POSIX_Permissions... 12
POSIX_Process_Environment... 13,
16
POSIX_Process_Identification...
13, 16
POSIX_Process_Primitives... 13
POSIX_Process_Scheduling... 14,
125
POSIX_Process_Times... 13
POSIX_Profiles... 15, 16, 43, 61, 79,
98, 111
POSIX_Semaphores... 125
POSIX_Shared_Memory_Objects...
125
POSIX_Sockets... 14, 124, 126
POSIX_Sockets_Extended... 14
POSIX_Sockets_Local... 14
POSIX_Supplement_To_Ada_IO... 10
POSIX_Terminal_Functions... 11,
13
POSIX_Timers... 126
POSIXUnsafe_Process_Primitiv
es... 13, 78
POSIX_User_Database... 16
POSIX_XTI... 126
System... 10
System_Storage_Elements... 10
patch utility... 122
pathconf() function... 7
pause() function... 8

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
pclose() function... 8
Pending_Signals subprogram... 15
perror() function... 6, 44
pipe() function... 7
platform
  application... 26
  development... 27
Poll option... 124
Poll subprogram... 124
poll() function... 9
popen() function... 8
POSIX package... 15
POSIX.1... 32
POSIX.5c... 32
POSIX_ADA_LANG_SUPPORT unit of functionality... 10, 17, 42, 60, 78, 97
POSIX_C_LANG_JUMP unit of functionality...
  4, 17, 40, 58, 76, 94
POSIX_C_LANG_MAT unit of functionality...
  58
POSIX_C_LANG_MATH unit of functionality...
  5, 17, 55, 76, 94
POSIX_C_LANG_SUPPOR unit of functionality...
  58
POSIX_C_LANG_SUPPORT unit of functionality...
  6, 17, 40, 76, 94
POSIX_C_LANG_WIDE_CHAR unit of functionality...
  6, 17, 55, 72, 90, 94
POSIX_C_LIB_EXT unit of functionality...
  40
POSIX_Calendar package... 15
POSIX_Condition_Vari ables package...
  124, 125
POSIX_Configurable_File_Limits package... 12
POSIX_Configurable_System_Limits package... 15
POSIX_Configurable_System_Limits
  .System_POSIX_Ada_Version subprogram... 45, 62, 81
POSIX_Configurable_System_Limits
  .System_POSIX_Version subprogram... 43, 62, 80
POSIX_DEVICE_IO unit of functionality...
  6, 11, 17, 40, 42, 58, 60, 76, 78, 94, 97
POSIX_DEVICE_SPECIFIC unit of functionality...
  7, 11, 17, 94, 97
POSIX_Error exception... 22, 43, 61, 80, 98
POSIX_Event_Management package...
  11, 124, 125
POSIX_EVENT_MGMT unit of functionality...
  7, 11, 17, 76, 78, 94, 97
posix_fadvise() function... 113
posix_fallocate() function... 113
POSIX_FD_MGMT unit of functionality...
  7, 12,
  17, 49, 58, 60, 76, 78, 94, 97
POSIX_FIFO unit of functionality...
  7, 12, 17,
  49, 66, 84, 94, 97
POSIX_FILE_ATTRIBUTES unit of functionality...
  7, 12, 17, 49, 66, 84, 94, 97
POSIX_File_Locking package... 12
POSIX_FILE_LOCKING unit of functionality...
  7, 12, 17, 49, 58, 76, 94
POSIX_Files package... 12
POSIX_Gen eric_Shared_Memory package... 125
POSIX_Group_Database package... 16
POSIX_IO package...
  11, 12, 13, 14, 125, 126
POSIX_IO.Generic_Read subprogram...
  45
POSIX_IO.Generic_Write subprogram...
  45
POSIX_IO.Open subprogram...
  45
POSIX_IO.Open_Or_Create subprogram...
  45
POSIX_IO.Read subprogram...
  45
POSIX_IO.Write subprogram...
  45
POSIX_JOB_CONTROL unit of functionality...
  7, 13, 17, 22, 94, 97, 114
POSIX_Limits package...
  15
POSIX_Limits.Child_Processes_Maxima'Last constant... 45, 62
POSIX_Limits.Child_Processes_Maxima'First type...
  98
POSIX_Limits.Groups_Maxima'First constant... 22
POSIX_Limits.Groups_Maxima'First constant...
  43, 61, 79
POSIX_Limits.Groups_Maxima'First type...
  98
POSIX_Limits.Realtime_Signals_Maxima'First type...
  42, 60, 79, 98
POSIX_Limits.Timers_Maxima'First type...
  42, 60, 79, 98
posix_madvise() function...
  113, 114, 115
posix_mem_offset() function...
  119
posix_memalign() function...
  113
POSIX_Memory_Mapping package...
  126
POSIX_MULTI_PROCESS unit of functionality...
  7, 13, 17, 76, 78, 94, 97

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

154
POSIX驲etime and Embedded Application Profiles

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

156
POSIX2_CDEV option... 99
POSIX2_CHAR_TERM option... 99, 100
POSIX2_FORT_RUN option... 99, 100
POSIX2_SW_DEV option... 16, 46, 47, 63, 64, 81, 82, 99, 100, 101
POSIX2_UPE option... 99, 100

pow() function... 5
powf() function... 5
powl() function... 5
pread() function... 9
printf() function... 6, 44
Prioritized I/O option... 20, 79, 97, 124
delay ceiling protocol... 50, 67, 85, 104
priority inheritance protocol... 50, 67, 85, 104
Priority Inversion... 28
priority inversion... 50, 67, 85, 104
Priority Process Scheduling option... 20, 79, 97, 125
delay protection protocol
see priority ceiling protocol
Process Shared and Mutexes option... 125
Process Shared option... 50, 67, 85, 104
Process Shared option... 21, 79, 98, 124
profile
call application environment... 26, 27
compact... 26
for ISO standardization... 28
generic application environment... 27
generic interface... 27
industry specific interface... 27
interface... 27
international standardized... 27, 33
realtime environment... 28
system... 28
profile, application environment... 33
profile, generic environment... 33
profile, protocol
priority ceiling... 50, 67, 85, 104
priority inheritance... 50, 67, 85, 104
ps utility... 122
PSE... 33
PSE51... 2, 33, 39
PSE52... 3, 33, 57
PSE53... 3, 33, 75
PSE54... 3, 33, 93
pselect() function... 7
pthread_attr_fork() function... 8, 117
pthread_attr_destroy() function... 8, 117
pthread_attr_getdetachstate() function... 8, 117
pthread_attr_getguardsize() function... 10
pthread_attr_getinheritsched() function... 116, 118
pthread_attr_getschedparam() function... 8, 117
pthread_attr_getschedpolicy() function... 116, 118
pthread_attr_getscope() function... 116, 118
pthread_attr_getstack() function... 10, 116, 117
pthread_attr_getstackaddr() function... 116, 117
pthread_attr_getstacksize() function... 116, 117, 121
pthread_attr_init() function... 8, 117
pthread_attr_setdetachstate() function... 8, 117
pthread_attr_setguardsize() function... 10
pthread_attr_setinheritsched() function... 116, 118
pthread_attr_setschedparam() function... 8, 117
pthread_attr_setschedpolicy() function... 116, 118
pthread_attr_setscope() function... 116, 118
pthread_attr_setstack() function... 10, 116, 117
pthread_attr_setstackaddr() function... 116, 117
pthread_attr_setstacksize() function... 116, 117, 121
pthread_barrier_destroy() function... 113, 117
pthread_barrier_init() function... 113, 117
pthread_barrier_wait() function... 113, 117
pthread_bARRIERATTR_destroy() function... 113, 117
pthread_bARRIERATTR_getshared() function... 113, 116, 117
pthread_bARRIERATTR_init() function... 113, 117
pthread_bARRIERATTR_setshared() function... 113, 116, 117
pthread_cancel() function... 8, 117
pthread_cleanup_pop() function... 8, 117
pthread_cleanup_push() function... 8, 117
pthread_cond_broadcast() function... 8, 117
pthread_cond_destroy() function... 8, 117
pthread_cond_init() function... 8, 117
pthread_cond_signal() function... 8, 117
pthread_cond_timedwait() function... 8, 117
pthread_cond_wait() function... 8, 117
pthread_barrierattr_destroy() function... 8, 117
pthread_barrierattr_getshared() function... 113, 116, 117
pthread_barrierattr_init() function... 113, 117
pthread_barrierattr_setshared() function... 113, 116, 117

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
pthread_condattr_getpshared() function...
117

pthread_condattr_init() function... 8, 117

pthread_condattr_setclock() function... 113, 117

pthread_condattr_setpshared() function...
116, 118

pthread_create() function... 8, 117

pthread_detach() function... 8, 117

pthread_equal() function... 8, 117

pthread_exit() function... 8, 117

pthread_getconcurrency() function... 10

pthread_getpwpid() function... 116, 117

pthread_getschedparam() function... 116,
118

pthread_getspecific() function... 8, 117

pthread_join() function... 8, 117

pthread_key_create() function... 8, 117

pthread_key_delete() function... 8, 117

pthread_kill() function... 8, 117

pthread_mutex_destroy() function... 8, 117

pthread_mutex_getprioceiling() function...
116, 118

pthread_mutex_init() function... 8, 117

pthread_mutex_lock() function... 8, 117

pthread_mutex_setprioceiling() function...
116, 118

pthread_mutex_timedlock() function... 8,
118

pthread_mutex_timedwrlock() function... 8,
118

pthread_mutex_trylock() function... 8, 118

pthread_mutexattr_destroy() function... 8,
118

pthread_mutexattr_getpshared() function...
8, 116, 118

pthread_mutexattr_init() function... 8, 118

pthread_mutexattr_setpshared() function...
8, 116, 118

PTHREAD_SCOPE_PROCESS constant... 78, 96

PTHREAD_SCOPE_SYSTEM constant... 78, 86,
96, 105

pthread_self() function... 8, 117

pthread_setcalcelstate() function... 8, 117

pthread_setconcurrency() function... 10

pthread_setschedparam() function... 116,
118

pthread_setschedprio() function... 116, 118

pthread_setspecific() function... 8, 117

pthread_sigmask() function... 8, 117

pthread_spins_destroy() function... 115, 117

pthread_spin_init() function... 115, 117

pthread_spins_lock() function... 115, 117

pthread_spins_unlock() function... 115, 117

pthread_testcancel() function... 8, 117

putsname() function... 9

putc() function... 6, 44

putc_unlocked() function... 7, 116

putchar() function... 6, 44

putchar_unlocked() function... 7, 116

putenv() function... 9

putmsg() function... 120

putpmsg() function... 120

putst() function... 6, 44

pututxline() function... 10

putwc() function... 9

putwchar() function... 9

putwrite() function... 9

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

qalter utility... 122
qdel utility... 122
qhold utility... 122
qmove utility... 122
qmsg utility... 122
qrls utility... 122
qrs select utility... 122
qrsig utility... 122
qrsort() function... 6
qrs stat utility... 122
qrs sub utility... 122
Queue_Signal subprogram... 125

R

raise() function... 8, 44, 62
rand() function... 6
rand_r() function... 6, 116
random() function... 9
Raw Sockets option... 89, 108
Read subprogram... 11, 46
read() function... xv, xvii, 6, 44
Read_Write_Execute constant... 45
readdir() function... 7
readdir_r() function... 7, 116
readlink() function... 8
readv() function... 9
malloc() function... 6
realpath() function... 9
Realtime Controller System Profile... 3, 57
realtime environment profile... 28
Realtime Signals option... 20, 42, 60, 79, 97, 125
Realtime System Profiles... 2
Realtime_Lang_C99 type... 40, 47, 58, 63, 76, 82, 94, 100
recv() function... 7
recvfrom() function... 7
recvmsg() function... 7
regcomp() function... 7
regerror() function... 7
regexec() function... 7
regfree() function... 7
Related Open Systems Standards... 127
remainder() function... 5
remainderf() function... 5
remainderl() function... 5
Remove subprogram... 11, 125
remove() function... 7
Remove_Directory subprogram... 13
remqau() function... 9
remqua() 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
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

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
sem_close() function... 115
sem_destroy() function... 115
sem_getvalue() function... 115
sem_init() function... 115
sem_open() function... 115
sem_post() function... 44, 62, 115
sem_timedwait() function... 115, 118
sem_trywait() function... 115
sem_unlink() function... 115
sem_wait() function... 115

Semaphores option... 21, 42, 60, 79, 97, 125

semctl() function... 9
semget() function... 9
semop() function... 9
send() function... 7
Send_Break subprogram... 11
Send_Signal subprogram... 15
sendmsg() function... 7
sendto() function... 7
Serve_Break subprogram... 11
Server Scheduling option... 67
Set_Allowed_Process_Permissions subprogram... 12
Set_Blocked_Signals subprogram... 15
Set_Buffer subprogram... 14
Set_Ceiling_Priority subprogram... 124
Set_Close_On_Exec subprogram... 12
Set_Data subprogram... 15
Set_Environment_Variable subprogram... 16
Set_Events subprogram... 124
Set_Family subprogram... 124, 126
Set_File subprogram... 124
Set_File_Control subprogram... 12
Set_File_Times subprogram... 13
Set_Flags subprogram... 124, 126
Set_Group_ID subprogram... 16
Set_Locking_Policy subprogram... 124
Set_Notification subprogram... 15
Set_process_Group_Id subprogram... 13
Set_process_Shared subprogram... 124, 125
Set_Protocol_Number subprogram... 124, 126
Set_Returned_Events subprogram... 124
Set_Signal subprogram... 15
Set_Socket_Group_Owner subprogram... 14
Set_Socket_Process_Owner subprogram... 14
Set_Socket_Type subprogram... 124, 126
Set_Stopped_Child_Signal subprogram... 13

Set_Terminal_Characteristics subprogram... 11
Set_User_ID subprogram... 16
setbuf() function... 6
setcontext() function... 9
setgid() function... 9
setenv() function... 8
seteuid() function... 9
setgid() function... 9
setgrent() function... 10
sethostent() function... 7
setitimer() function... 10
setjmp() function... 4
setkey() function... 120
setlocale() function... 6
setlogmask() function... 9
setnetent() function... 7
setpgid() function... 7
setpgrp() function... 9
setpriority() function... 9
setprotoent() function... 7
setpwent() function... 9
setregid() function... 10
setreuid() function... 10
setrlimit() function... 9
setservent() function... 7
setsid() function... 7
setsockopt() function... 7
setstate() function... 9
setuid() function... 9
setutxent() function... 10
setvbuf() function... 6
sh utility... 121
shall
terminology... 25
Shared Memory option... 123
Shared Memory Objects option... 87
Shared Memory Objects option... 21, 42, 60, 79, 98, 125
shm_open() function... 115
shm_unlink() function... 115
shmat() function... 9
shmctl() function... 9
shmget() function... 9
should
terminology... 25
shutdown() function... 7
sigaction() function... 8, 44, 62
sigaddset() function... 8, 44, 62
sigaltstack() function... 9
sigdelset function... 44

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
For_Every_Environment_Variable... 16
For_Every_File_In... 11
For_Every_Item... 124, 126
Generic_Read... 11
Generic_Write... 11
Get... 46
Get_Allowed_Process_Permissions... 12
Get_Buffer... 14
Get_Canonical_Name... 124, 126
Get_Ceiling_Priority... 124
Get_Close_On_Exec... 12
Get_Controlling_Terminal_Name... 11
Get_Data... 15
Get_Effective_Group_ID... 16
Get_Effective_User_ID... 16
GetEvents... 124
Get_Family... 124, 126
Get_File... 124
Get_File_Control... 12
Get_Flags... 124, 126
Get_Groups... 16
Get_Locking_Policy... 124
Get_Login_Name... 16
Get_Maximum_Priority... 14
Get_Minimum_Priority... 14
Get_Notification... 15
Get_Owner... 14
Get_Parent_Process_Id... 13
Get_Process_Group_Id... 13
Get_Process_Id... 13
Get_Process_Socket_Address_Info... 124, 126
Get_Signal... 15
Get_Socket_Type... 124, 126
Get_Terminal_Characteristics... 11
Get_Terminal_Name... 11
Get_Working_Directory... 13
Ignore_Signal... 15
Image... 10, 43, 61, 80, 98
In_Set... 11, 125
Input_Baud_Rate_Of... 11

Await_Signal... 15, 125
Await_Signal_Or_Timeout... 15, 125
Bits_Per_Character_Of... 11
Block.Signals... 15
Blocked_Signals... 15
Change_Owner_And_Group... 12
Change_Permissions... 12, 123
Change_Protection... 123
Change_Working_Directory... 13
Clear_Environment... 16
Close... 11
Copy_Environment... 16
Copy_From_Current_Environment... 16
Create... 46
Create_Directory... 13
Create_FIFO... 12
Create_Pipe... 14
Create_Process_Group... 13
Create_Session... 16
Define_Bits_Per_Character... 11
Define_Input_Baud_Rate... 11
Define_Input_Time... 11
Define_Minimum_Input_Count... 11
Define_Output_Baud_Rate... 11
Define_Special_Control_Character... 11
Define_Terminal_Modes... 11
Delete... 46
Delete_All_Signals... 15
Delete_Data... 16
Delete_Environment_Variable... 16
Delete_Signal... 15
Disable_Control_Character... 11
Disable_Draining... 125
Discard_Data... 11
Drain... 11
Duplicate... 12
Duplicate_And_Close... 12
Enable_Discarding... 125
Environment_Value_Of... 16
Existence... 13
File_Position... 12
File_Size... 12
Filename_Of... 13
Flow... 11
For_Every_Current_Environment_Variable... 16
For_Every_Directory_Entry... 13

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Input_Time_Of... 11
Install_Empty_Handler... 15
Interrupt_Task... 15
Is_A_Terminal... 11
Is_Accessible... 13
Is_Block_Special_File... 13
Is_Character_Special_File... 13
Is_Directory... 13
Is_Environment_variable... 16
Is_FIFO... 13
Is_File... 13
Is_File_Present... 13
Is_Ignored... 15
Is_Member... 15
Is_Open... 11
Is_Socket... 13
Length... 16
Link... 13
Lock_Shared_Memory... 125
Make_Empty... 11
Map_Memory... 123
Minimum_Input_Count_Of... 11
Open... 11, 46
Open_Or_Create... 13
Output_Baud_Rate_Of... 11
Pending_Signals... 15
Poll... 124
POSIX_Configurable_System_Limit
its.System_POSIX_Ada_Versi
on... 45, 62, 81
POSIX_Configurable_System_Lim
its.System_POSIX_Versio
on... 45, 62, 80
POSIX_IO_Generic_Read... 45
POSIX_IO_Generic_Write... 45
POSIX_IO_Open... 45
POSIX_IO_Open_Or_Create... 45
POSIX_IO_Read... 45
POSIX_IO_Write... 45
Posix_Process_Primitives.Star
t_Process... 81
Posix_Process_Primitives.Star
t_Process_Search... 81
POSIX_Signals_Set_Stopped_Child
Signal... 22, 43, 61, 79
POSIX_Signals_Signaled_Child_S
ignal_Enabled... 22, 43, 61, 79
POSIX_Terminal_Functions.Disa
ble_Control_Character... 22, 98
Posix_Unsafe_Process_Primitiv
... 81
Put... 46
Queue_Signal... 125
Read... 11, 46
Remove... 11, 125
Remove_Directory... 13
Rename... 13
Seek... 12
Select_File... 11, 125
Send_Break... 11
Send_Signal... 15
Set_Allowed_Process_Permissio
ns... 12
Set_Blacklisted_Signals... 15
Set_Buffer... 14
Set_Ceiling_Priority... 124
Set_Close_On_Exec... 12
Set_Data... 15
Set_Environment_variable... 16
Set_Events... 124
Set_Family... 124, 126
Set_File... 124
Set_File_Control... 12
Set_File_Times... 13
Set_Flags... 124, 126
Set_Group_ID... 16
Set_Locking_Policy... 124
Set_Notification... 15
Set_Process_Group_Id... 13
Set_Process_Shared... 124, 126
Set_Proto_Col_Number... 124, 126
Set_Returned_Events... 124
Set_Signal... 15
Set_Socket_Group_Owner... 14
Set_Socket_Process_Owner... 14
Set_Socket_Type... 124, 126
Set_Socket_Group... 14
Set_Stopped_Child_Signal... 13
Set_Terminal_Characteristics... 11
Set_User_ID... 16
Special_Control_Character_of... 11
Stopped_Child_Signal_Enabled... 13
Synchronize_Data... 123, 126
Synchronize_File... 123
Synchronize_Memory... 123, 126
Terminal_Modes_Of... 11
Truncate_File... 123, 125
Unblock_Signals... 15
Unignore_Signal... 15
Unlink... 13

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Unlock_Shared_Memory... 125
Unmap_Memory... 123
Value... 10, 43, 61, 80, 98
Write... 11, 46

Summary of Profile Features... 17
swab() function... 9
swapcontext() function... 9
swprintf() function... 6
swscanf() function... 6
symlink() function... 8
sync() function... 9
Synchronize_Data subprogram... 123, 126
Synchronize_File subprogram... 123
Synchronize_Memory subprogram... 123,
  126
Synchronized I/O option... 21, 42, 60, 79, 98,
  123, 126
sysconf() function... 8, 44, 62, 80
syslog() function... 9
System package... 10
system() function... 8
System_Storage_Elements package... 10

t tabs utility... 122
talk utility... 122
tan() function... 5
tanf() function... 5
tanh() function... 5
tanhf() function... 5
tanhl() function... 5
tanl() function... 5
tcdrain() function... 7
tcflow() function... 7
tcflush() function... 7
tcgetattr() function... 7
tcgetpgrp() function... 7
tcgetsid() function... 9
tcsendbreak() function... 7
tcsetattr() function... 7
tcsetpgrp() function... 7
tdelete() function... 9
telldir() function... 9
tempnam() function... 9
Terminal_Modes_Of subprogram... 11
Terminology... 25
tfind() function... 9
tgamma() function... 5
tgammal() function... 5
time utility... 122
time() function... 6, 44, 62
Timeouts option... 52, 70, 88, 107
timer_create() function... 118
timer_delete() function... 118
timer_getoverrun() function... 44, 62, 118
timer_gettime() function... 44, 62, 118
timer_settime() function... 44, 62, 118
Timers option... 21, 42, 60, 79, 98, 126
times() function... 7, 44, 62
timezone() function... 9
tmpfile() function... 7
tmpnam() function... 7
toascii() function... 9	olower() function... 6
toupper() function... 6
towtrans() function... 6
towlower() function... 6
towupper() function... 6
tput utility... 122
Trace Event Filtering option... 70, 89, 107
Trace Log option... 70, 89, 107
True...True constant... 39, 40, 42, 47, 57, 58,
  60, 63, 64, 75, 76, 79, 82, 93, 94, 97, 100,
  101, 111
trunc() function... 5
truncate() function... 9
Truncate_File subprogram... 123, 125
truncf() function... 5
trunc() function... 5
tsearch() function... 9
ttyname() function... 7	ttyname_r() function... 7, 116
twalk() function... 9
type

  Boolean... 39, 40, 57, 58, 75, 76, 93, 94,
  111
  FILE *... 49, 66, 84, 102
  long... 96
  off_t... 96
  POSIX_Limits.Child_Processes_Maxima'First... 98
  POSIX_Limits.Groups_Maxima'Fi
  rst... 98
  POSIX_Limits.Realtime_Signals_Maxima'Fi
  rst... 42, 60, 79,
  98
  POSIX_Limits.Timers_Maxima'Fi
  rst... 42, 60, 79, 98
  POSIX_Profiles... 40, 47, 58, 63, 76,
  82, 94, 100
POSIX® REALTIME AND EMBEDDED APPLICATION PROFILES IEEE Std P1003.13-Draft 2.1, Feb 2003

POSIX_Profiles.Realtime_Controller... 57
POSIX_Profiles.Realtime_Dedicated... 75
POSIX_Profiles.Realtime_Lang_Ada95... 40, 47, 58, 64, 76, 82, 94, 101
POSIX_Profiles.Realtime_Minimal... 39
POSIX_Profiles.Realtime_Multi... 93
Realtime_Lang_C99... 40, 47, 58, 63, 76, 82, 94, 100
Signal... 15
Signal_Event... 15
Signal_Info... 126
sigset... 9
tzname() function... 6
tzset() function... 6

U
unalarm() function... 9
ulimit() function... 9
unmask() function... 7
unalias utility... 122
uname() function... 8, 44, 62
Unblock_Signals subprogram... 15
Unbounded_Priority_Inversion... 28
undefined... 36
terminology... 26
unexpand utility... 122
ungetc() function... 6
ungetc() function... 9
Unignore_Signal subprogram... 15
unit of functionality... 29
POSIX_ADA_LANG_SUPPORT... 10, 17, 42, 60, 78, 97
POSIX_C_LANG_JUMP... 4, 17, 40, 58, 76, 94
POSIX_C_LANG_MAT... 58
POSIX_C_LANG_MATH... 5, 17, 55, 76, 94
POSIX_C_LANG_SUPP... 58
POSIX_C_LANG_SUPPORT... 6, 17, 40, 76, 94
POSIX_C_LANG_WIDE_CHAR... 6, 17, 55, 72, 90, 94
POSIX_C_LIB_EXT... 40
POSIX_DEVICE_IO... 6, 11, 17, 40, 42, 58, 60, 76, 78, 94, 97
POSIX_DEVICE_SPECIFIC... 7, 11, 17, 94, 97
POSIX_EVENT_MGMT... 7, 11, 17, 76, 78, 94, 97
POSIX_FD_MGMT... 7, 12, 17, 49, 58, 60, 76, 78, 94, 97
POSIX_FIFO... 7, 12, 17, 49, 66, 84, 94, 97
POSIX_FILE_ATTRIBUTES... 7, 12, 17, 49, 66, 84, 94, 97
POSIX_FILE_LOCKING... 7, 17, 40, 42, 58, 76, 94
POSIX_FILE_SYSTEM... 7, 12, 17, 49, 58, 60, 76, 78, 94, 97
POSIX_FILE_SYSTEM_EXT... 7, 17, 49, 66, 84, 94
POSIX_JOB_CONTROL... 7, 13, 17, 22, 94, 97, 114
POSIX_MULTI_PROCESS... 7, 13, 17, 76, 78, 94, 97
POSIX_NETWORKING... 7, 14, 17, 76, 78, 94, 97
POSIX_PIPE... 7, 17, 76, 78, 95, 97
POSIX_PIPES... 14
POSIX_PRIORITY_RANGES... 7, 14, 17, 40, 51, 58, 67, 86, 104
POSIX_REGEXP... 7, 17, 54, 71, 89, 95, 108, 115
POSIX_RW_LOCKS... 8, 10, 17, 115
POSIX_SHELL_FUNC... 8, 17, 54, 71, 89, 95, 108
POSIX_SIGNAL_JUMP... 8, 17, 76, 95
POSIX_SIGNALS... 8, 15, 17, 40, 42, 58, 60, 76, 78, 95, 97
POSIX_SINGLE_PROCESS... 8, 15, 17, 40, 42, 58, 60, 76, 78, 95, 97
POSIX_STRING_MATCHING... 8, 17, 95
POSIX_SYMBOLIC_LINKS... 8, 17, 95
POSIX_SYSTEM_DATABASE... 8, 16, 17, 95, 97
POSIX_THREADS_BASE... 8, 10, 17, 40, 53, 58, 70, 76, 88, 95, 107
POSIX_USER_GROUPS... 9, 16, 17, 95, 97
POSIX_WIDE_CHAR_IO... 9, 17, 95
XSI_C_LANG_SUPPORT... 9, 17, 54, 71, 90, 108, 121
XSI_DBM... 9, 17, 54, 71, 90, 108, 121
XSI_DEVICE_IO... 9, 18, 54, 71, 90, 108, 121
XSI_DEVICE_SPECIFIC... 9, 18, 54, 71, 90, 108, 121
XSI_DYNAMIC_LINKING... 9, 18, 54, 71, 90, 95, 108, 121
XSI_FD_MGMT... 9, 18, 54, 71, 90, 108, 121

Copyright © 2003 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
Use_Error exception... 45, 63, 81
utility
  usleep() function... 9
du... 122
  ex... 122
expand... 122
  fc... 122
fg... 122
file... 122
  for77... 122
fsck... xvii
  jobs... 122
lex... 121
make... 122
msg... 122
more... 122
newgrp... 122
nice... 122
nm... 122
patch... 122
ps... 122
qalter... 122
qdel... 122
qhold... 122
qmove... 122
qmsg... 122
qrerun... 122
qrls... 122
qselect... 122
qstat... 122
qsub... 122
renice... 122
sh... 121
split... 122
strings... 122
strip... 122
tabs... 122
talk... 122
time... 122
tput... 122
unalias... 122
unexpand... 122
uuencode... 122
uuencode... 122
vi... 122
who... 122
write... 122
yacc... 121
utime() function... 7
utimes() function... 120
uudecode utility... 122
uuencode utility... 122

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

va_arg() function... 6
va_copy() function... 6
va_end() function... 6
va_start() function... 6
Value subprogram... 10, 43, 61, 80, 98
vfork() function... 9
vfprintf() function... 6, 44
vfscanf() function... 6, 44
vfscanf() function... 9
vprintf() function... 6, 44
vscanf() function... 6, 44
vsnprintf() function... 6
vsprintf() function... 6
vsscanf() function... 6
vswprintf() function... 6
vswscanf() function... 6
vwprintf() function... 9
vwscanf() function... 9

W

wait() function... 7
waitid() function... 9
waitpid() function... 7
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
wait3() function... 6
wait4() function... 6
write() function... xv, xvii, 6, 44
writev() function... 9
writes() function... 6
write() function... xv, xvii, 6, 44
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, 108, 121
XSI_FILESYSTEM unit of functionality... 9,
18, 54, 71, 90, 108, 121
XSI_FILESYSTEM unit of functionality... 9,
18, 54, 71, 90, 108, 121
XSI_JOB_CONTROL unit of functionality... 9,
18, 54, 71, 90, 108, 121
XSI_JUMP unit of functionality... 9, 18, 54, 71,
XSI_MATH unit of functionality... 9, 18, 54, 71, 90, 108, 121
XSI_MULTI_PROCESS unit of functionality... 9, 18, 54, 71, 90, 108, 121
XSI_SIGNALS unit of functionality... 9, 18, 54, 71, 90, 108, 121
XSI_SINGLE_PROCESS unit of functionality... 9, 18, 54, 71, 90, 108, 121
XSI_SYSTEM_DATABASE unit of functionality... 9, 18, 54, 71, 90, 108, 121
XSI_SYSTEM_LOGGING unit of functionality... 9, 18, 54, 71, 90, 95, 109, 121
XSI_THREAD_MUTEX_EXT unit of functionality... 10, 18, 40, 54, 58, 72, 76, 90, 95, 109, 121
XSI_THREADS_EXT unit of functionality... 10, 18, 40, 54, 58, 72, 77, 90, 95, 109, 121
XSI_TIMERS unit of functionality... 10, 18, 54, 71, 90, 108, 121
XSI_USER_GROUPS unit of functionality... 10, 18, 54, 71, 90, 108, 121
XSI_WIDE_CHAR unit of functionality... 10, 18, 54, 71, 90, 108, 121
XTI Detailed Network Interface option... 126

Y

y0() function... 9
y1() function... 9
yacc utility... 121
yn() function... 9

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