# RTEMS POSIX 1003.1 Compliance Guide

Edition 4.10.0, for RTEMS 4.10.0

8 February 2011

**On-Line Applications Research Corporation**

On-Line Applications Research Corporation
TEXinfo 2009-08-14.15

COPYRIGHT © 1988 - 2008.
On-Line Applications Research Corporation (OAR).

The authors have used their best efforts in preparing this material. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. No warranty of any kind, expressed or implied, with regard to the software or the material contained in this document is provided. No liability arising out of the application or use of any product described in this document is assumed. The authors reserve the right to revise this material and to make changes from time to time in the content hereof without obligation to notify anyone of such revision or changes.

The RTEMS Project is hosted at `http://www.rtems.com`. Any inquiries concerning RTEMS, its related support components, its documentation, or any custom services for RTEMS should be directed to the contacts listed on that site. A current list of RTEMS Support Providers is at `http://www.rtems.com/support.html`.

# Table of Contents

**Preface** ..... 1

**1 General** ..... 3
1.1 Scope ..... 3
1.2 Normative References ..... 3
1.3 Conformance ..... 3

**2 Terminology and General Requirements** ..... 5
2.1 Conventions ..... 5
2.2 Definitions ..... 5
2.3 General Concepts ..... 5
2.4 Error Numbers ..... 5
2.5 Primitive System Types ..... 6
2.6 Environment Description ..... 6
2.7 C Language Definitions ..... 6
2.7.1 Symbols From the C Standard ..... 6
2.7.2 POSIX.1 Symbols ..... 6
2.8 Numerical Limits ..... 6
2.9 C Language Limits ..... 6
2.9.1 Minimum Values ..... 7
2.9.2 Run-Time Increasable Values ..... 8
2.9.3 Run-Time Invariant Values (Possible Indeterminate) ..... 8
2.9.4 Pathname Variable Values ..... 8
2.9.5 Invariant Values ..... 8
2.9.6 Maximum Values ..... 8
2.10 Symbolic Constants ..... 8
2.10.1 Symbolic Constants for the access Function ..... 8
2.10.2 Symbolic Constants for the lseek Function ..... 9
2.10.3 Compile-Time Symbolic Constants for Portability Specifications ..... 9
2.10.4 Execution-Time Symbolic Constants for Portability Specifications ..... 9

**3 Process Primitives** ..... 11
3.1 Process Creation and Execution ..... 11
3.1.1 Process Creation ..... 11
3.1.2 Execute a File ..... 11
3.1.3 Register Fork Handlers ..... 11
3.2 Process Termination ..... 11
3.2.1 Wait for Process Termination ..... 11
3.2.2 Terminate a Process ..... 11
3.3 Signals ..... 11

3.3.1 Signal Concepts ..... 11
3.3.1.1 Signal Names ..... 11
3.3.1.2 Signal Generation and Delivery ..... 12
3.3.1.3 Signal Actions ..... 12
3.3.2 Send a Signal to a Process ..... 12
3.3.3 Manipulate Signal Sets ..... 12
3.3.4 Examine and Change Signal Action ..... 13
3.3.5 Examine and Change Blocked Signals ..... 13
3.3.6 Examine Pending Signals ..... 13
3.3.7 Wait for a Signal ..... 13
3.3.8 Synchronously Accept a Signal ..... 13
3.3.9 Queue a Signal to a Process ..... 13
3.3.10 Send a Signal to a Thread ..... 13
3.4 Timer Operations ..... 13
3.4.1 Schedule Alarm ..... 13
3.4.2 Suspend Process Execution ..... 13
3.4.3 Delay Process Execution ..... 13

**4 Process Environment ..... 15**
4.1 Process Identification ..... 15
4.1.1 Get Process and Parent Process IDs ..... 15
4.2 User Identification ..... 15
4.2.1 Get Real User Effective User Real Group and Effective Group IDs ..... 15
4.2.2 Set User and Group IDs ..... 15
4.2.3 Get Supplementary Group IDs ..... 15
4.2.4 Get User Name ..... 15
4.3 Process Groups ..... 15
4.3.1 Get Process Group ID ..... 15
4.3.2 Create Session and Set Process Group ID ..... 15
4.3.3 Set Process Group ID for Job Control ..... 15
4.4 System Identification ..... 15
4.4.1 Get System Name ..... 15
4.5 Time ..... 16
4.5.1 Get System Time ..... 16
4.5.2 Get Process Times ..... 16
4.6 Environment Variables ..... 16
4.6.1 Environment Access ..... 16
4.7 Terminal Identification ..... 16
4.7.1 Generate Terminal Pathname ..... 16
4.7.2 Determine Terminal Device Name ..... 16
4.8 Configurable System Variables ..... 16
4.8.1 Get Configurable System Variables ..... 16

**5 Files and Directories ... 19**

5.1 Directories ... 19
5.1.1 Format of Directory Entries ... 19
5.1.2 Directory Operations ... 19
5.2 Working Directory ... 19
5.2.1 Change Current Working Directory ... 19
5.2.2 Get Working Directory Pathname ... 19
5.3 General File Creation ... 19
5.3.1 Open a File ... 19
5.3.2 Create a New File or Rewrite an Existing One ... 19
5.3.3 Set File Creation Mask ... 20
5.3.4 Link to a File ... 20
5.4 Special File Creation ... 20
5.4.1 Make a Directory ... 20
5.4.2 Make a FIFO Special File ... 20
5.5 File Removal ... 20
5.5.1 Remove Directory Entries ... 20
5.5.2 Remove a Directory ... 20
5.5.3 Rename a File ... 20
5.6 File Characteristics ... 20
5.6.1 File Characteristics Header and Data Structure ... 20
5.6.1.1 <sys/stat.h> File Types ... 20
5.6.1.2 <sys/stat.h> File Modes ... 21
5.6.1.3 <sys/stat.h> Time Entries ... 21
5.6.2 Get File Status ... 21
5.6.3 Check File Accessibility ... 21
5.6.4 Change File Modes ... 21
5.6.5 Change Owner and Group of a File ... 21
5.6.6 Set File Access and Modification Times ... 21
5.6.7 Truncate a File to a Specified Length ... 21
5.7 Configurable Pathname Variable ... 21
5.7.1 Get Configurable Pathname Variables ... 21

**6 Input and Output Primitives ... 23**

6.1 Pipes ... 23
6.1.1 Create an Inter-Process Channel ... 23
6.2 File Descriptor Manipulation ... 23
6.2.1 Duplicate an Open File Descriptor ... 23
6.3 File Descriptor Deassignment ... 23
6.3.1 Close a File ... 23
6.4 Input and Output ... 23
6.4.1 Read from a File ... 23
6.4.2 Write to a File ... 23
6.5 Control Operations on Files ... 23
6.5.1 Data Definitions for File Control Operations ... 23
6.5.2 File Control ... 23
6.5.3 Reposition Read/Write File Offset ... 24
6.6 File Synchronization ... 24

6.6.1 Synchronize the State of a File ..... 24
6.6.2 Synchronize the Data of a File ..... 24
6.7 Asynchronous Input and Output ..... 24
6.7.1 Data Definitions for Asynchronous Input and Output ..... 24
6.7.1.1 Asynchronous I/O Control Block ..... 24
6.7.1.2 Asynchronous I/O Manifest Constants ..... 24
6.7.2 Asynchronous Read ..... 24
6.7.3 Asynchronous Write ..... 24
6.7.4 List Directed I/O ..... 24
6.7.5 Retrieve Error Status of Asynchronous I/O Operation ..... 25
6.7.6 Retrieve Return Status of Asynchronous I/O Operation ..... 25
6.7.7 Cancel Asynchronous I/O Request ..... 25
6.7.8 Wait for Asynchronous I/O Request ..... 25
6.7.9 Asynchronous File Synchronization ..... 25

**7 Device- and Class-Specific Functions ..... 27**
7.1 General Terminal Interface ..... 27
7.1.1 Interface Characteristics ..... 27
7.1.1.1 Opening a Terminal Device File ..... 27
7.1.1.2 Process Groups (TTY) ..... 27
7.1.1.3 The Controlling Terminal ..... 27
7.1.1.4 Terminal Access Control ..... 27
7.1.1.5 Input Processing and Reading Data ..... 27
7.1.1.6 Canonical Mode Input Processing ..... 27
7.1.1.7 Noncanonical Mode Input Processing ..... 27
7.1.1.8 Writing Data and Output Processing ..... 27
7.1.1.9 Special Characters ..... 27
7.1.1.10 Modem Disconnect ..... 27
7.1.1.11 Closing a Terminal Device File ..... 27
7.1.2 Parameters That Can Be Set ..... 27
7.1.2.1 termios Structure ..... 27
7.1.2.2 Input Modes ..... 28
7.1.2.3 Output Modes ..... 28
7.1.2.4 Control Modes ..... 28
7.1.2.5 Local Modes ..... 28
7.1.2.6 Special Control Characters ..... 28
7.1.3 Baud Rate Values ..... 29
7.1.3.1 Baud Rate Functions ..... 29
7.2 General Terminal Interface Control Functions ..... 29
7.2.1 Get and Set State ..... 29
7.2.2 Line Control Functions ..... 29
7.2.3 Get Foreground Process Group ID ..... 30
7.2.4 Set Foreground Process Group ID ..... 30

**8 Language-Specific Services for the C Programming Language ... 31**

8.1 Referenced C Language Routines ... 31
8.1.1 Extensions to Time Functions ... 34
8.1.2 Extensions to setlocale Function ... 34
8.2 C Language Input/Output Functions ... 34
8.2.1 Map a Stream Pointer to a File Descriptor ... 34
8.2.2 Open a Stream on a File Descriptor ... 34
8.2.3 Interactions of Other FILE-Type C Functions ... 34
8.2.4 Operations on Files - the remove Function ... 34
8.2.5 Temporary File Name - the tmpnam Function ... 34
8.2.6 Stdio Locking Functions ... 34
8.2.7 Stdio With Explicit Client Locking ... 34
8.3 Other C Language Functions ... 34
8.3.1 Nonlocal Jumps ... 34
8.3.2 Set Time Zone ... 35
8.3.3 Find String Token ... 35
8.3.4 ASCII Time Representation ... 35
8.3.5 Current Time Representation ... 35
8.3.6 Coordinated Universal Time ... 35
8.3.7 Local Time ... 35
8.3.8 Pseudo-Random Sequence Generation Functions ... 35

**9 System Databases ... 37**

9.1 System Databases Section ... 37
9.2 Database Access ... 37
9.2.1 Group Database Access ... 37
9.2.2 User Database Access ... 37

**10 Data Interchange Format ... 39**

10.1 Archive/Interchange File Format ... 39
10.1.1 Extended tar Format ... 39
10.1.2 Extended cpio Format ... 39
10.1.3 Multiple Volumes ... 40

**11 Synchronization ... 41**

11.1 Semaphore Characteristics ... 41
11.2 Semaphore Functions ... 41
11.2.1 Initialize an Unnamed Semaphore ... 41
11.2.2 Destroy an Unnamed Semaphore ... 41
11.2.3 Initialize/Open a Named Semaphore ... 41
11.2.4 Close a Named Semaphore ... 41
11.2.5 Remove a Named Semaphore ... 41
11.2.6 Lock a Semaphore ... 41
11.2.7 Unlock a Semaphore ... 41
11.2.8 Get the Value of a Semaphore ... 41
11.3 Mutexes ... 41

11.3.1 Mutex Initialization Attributes ..... 41
11.3.2 Initializing and Destroying a Mutex ..... 42
11.3.3 Locking and Unlocking a Mutex ..... 42
11.4 Condition Variables ..... 42
11.4.1 Condition Variable Initialization Attributes ..... 42
11.4.2 Initialization and Destroying Condition Variables ..... 42
11.4.3 Broadcasting and Signaling a Condition ..... 42
11.4.4 Waiting on a Condition ..... 42

**12 Memory Management ..... 43**

12.1 Memory Locking Functions ..... 43
12.1.1 Lock/Unlock the Address Space of a Process ..... 43
12.1.2 Lock/Unlock a Rand of Process Address Space ..... 43
12.2 Memory Mapping Functions ..... 43
12.2.1 Map Process Addresses to a Memory Object ..... 43
12.2.2 Unmap Previously Mapped Addresses ..... 43
12.2.3 Change Memory Protection ..... 43
12.2.4 Memory Object Synchronization ..... 43
12.3 Shared Memory Functions ..... 43
12.3.1 Open a Shared Memory Object ..... 43
12.3.2 Remove a Shared Memory Object ..... 43

**13 Execution Scheduling ..... 45**

13.1 Scheduling Parameters ..... 45
13.2 Scheduling Policies ..... 45
13.2.1 SCHED_FIFO ..... 45
13.2.2 SCHED_RR ..... 45
13.2.3 SCHED_OTHER ..... 45
13.3 Process Scheduling Functions ..... 45
13.3.1 Set Scheduling Parameters ..... 45
13.3.2 Get Scheduling Parameters ..... 45
13.3.3 Set Scheduling Policy and Scheduling Parameters ..... 45
13.3.4 Get Scheduling Policy ..... 45
13.3.5 Yield Processor ..... 45
13.3.6 Get Scheduling Parameter Limits ..... 45
13.4 Thread Scheduling ..... 46
13.4.1 Thread Scheduling Attributes ..... 46
13.4.2 Scheduling Contention Scope ..... 46
13.4.3 Scheduling Allocation Domain ..... 46
13.4.4 Scheduling Documentation ..... 46
13.5 Thread Scheduling Functions ..... 46
13.5.1 Thread Creation Scheduling Attributes ..... 46
13.5.2 Dynamic Thread Scheduling Parameters Access ..... 46
13.6 Synchronization Scheduling ..... 46
13.6.1 Mutex Initialization Scheduling Attributes ..... 46
13.6.2 Change the Priority Ceiling of a Mutex ..... 46

**14 Clocks and Timers ..... 47**
14.1 Data Definitions for Clocks and Timers ..... 47
14.1.1 Time Value Specification Structures ..... 47
14.1.2 Timer Event Notification Control Block ..... 47
14.1.3 Type Definitions ..... 47
14.1.4 Timer Event Notification Manifest Constants ..... 47
14.2 Clock and Timer Functions ..... 47
14.2.1 Clocks ..... 47
14.2.2 Create a Per-Process Timer ..... 47
14.2.3 Delete a Per-Process Timer ..... 47
14.2.4 Per-Process Timers ..... 47
14.2.5 High Resolution Sleep ..... 47

**15 Message Passing ..... 49**
15.1 Data Definitions for Message Queues ..... 49
15.1.1 Data Structures ..... 49
15.2 Message Passing Functions ..... 49
15.2.1 Open a Message Queue ..... 49
15.2.2 Close a Message Queue ..... 49
15.2.3 Remove a Message Queue ..... 49
15.2.4 Send a Message to a Message Queue ..... 49
15.2.5 Receive a Message From a Message Queue ..... 49
15.2.6 Notify Process That a Message is Available on a Queue ..... 49
15.2.7 Set Message Queue Attributes ..... 49
15.2.8 Get Message Queue Attributes ..... 49

**16 Thread Management ..... 51**
16.1 Threads ..... 51
16.2 Thread Functions ..... 51
16.2.1 Thread Creation Attributes ..... 51
16.2.2 Thread Creation ..... 51
16.2.3 Wait for Thread Termination ..... 51
16.2.4 Detaching a Thread ..... 51
16.2.5 Thread Termination ..... 51
16.2.6 Get Thread ID ..... 51
16.2.7 Compare Thread IDs ..... 51
16.2.8 Dynamic Package Initialization ..... 51

**17 Thread-Specific Data ..... 53**
17.1 Thread-Specific Data Functions ..... 53
17.1.1 Thread-Specific Data Key Creation ..... 53
17.1.2 Thread-Specific Data Management ..... 53
17.1.3 Thread-Specific Data Key Deletion ..... 53

**18 Thread Cancellation ..... 55**
18.1 Thread Cancellation Overview ..... 55
18.1.1 Cancelability States ..... 55
18.1.2 Cancellation Points ..... 55
18.1.3 Thread Cancellation Cleanup Handlers ..... 55
18.1.4 Async-Cancel Safety ..... 55
18.2 Thread Cancellation Functions ..... 55
18.2.1 Canceling Execution of a Thread ..... 55
18.2.2 Setting Cancelability State ..... 55
18.2.3 Establishing Cancellation Handlers ..... 55
18.3 Language-Independent Cancellation Functionality ..... 55
18.3.1 Requesting Cancellation ..... 55
18.3.2 Associating Cleanup Code With Scopes ..... 55
18.3.3 Controlling Cancellation Within Scopes ..... 55
18.3.4 Defined Cancellation Sequence ..... 55
18.3.5 List of Cancellation Points ..... 55

**19 Compliance Summary ..... 57**
19.1 General Chapter ..... 57
19.2 Terminology and General Requirements Chapter ..... 58
19.3 Process Primitives Chapter ..... 59
19.4 Process Environment Chapter ..... 60
19.5 Files and Directories Chapter ..... 61
19.6 Input and Output Primitives Chapter ..... 62
19.7 Device- and Class-Specific Functions Chapter ..... 63
19.8 Language-Specific Services for the C Programming Language Chapter ..... 64
19.9 System Databases Chapter ..... 65
19.10 Data Interchange Format Chapter ..... 66
19.11 Synchronization Chapter ..... 67
19.12 Memory Management Chapter ..... 68
19.13 Execution Scheduling Chapter ..... 69
19.14 Clocks and Timers Chapter ..... 70
19.15 Message Passing Chapter ..... 71
19.16 Thread Management Chapter ..... 72
19.17 Thread-Specific Data Chapter ..... 73
19.18 Thread Cancellation Chapter ..... 74
19.19 Overall Summary ..... 75

**Command and Variable Index ..... 77**

**Concept Index ..... 79**

# Preface

This document lists the functions, constant, macros, feature flags, and types defined in the POSIX 1003.1 standard. Each section in this document corresponds to a section in the 1003.1 standard and the implementation status of the items required by the standard are listed.

RTEMS supports a number of POSIX process, user, and group oriented routines in what is referred to as a "SUSP" (Single-User, Single Process) manner. RTEMS supports a single process, multithreaded POSIX 1003.1b environment. In a pure world, there would be no reason to even include routines like `getpid()` when there can only be one process. But providing routines like `getpid()` and making them work in a sensible fashion for an embedded environment while not returning ENOSYS (for not implemented) makes it significantly easier to port code from a UNIX environment without modifying it.

# 1 General

## 1.1 Scope

## 1.2 Normative References

## 1.3 Conformance

NGROUPS_MAX, Feature Flag,
_POSIX_ASYNCHRONOUS_IO, Feature Flag,
_POSIX_CHOWN_RESTRICTED, Feature Flag,
_POSIX_FSYNC, Feature Flag,
_POSIX_JOB_CONTROL, Feature Flag,
_POSIX_MAPPED_FILES, Feature Flag,
_POSIX_MEMLOCK, Feature Flag,
_POSIX_MEMLOCK_RANGE, Feature Flag,
_POSIX_MEMORY_PROTECTION, Feature Flag,
_POSIX_MESSAGE_PASSING, Feature Flag,
_POSIX_PRIORITIZED_IO, Feature Flag,
_POSIX_PRIORITY_SCHEDULING, Feature Flag,
_POSIX_REALTIME_SIGNALS, Feature Flag,
_POSIX_SEMAPHORES, Feature Flag,
_POSIX_SHARED_MEMORY_OBJECTS, Feature Flag,
_POSIX_SYNCHRONIZED_IO, Feature Flag,
_POSIX_TIMERS, Feature Flag,
_POSIX_THREAD_PRIO_INHERIT, Feature Flag,
_POSIX_THREAD_PRIORITY_SCHEDULING, Feature Flag,
_POSIX_THREADS, Feature Flag,
_POSIX_THREAD_SAFE_FUNCTIONS, Feature Flag,

# 2 Terminology and General Requirements

## 2.1 Conventions

## 2.2 Definitions

## 2.3 General Concepts

## 2.4 Error Numbers

E2BIG, Constant, Implemented
EACCES, Constant, Implemented
EAGAIN, Constant, Implemented
EBADF, Constant, Implemented
EBADMSG, Constant, Implemented
EBUSY, Constant, Implemented
ECANCELED, Constant, Unimplemented
ECHILD, Constant, Implemented
EDEADLK, Constant, Implemented
EDOM, Constant, Implemented
EEXIST, Constant, Implemented
EFAULT, Constant, Implemented
EFBIG, Constant, Implemented
EINPROGRESS, Constant, Implemented
EINTR, Constant, Implemented
EINVAL, Constant, Implemented
EIO, Constant, Implemented
EISDIR, Constant, Implemented
EMFILE, Constant, Implemented
EMLINK, Constant, Implemented
EMSGSIZE, Constant, Implemented
ENAMETOOLONG, Constant, Implemented
ENFILE, Constant, Implemented
ENODEV, Constant, Implemented
ENOENT, Constant, Implemented
ENOEXEC, Constant, Implemented
ENOLCK, Constant, Implemented
ENOMEM, Constant, Implemented
ENOSPC, Constant, Implemented
ENOSYS, Constant, Implemented
ENOTDIR, Constant, Implemented
ENOTEMPTY, Constant, Implemented
ENOTSUP, Constant, Implemented
ENOTTY, Constant, Implemented
ENXIO, Constant, Implemented
EPERM, Constant, Implemented

EPIPE, Constant, Implemented
ERANGE, Constant, Implemented
EROFS, Constant, Implemented
ESPIPE, Constant, Implemented
ESRCH, Constant, Implemented
ETIMEDOUT, Constant, Implemented
EXDEV, Constant, Implemented

## 2.5 Primitive System Types

dev_t, Type, Implemented
gid_t, Type, Implemented
ino_t, Type, Implemented
mode_t, Type, Implemented
nlink_t, Type, Implemented
off_t, Type, Implemented
pid_t, Type, Implemented
pthread_t, Type, Implemented
pthread_attr_t, Type, Implemented
pthread_mutex_t, Type, Implemented
pthread_mutex_attr_t, Type, Implemented
pthread_cond_t, Type, Implemented
pthread_cond_attr_t, Type, Implemented
pthread_key_t, Type, Implemented
pthread_once_t, Type, Implemented
size_t, Type, Implemented
ssize_t, Type, Implemented
time_t, Type, Implemented
uid_t, Type, Implemented

NOTE: time_t is not listed in this section but is used by many functions.

## 2.6 Environment Description

## 2.7 C Language Definitions

### 2.7.1 Symbols From the C Standard

NULL, Constant, Implemented

### 2.7.2 POSIX.1 Symbols

_POSIX_C_SOURCE, Feature Flag,

## 2.8 Numerical Limits

## 2.9 C Language Limits

CHAR_BIT, Constant, Implemented
CHAR_MAX, Constant, Implemented

CHAR_MIN, Constant, Implemented
INT_MAX, Constant, Implemented
INT_MIN, Constant, Implemented
LONG_MAX, Constant, Implemented
LONG_MIN, Constant, Implemented
MB_LEN_MAX, Constant, Implemented
SCHAR_MAX, Constant, Implemented
SCHAR_MIN, Constant, Implemented
SHRT_MAX, Constant, Implemented
SHRT_MIN, Constant, Implemented
UCHAR_MAX, Constant, Implemented
UINT_MAX, Constant, Implemented
ULONG_MAX, Constant, Implemented
USHRT_MAX, Constant, Implemented

NOTE: These are implemented in GCC's limits.h file.

### 2.9.1 Minimum Values

_POSIX_AIO_LISTIO_MAX, Constant, Implemented
_POSIX_AIO_MAX, Constant, Implemented
_POSIX_ARG_MAX, Constant, Implemented
_POSIX_CHILD_MAX, Constant, Implemented
_POSIX_DELAYTIMER_MAX, Constant, Implemented
_POSIX_LINK_MAX, Constant, Implemented
_POSIX_LOGIN_NAME_MAX, Constant, Implemented
_POSIX_MAX_CANON, Constant, Implemented
_POSIX_MAX_INPUT, Constant, Implemented
_POSIX_MQ_OPEN_MAX, Constant, Implemented
_POSIX_MQ_PRIO_MAX, Constant, Implemented
_POSIX_NAME_MAX, Constant, Implemented
_POSIX_NGROUPS_MAX, Constant, Implemented
_POSIX_OPEN_MAX, Constant, Implemented
_POSIX_PATH_MAX, Constant, Implemented
_POSIX_PIPE_BUF, Constant, Implemented
_POSIX_RTSIG_MAX, Constant, Implemented
_POSIX_SEM_NSEMS_MAX, Constant, Implemented
_POSIX_SEM_VALUE_MAX, Constant, Implemented
_POSIX_SIGQUEUE_MAX, Constant, Implemented
_POSIX_SSIZE_MAX, Constant, Implemented
_POSIX_STREAM_MAX, Constant, Implemented
_POSIX_THREAD_DESTRUCTOR_ITERATIONS, Constant, Implemented
_POSIX_THREAD_KEYS_MAX, Constant, Implemented
_POSIX_THREAD_THREADS_MAX, Constant, Implemented
_POSIX_TTY_NAME_MAX, Constant, Implemented
_POSIX_TIME_MAX, Constant, Unimplemented
_POSIX_TZNAME_MAX, Constant, Implemented

### 2.9.2 Run-Time Increasable Values

_POSIX_NGROUPS_MAX, Constant, Implemented

### 2.9.3 Run-Time Invariant Values (Possible Indeterminate)

AIO_LISTIO_MAX, Constant, Implemented
AIO_MAX, Constant, Implemented
AIO_PRIO_DELTA_MAX, Constant, Implemented
ARG_MAX, Constant, Implemented
CHILD_MAX, Constant, Implemented
DELAYTIMER_MAX, Constant, Implemented
LOGIN_NAME_MAX, Constant, Implemented
MQ_OPEN_MAX, Constant, Implemented
OPEN_MAX, Constant, Implemented
PAGESIZE, Constant, Implemented
PTHREAD_DESTRUCTOR_ITERATIONS, Constant, Implemented
PTHREAD_KEYS_MAX, Constant, Implemented
PTHREAD_STACK_MIN, Constant, Implemented
PTHJREAD_THREADS_MAX, Constant, Implemented
RTSIG_MAX, Constant, Implemented
SEM_NSEMS_MAX, Constant, Implemented
SEM_VALUE_MAX, Constant, Implemented
SIGQUEUE_MAX, Constant, Implemented
STREAM_MAX, Constant, Implemented
TIMER_MAX, Constant, Implemented
TTY_NAME_MAX, Constant, Implemented
TZNAME_MAX, Constant, Implemented

### 2.9.4 Pathname Variable Values

LINK_MAX, Constant, Implemented
MAX_CANON, Constant, Implemented
MAX_INPUT, Constant, Implemented
NAME_MAX, Constant, Implemented
PATH_MAX, Constant, Implemented
PIPE_BUF, Constant, Implemented

### 2.9.5 Invariant Values

SSIZE_MAX, Constant, Implemented

### 2.9.6 Maximum Values

_POSIX_CLOCKRES_MIN, Constant, Implemented

## 2.10 Symbolic Constants

### 2.10.1 Symbolic Constants for the access Function

R_OK, Constant, Implemented
W_OK, Constant, Implemented

X_OK, Constant, Implemented
F_OK, Constant, Implemented

### 2.10.2 Symbolic Constants for the lseek Function

SEEK_SET, Constant, Implemented
SEEK_CUR, Constant, Implemented
SEEK_END, Constant, Implemented

### 2.10.3 Compile-Time Symbolic Constants for Portability Specifications

_POSIX_ASYNCHRONOUS_IO, Feature Flag,
_POSIX_FSYNC, Feature Flag,
_POSIX_JOB_CONTROL, Feature Flag,
_POSIX_MAPPED_FILES, Feature Flag,
_POSIX_MEMLOCK, Feature Flag,
_POSIX_MEMLOCK_RANGE, Feature Flag,
_POSIX_MEMORY_PROTECTION, Feature Flag,
_POSIX_MESSAGE_PASSING, Feature Flag,
_POSIX_PRIORITIZED_IO, Feature Flag,
_POSIX_PRIORITY_SCHEDULING, Feature Flag,
_POSIX_REALTIME_SIGNALS, Feature Flag,
_POSIX_SAVED_IDS, Feature Flag,
_POSIX_SEMAPHORES, Feature Flag,
_POSIX_SHARED_MEMORY_OBJECTS, Feature Flag,
_POSIX_SYNCHRONIZED_IO, Feature Flag,
_POSIX_THREADS, Feature Flag,
_POSIX_THREAD_ATTR_STACKADDR, Feature Flag,
_POSIX_THREAD_ATTR_STACKSIZE, Feature Flag,
_POSIX_THREAD_PRIORITY_SCHEDULING, Feature Flag,
_POSIX_THREAD_PRIO_INHERIT, Feature Flag,
_POSIX_THREAD_PRIO_CEILING, Feature Flag,
_POSIX_THREAD_PROCESS_SHARED, Feature Flag,
_POSIX_THREAD_SAFE_FUNCTIONS, Feature Flag,
_POSIX_TIMERS, Feature Flag,
_POSIX_VERSION, Feature Flag,

### 2.10.4 Execution-Time Symbolic Constants for Portability Specifications

_POSIX_ASYNC_IO, Feature Flag,
_POSIX_CHOWN_RESTRICTED, Feature Flag,
_POSIX_NO_TRUNC, Feature Flag,
_POSIX_PRIO_IO, Feature Flag,
_POSIX_SYNC_IO, Feature Flag,
_POSIX_VDISABLE, Feature Flag,

# 3 Process Primitives

## 3.1 Process Creation and Execution

### 3.1.1 Process Creation

`fork(), Function, Unimplementable, Requires Processes`

### 3.1.2 Execute a File

`execl(), Function, Unimplementable, Requires Processes`
`execv(), Function, Unimplementable, Requires Processes`
`execle(), Function, Unimplementable, Requires Processes`
`execve(), Function, Unimplementable, Requires Processes`
`execlp(), Function, Unimplementable, Requires Processes`
`execvp(), Function, Unimplementable, Requires Processes`

### 3.1.3 Register Fork Handlers

`pthread_atfork(), Function, Unimplementable, Requires Processes`

## 3.2 Process Termination

### 3.2.1 Wait for Process Termination

`wait(), Function, Unimplementable, Requires Processes`
`waitpid(), Function, Unimplementable, Requires Processes`
`WNOHANG, Constant, Unimplementable, Requires Processes`
`WUNTRACED, Constant, Unimplementable, Requires Processes`
`WIFEXITED(), Function, Unimplementable, Requires Processes`
`WEXITSTATUS(), Function, Unimplementable, Requires Processes`
`WIFSIGNALED(), Function, Unimplementable, Requires Processes`
`WTERMSIG(), Function, Unimplementable, Requires Processes`
`WIFSTOPPED(), Function, Unimplementable, Requires Processes`
`WSTOPSIG(), Function, Unimplementable, Requires Processes`

### 3.2.2 Terminate a Process

`_exit(), Function, Implemented`

## 3.3 Signals

### 3.3.1 Signal Concepts

#### 3.3.1.1 Signal Names

`sigset_t, Type, Implemented`
`SIG_DFL, Constant, Implemented`
`SIG_IGN, Constant, Implemented`
`SIG_ERR, Constant, Implemented`
`SIGABRT, Constant, Implemented`

SIGALRM, Constant, Implemented
SIGFPE, Constant, Implemented
SIGHUP, Constant, Implemented
SIGILL, Constant, Implemented
SIGINT, Constant, Implemented
SIGKILL, Constant, Implemented
SIGPIPE, Constant, Implemented
SIGQUIT, Constant, Implemented
SIGSEGV, Constant, Implemented
SIGTERM, Constant, Implemented
SIGUSR1, Constant, Implemented
SIGUSR2, Constant, Implemented
SIGCHLD, Constant, Unimplemented
SIGCONT, Constant, Unimplemented
SIGSTOP, Constant, Unimplemented
SIGTSTP, Constant, Unimplemented
SIGTTIN, Constant, Unimplemented
SIGTTOU, Constant, Unimplemented
SIGBUS, Constant, Implemented
SIGRTMIN, Constant, Implemented
SIGRTMAX, Constant, Implemented

NOTE: SIG_ERR is technically an extension to the C Library which is not documented anywhere else according to the index.

#### 3.3.1.2 Signal Generation and Delivery

struct sigevent, Type, Implemented
union sigval, Type, Implemented
SIGEV_NONE, Constant, Implemented
SIGEV_SIGNAL, Constant, Implemented
SIGEV_THREAD, Constant, Implemented

#### 3.3.1.3 Signal Actions

siginfo_t, Type, Implemented
SI_USER, Constant, Implemented
SI_QUEUE, Constant, Implemented
SI_TIMER, Constant, Implemented
SI_ASYNCIO, Constant, Implemented
SI_MESGQ, Constant, Implemented

### 3.3.2 Send a Signal to a Process

kill(), Function, Implemented

### 3.3.3 Manipulate Signal Sets

sigemptyset(), Function, Implemented
sigfillset(), Function, Implemented
sigaddset(), Function, Implemented

sigdelset(), Function, Implemented
sigismember(), Function, Implemented

### 3.3.4 Examine and Change Signal Action

sigaction(), Function, Implemented
sigaction, Type, Implemented
SA_NOCLDSTOP, Constant, Implemented
SA_SIGINFO, Constant, Implemented

### 3.3.5 Examine and Change Blocked Signals

pthread_sigmask(), Function, Implemented
sigprocmask(), Function, Implemented
SIG_BLOCK, Constant, Implemented
SIG_UNBLOCK, Constant, Implemented
SIG_SETMASK, Constant, Implemented

### 3.3.6 Examine Pending Signals

sigpending(), Function, Implemented

### 3.3.7 Wait for a Signal

sigsuspend(), Function, Implemented

### 3.3.8 Synchronously Accept a Signal

sigwait(), Function, Implemented
sigwaitinfo(), Function, Implemented
sigtimedwait(), Function, Implemented

### 3.3.9 Queue a Signal to a Process

sigqueue(), Function, Implemented

### 3.3.10 Send a Signal to a Thread

pthread_kill(), Function, Implemented

## 3.4 Timer Operations

### 3.4.1 Schedule Alarm

alarm(), Function, Implemented

### 3.4.2 Suspend Process Execution

pause(), Function, Implemented

### 3.4.3 Delay Process Execution

sleep(), Function, Implemented

# 4 Process Environment

## 4.1 Process Identification

### 4.1.1 Get Process and Parent Process IDs

`getpid(), Function, Implemented, SUSP Functionality`
`getppid(), Function, Implemented, SUSP Functionality`

## 4.2 User Identification

### 4.2.1 Get Real User Effective User Real Group and Effective Group IDs

`getuid(), Function, Implemented, SUSP Functionality`
`geteuid(), Function, Implemented, SUSP Functionality`
`getgid(), Function, Implemented, SUSP Functionality`
`getegid(), Function, Implemented, SUSP Functionality`

### 4.2.2 Set User and Group IDs

`setuid(), Function, Implemented, SUSP Functionality`
`setgid(), Function, Implemented, SUSP Functionality`

### 4.2.3 Get Supplementary Group IDs

`getgroups(), Function, Implemented, SUSP Functionality`

### 4.2.4 Get User Name

`getlogin(), Function, Implemented, SUSP Functionality`
`getlogin_r(), Function, Implemented, SUSP Functionality`

## 4.3 Process Groups

### 4.3.1 Get Process Group ID

`getpgrp(), Function, Implemented, SUSP Functionality`

### 4.3.2 Create Session and Set Process Group ID

`setsid(), Function, Implemented, SUSP Functionality`

### 4.3.3 Set Process Group ID for Job Control

`setpgid(), Function, Dummy Implementation`

## 4.4 System Identification

### 4.4.1 Get System Name

`struct utsname, Type, Implemented`
`uname(), Function, Implemented`

## 4.5 Time

### 4.5.1 Get System Time

`time()`, Function, Implemented

### 4.5.2 Get Process Times

`struct tms`, Type, Implemented
`times()`, Function, Implemented

NOTE: `times` always returns 0 for tms_stime, tms_cutime, and tms_cstime fields of the `struct tms` returned.

## 4.6 Environment Variables

### 4.6.1 Environment Access

`getenv()`, Function, Implemented

## 4.7 Terminal Identification

### 4.7.1 Generate Terminal Pathname

`ctermid()`, Function, Implemented

### 4.7.2 Determine Terminal Device Name

`ttyname()`, Function, Implemented, untested
`ttyname_r()`, Function, Implemented, untested
`isatty()`, Function, Implemented

## 4.8 Configurable System Variables

### 4.8.1 Get Configurable System Variables

`sysconf()`, Function, Dummy Implementation
`_SC_AIO_LISTIO_MAX`, Constant, Implemented
`_SC_AIO_MAX`, Constant, Implemented
`_SC_AIO_PRIO_DELTA_MAX`, Constant, Implemented
`_SC_ARG_MAX`, Constant, Implemented
`_SC_CHILD_MAX`, Constant, Implemented
`_SC_CLK_TCK`, Constant, Implemented
`CLK_TCK`, Constant, Implemented
`_SC_DELAYTIMER_MAX`, Constant, Implemented
`_SC_GETGR_R_SIZE_MAX`, Constant, Implemented
`_SC_GETPW_R_SIZE_MAX`, Constant, Implemented
`_SC_LOGIN_NAME_MAX`, Constant, Implemented
`_SC_MQ_OPEN_MAX`, Constant, Implemented
`_SC_MQ_PRIO_MAX`, Constant, Implemented
`_SC_NGROUPS_MAX`, Constant, Implemented
`_SC_OPEN_MAX`, Constant, Implemented

```
_SC_PAGESIZE, Constant, Implemented
_SC_RTSIG_MAX, Constant, Implemented
_SC_SEM_NSEMS_MAX, Constant, Implemented
_SC_SEM_VALUE_MAX, Constant, Implemented
_SC_SIGQUEUE_MAX, Constant, Implemented
_SC_STREAM_MAX, Constant, Implemented
_SC_THREAD_DESTRUCTOR_ITERATIONS, Constant, Implemented
_SC_THREAD_KEYS_MAX, Constant, Implemented
_SC_THREAD_STACK_MIN, Constant, Implemented
_SC_THREAD_THREADS_MAX, Constant, Implemented
_SC_TIMER_MAX, Constant, Implemented
_SC_TTY_NAME_MAX, Constant, Implemented
_SC_TZNAME_MAX, Constant, Implemented
_SC_ASYNCHRONOUS_IO, Constant, Implemented
_SC_FSYNC, Constant, Implemented
_SC_JOB_CONROL, Constant, Implemented
_SC_MAPPED_FILES, Constant, Implemented
_SC_MEMLOCK, Constant, Implemented
_SC_MEMLOCK_RANGE, Constant, Implemented
_SC_MEMORY_PROTECTION, Constant, Implemented
_SC_MESSAGE_PASSING, Constant, Implemented
_SC_PRIORITIZED_IO, Constant, Implemented
_SC_PRIORITY_SCHEDULING, Constant, Unimplemented
_SC_REALTIME_SIGNALS, Constant, Implemented
_SC_SAVED_IDS, Constant, Implemented
_SC_SEMAPHORES, Constant, Implemented
_SC_SHARED_MEMORY_OBJECTS, Constant, Implemented
_SC_SYNCHRONIZED_IO, Constant, Implemented
_SC_TIMERS, Constant, Implemented
_SC_THREADS, Constant, Implemented
_SC_THREAD_ATTR_STACKADDR, Constant, Implemented
_SC_THREAD_ATTR_STACKSIZE, Constant, Implemented
_SC_THREAD_PRIORITY_SCHEDULING, Constant, Implemented
_SC_THREAD_PRIO_INHERIT, Constant, Implemented
_SC_THREAD_PRIO_PROTECT, Constant, Unimplemented
_SC_THREAD_PROCESS_SHARED, Constant, Implemented
_SC_THREAD_SAFE_FUNCTIONS, Constant, Implemented
_SC_VERSION, Constant, Implemented
```

# 5 Files and Directories

## 5.1 Directories

### 5.1.1 Format of Directory Entries

### 5.1.2 Directory Operations

```
struct dirent, Type, Implemented
opendir(), Function, Implemented
readdir(), Function, Implemented
readdir_r(), Function, Implemented
rewinddir(), Function, Implemented
closedir(), Function, Implemented
```

## 5.2 Working Directory

### 5.2.1 Change Current Working Directory

```
chdir(), Function, Implemented
```

### 5.2.2 Get Working Directory Pathname

```
getcwd(), Function, Implemented
```

## 5.3 General File Creation

### 5.3.1 Open a File

```
open(), Function, Implemented
O_RDONLY, Constant, Implemented
O_WRONLY, Constant, Implemented
O_RDWR, Constant, Implemented
O_APPEND, Constant, Implemented
O_CREAT, Constant, Implemented
O_DSYNC, Constant, Unimplemented
O_EXCL, Constant, Implemented
O_NOCTTY, Constant, Implemented
O_NONBLOCK, Constant, Implemented
O_RSYNC, Constant, Unimplemented
O_SYNC, Constant, Implemented
O_TRUNC, Constant, Implemented
```

NOTE: In the newlib fcntl.h, O_SYNC is defined only if _POSIX_SOURCE is not defined. This seems wrong.

### 5.3.2 Create a New File or Rewrite an Existing One

```
creat(), Function, Implemented
```

### 5.3.3 Set File Creation Mask

`umask(), Function, Implemented`

### 5.3.4 Link to a File

`link(), Function, Implemented`

## 5.4 Special File Creation

### 5.4.1 Make a Directory

`mkdir(), Function, Implemented`

### 5.4.2 Make a FIFO Special File

`mkfifo(), Function, Untested Implementation`

NOTE: mkfifo() is implemented but no filesystem supports FIFOs.

## 5.5 File Removal

### 5.5.1 Remove Directory Entries

`unlink(), Function, Implemented`

### 5.5.2 Remove a Directory

`rmdir(), Function, Implemented`

### 5.5.3 Rename a File

`rename(), Function, Implemented`

## 5.6 File Characteristics

### 5.6.1 File Characteristics Header and Data Structure

`struct stat, Type, Implemented`

#### 5.6.1.1 <sys/stat.h> File Types

```
S_ISBLK(), Function, Implemented
S_ISCHR(), Function, Implemented
S_ISDIR(), Function, Implemented
S_ISFIFO(), Function, Implemented
S_ISREG(), Function, Implemented
S_TYPEISMQ(), Function, Unimplemented
S_TYPEISSEM(), Function, Unimplemented
S_TYPEISSHM(), Function, Unimplemented
```

#### 5.6.1.2 <sys/stat.h> File Modes

S_IRWXU, Constant, Implemented
S_IRUSR, Constant, Implemented
S_IWUSR, Constant, Implemented
S_IXUSR, Constant, Implemented
S_IRWXG, Constant, Implemented
S_IRGRP, Constant, Implemented
S_IWGRP, Constant, Implemented
S_IXGRP, Constant, Implemented
S_IRWXO, Constant, Implemented
S_IROTH, Constant, Implemented
S_IWOTH, Constant, Implemented
S_IXOTH, Constant, Implemented
S_ISUID, Constant, Implemented
S_ISGID, Constant, Implemented

#### 5.6.1.3 <sys/stat.h> Time Entries

### 5.6.2 Get File Status

stat(), Function, Implemented
fstat(), Function, Implemented

### 5.6.3 Check File Accessibility

access(), Function, Implemented

### 5.6.4 Change File Modes

chmod(), Function, Implemented
fchmod(), Function, Implemented

### 5.6.5 Change Owner and Group of a File

chown(), Function, Implemented

### 5.6.6 Set File Access and Modification Times

struct utimbuf, Type, Implemented
utime(), Function, Implemented

### 5.6.7 Truncate a File to a Specified Length

ftruncate(), Function, Implemented

## 5.7 Configurable Pathname Variable

### 5.7.1 Get Configurable Pathname Variables

pathconf(), Function, Implemented
fpathconf(), Function, Implemented
_PC_LINK_MAX, Constant, Implemented
_PC_MAX_CANON, Constant, Implemented

```
_PC_MAX_INPUT, Constant, Implemented
_PC_MAX_INPUT, Constant, Implemented
_PC_NAME_MAX, Constant, Implemented
_PC_PATH_MAX, Constant, Implemented
_PC_PIPE_BUF, Constant, Implemented
_PC_ASYNC_IO, Constant, Implemented
_PC_CHOWN_RESTRICTED, Constant, Implemented
_PC_NO_TRUNC, Constant, Implemented
_PC_PRIO_IO, Constant, Implemented
_PC_SYNC_IO, Constant, Implemented
_PC_VDISABLE, Constant, Implemented
```

NOTE: The newlib unistd.h and sys/unistd.h are installed and the include search patch is used to get the right one. There are conflicts between the newlib unistd.h and RTEMS' version.

# 6 Input and Output Primitives

## 6.1 Pipes

### 6.1.1 Create an Inter-Process Channel

`pipe(), Function, Dummy Implementation`

NOTE: pipe() returns ENOSYS.

## 6.2 File Descriptor Manipulation

### 6.2.1 Duplicate an Open File Descriptor

`dup(), Function, Implemented`
`dup2(), Function, Implemented`

## 6.3 File Descriptor Deassignment

### 6.3.1 Close a File

`close(), Function, Implemented`

## 6.4 Input and Output

### 6.4.1 Read from a File

`read(), Function, Implemented`

### 6.4.2 Write to a File

`write(), Function, Implemented`

## 6.5 Control Operations on Files

### 6.5.1 Data Definitions for File Control Operations

### 6.5.2 File Control

`struct flock, Type, Implemented`
`fcntl(), Function, Implemented`
`F_DUPFD, Constant, Implemented`
`F_GETFD, Constant, Implemented`
`F_GETLK, Constant, Implemented`
`F_SETFD, Constant, Implemented`
`F_GETFL, Constant, Implemented`
`F_SETFL, Constant, Implemented`
`F_SETLK, Constant, Implemented`
`F_SETLKW, Constant, Implemented`
`FD_CLOEXEC, Constant, Implemented`
`F_RDLCK, Constant, Implemented`

F_UNLCK, Constant, Implemented
F_WRLCK, Constant, Implemented
O_ACCMODE, Constant, Implemented

NOTE: A number of constants are used by both `open` and `fcntl`. `O_CREAT`, `O_EXCL`, `O_NOCTTY`, `O_TRUNC`, `O_APPEND`, `O_DSYNC`, `O_NONBLOCK`, `O_RSYNC`, `O_SYNC`, `O_RDONLY`, `O_RDWR`, and `O_WRONLY` are also included in another section. See Section 5.3.1 [Open a File], page 19.

### 6.5.3 Reposition Read/Write File Offset

lseek(), Function, Implemented
SEEK_SET, Constant, Implemented
SEEK_CUR, Constant, Implemented
SEEK_END, Constant, Implemented

## 6.6 File Synchronization

### 6.6.1 Synchronize the State of a File

fsync(), Function, Implemented

### 6.6.2 Synchronize the Data of a File

fdatasync(), Function, Implemented

## 6.7 Asynchronous Input and Output

### 6.7.1 Data Definitions for Asynchronous Input and Output

#### 6.7.1.1 Asynchronous I/O Control Block

struct aiocb, Type, Untested Implementation

#### 6.7.1.2 Asynchronous I/O Manifest Constants

AIO_CANCELED, Constant, Implemented
AIO_NOTCANCELED, Constant, Implemented
AIO_ALLDONE, Constant, Implemented
LIO_WAIT, Constant, Implemented
LIO_NOWAIT, Constant, Implemented
LIO_READ, Constant, Implemented
LIO_WRITE, Constant, Implemented
LIO_NOP, Constant, Implemented

### 6.7.2 Asynchronous Read

aio_read(), Function, Dummy Implementation

### 6.7.3 Asynchronous Write

aio_write(), Function, Dummy Implementation

### 6.7.4 List Directed I/O

lio_listio(), Function, Dummy Implementation

### 6.7.5 Retrieve Error Status of Asynchronous I/O Operation

`aio_error()`, Function, Dummy Implementation

### 6.7.6 Retrieve Return Status of Asynchronous I/O Operation

`aio_return()`, Function, Dummy Implementation

### 6.7.7 Cancel Asynchronous I/O Request

`aio_cancel()`, Function, Dummy Implementation

### 6.7.8 Wait for Asynchronous I/O Request

`aio_suspend()`, Function, Dummy Implementation

### 6.7.9 Asynchronous File Synchronization

`aio_fsync()`, Function, Dummy Implementation

# 7 Device- and Class-Specific Functions

## 7.1 General Terminal Interface

### 7.1.1 Interface Characteristics

#### 7.1.1.1 Opening a Terminal Device File

#### 7.1.1.2 Process Groups (TTY)

#### 7.1.1.3 The Controlling Terminal

#### 7.1.1.4 Terminal Access Control

#### 7.1.1.5 Input Processing and Reading Data

#### 7.1.1.6 Canonical Mode Input Processing

#### 7.1.1.7 Noncanonical Mode Input Processing

- Case A - MIN > 0 and TIME > 0
- Case B - MIN > 0 and TIME = 0
- Case C - MIN = 0 and TIME > 0
- Case D - MIN = 0 and TIME = 0

#### 7.1.1.8 Writing Data and Output Processing

#### 7.1.1.9 Special Characters

```
INTR, Constant, Implemented
QUIT, Constant, Implemented
ERASE, Constant, Implemented
KILL, Constant, Implemented
EOF, Constant, Implemented
NL, Constant, Implemented
EOL, Constant, Implemented
SUSP, Constant, Implemented
STOP, Constant, Implemented
START, Constant, Implemented
CR, Constant, Implemented
```

#### 7.1.1.10 Modem Disconnect

#### 7.1.1.11 Closing a Terminal Device File

### 7.1.2 Parameters That Can Be Set

#### 7.1.2.1 termios Structure

```
tcflag_t, Type, Implemented
cc_t, Type, Implemented
struct termios, Type, Implemented
```

#### 7.1.2.2 Input Modes

BRKINT, Constant, Implemented
ICRNL, Constant, Implemented
IGNBREAK, Constant, Unimplemented
IGNCR, Constant, Implemented
IGNPAR, Constant, Implemented
INLCR, Constant, Implemented
INPCK, Constant, Implemented
ISTRIP, Constant, Implemented
IXOFF, Constant, Implemented
IXON, Constant, Implemented
PARMRK, Constant, Implemented

#### 7.1.2.3 Output Modes

OPOST, Constant, Implemented

#### 7.1.2.4 Control Modes

CLOCAL, Constant, Implemented
CREAD, Constant, Implemented
CSIZE, Constant, Implemented
CS5, Constant, Implemented
CS6, Constant, Implemented
CS7, Constant, Implemented
CS8, Constant, Implemented
CSTOPB, Constant, Implemented
HUPCL, Constant, Implemented
PARENB, Constant, Implemented
PARODD, Constant, Implemented

#### 7.1.2.5 Local Modes

ECHO, Constant, Implemented
ECHOE, Constant, Implemented
ECHOK, Constant, Implemented
ECHONL, Constant, Implemented
ICANON, Constant, Implemented
IEXTEN, Constant, Implemented
ISIG, Constant, Implemented
NOFLSH, Constant, Implemented
TOSTOP, Constant, Implemented

#### 7.1.2.6 Special Control Characters

VEOF, Constant, Implemented
VEOL, Constant, Implemented
VERASE, Constant, Implemented
VINTR, Constant, Implemented
VKILL, Constant, Implemented

VQUIT, Constant, Implemented
VSUSP, Constant, Implemented
VSTART, Constant, Implemented
VSTOP, Constant, Implemented
VMIN, Constant, Implemented
VTIME, Constant, Implemented

### 7.1.3 Baud Rate Values

B0, Constant, Implemented
B50, Constant, Implemented
B75, Constant, Implemented
B110, Constant, Implemented
B134, Constant, Implemented
B150, Constant, Implemented
B200, Constant, Implemented
B300, Constant, Implemented
B600, Constant, Implemented
B1200, Constant, Implemented
B1800, Constant, Implemented
B2400, Constant, Implemented
B4800, Constant, Implemented
B9600, Constant, Implemented
B19200, Constant, Implemented
B38400, Constant, Implemented

#### 7.1.3.1 Baud Rate Functions

cfgetospeed(), Function, Implemented
cfsetospeed(), Function, Implemented
cfgetispeed(), Function, Implemented
cfsetispeed(), Function, Implemented
TCIFLUSH, Constant, Implemented
TCOFLUSH, Constant, Implemented
TCIOFLUSH, Constant, Implemented
TCOOFF, Constant, Implemented
TCOON, Constant, Implemented
TCIOOFF, Constant, Implemented
TCIOON, Constant, Implemented

## 7.2 General Terminal Interface Control Functions

### 7.2.1 Get and Set State

tcgetattr(), Function, Implemented
tcsetattr(), Function, Implemented

### 7.2.2 Line Control Functions

tcsendbreak(), Function, Dummy Implementation

```
tcdrain(), Function, Implemented
tcflush(), Function, Dummy Implementation
tcflow(), Function, Dummy Implementation
```

### 7.2.3 Get Foreground Process Group ID

```
tcgetprgrp(), Function, Implemented, SUSP
```

### 7.2.4 Set Foreground Process Group ID

```
tcsetprgrp(), Function, Dummy Implementation
```

# 8 Language-Specific Services for the C Programming Language

## 8.1 Referenced C Language Routines

ANSI C Section 4.2 — Diagnostics

```
assert(), Function, Implemented
```

ANSI C Section 4.3 — Character Handling

```
isalnum(), Function, Implemented
isalpha(), Function, Implemented
iscntrl(), Function, Implemented
isdigit(), Function, Implemented
isgraph(), Function, Implemented
islower(), Function, Implemented
isprint(), Function, Implemented
ispunct(), Function, Implemented
isspace(), Function, Implemented
isupper(), Function, Implemented
isxdigit(), Function, Implemented
tolower(), Function, Implemented
toupper(), Function, Implemented
```

ANSI C Section 4.4 — Localization

```
setlocale(), Function, Implemented
```

ANSI C Section 4.5 — Mathematics

```
acos(), Function, Implemented
asin(), Function, Implemented
atan(), Function, Implemented
atan2(), Function, Implemented
cos(), Function, Implemented
sin(), Function, Implemented
tan(), Function, Implemented
cosh(), Function, Implemented
sinh(), Function, Implemented
tanh(), Function, Implemented
exp(), Function, Implemented
frexp(), Function, Implemented
ldexp(), Function, Implemented
log(), Function, Implemented
log10(), Function, Implemented
modf(), Function, Implemented
pow(), Function, Implemented
sqrt(), Function, Implemented
ceil(), Function, Implemented
fabs(), Function, Implemented
```

```
floor(), Function, Implemented
fmod(), Function, Implemented
```

ANSI C Section 4.6 — Non-Local Jumps

```
setjmp(), Function, Implemented
longjmp(), Function, Implemented
```

ANSI C Section 4.9 — Input/Output

```
FILE, Type, Implemented
clearerr(), Function, Implemented
fclose(), Function, Implemented
feof(), Function, Implemented
ferror(), Function, Implemented
fflush(), Function, Implemented
fgetc(), Function, Implemented
fgets(), Function, Implemented
fopen(), Function, Implemented
fputc(), Function, Implemented
fputs(), Function, Implemented
fread(), Function, Implemented
freopen(), Function, Implemented
fseek(), Function, Implemented
ftell(), Function, Implemented
fwrite(), Function, Implemented
getc(), Function, Implemented
getchar(), Function, Implemented
gets(), Function, Implemented
perror(), Function, Implemented
printf(), Function, Implemented
fprintf(), Function, Implemented
sprintf(), Function, Implemented
putc(), Function, Implemented
putchar(), Function, Implemented
puts(), Function, Implemented
remove(), Function, Implemented
rewind(), Function, Implemented
scanf(), Function, Implemented
fscanf(), Function, Implemented
sscanf(), Function, Implemented
setbuf(), Function, Implemented
tmpfile(), Function, Implemented
tmpnam(), Function, Implemented
ungetc(), Function, Implemented
```

NOTE: `rename` is also included in another section. Section 5.5.3 [Rename a File], page 20.

ANSI C Section 4.10 — General Utilities

```
abs(), Function, Implemented
atof(), Function, Implemented
atoi(), Function, Implemented
atol(), Function, Implemented
rand(), Function, Implemented
srand(), Function, Implemented
calloc(), Function, Implemented
free(), Function, Implemented
malloc(), Function, Implemented
realloc(), Function, Implemented
abort(), Function, Implemented
exit(), Function, Implemented
bsearch(), Function, Implemented
qsort(), Function, Implemented
```

NOTE: `getenv` is also included in another section. Section 4.6.1 [Environment Access], page 16.

ANSI C Section 4.11 — String Handling

```
strcpy(), Function, Implemented
strncpy(), Function, Implemented
strcat(), Function, Implemented
strncat(), Function, Implemented
strcmp(), Function, Implemented
strncmp(), Function, Implemented
strchr(), Function, Implemented
strcspn(), Function, Implemented
strpbrk(), Function, Implemented
strrchr(), Function, Implemented
strspn(), Function, Implemented
strstr(), Function, Implemented
strtok(), Function, Implemented
strlen(), Function, Implemented
```

ANSI C Section 4.12 — Date and Time Handling

```
asctime(), Function, Implemented
ctime(), Function, Implemented
gmtime(), Function, Implemented
localtime(), Function, Implemented
mktime(), Function, Implemented
strftime(), Function, Implemented
```

NOTE: RTEMS has no notion of time zones.

NOTE: `time` is also included in another section. Section 4.5.1 [Get System Time], page 16.

From Surrounding Text

```
EXIT_SUCCESS, Constant, Implemented
```

EXIT_FAILURE, Constant, Implemented

### 8.1.1 Extensions to Time Functions

### 8.1.2 Extensions to setlocale Function

LC_CTYPE, Constant, Implemented
LC_COLLATE, Constant, Implemented
LC_TIME, Constant, Implemented
LC_NUMERIC, Constant, Implemented
LC_MONETARY, Constant, Implemented
LC_ALL, Constant, Implemented

## 8.2 C Language Input/Output Functions

### 8.2.1 Map a Stream Pointer to a File Descriptor

fileno(), Function, Implemented
STDIN_FILENO, Constant, Implemented
STDOUT_FILENO, Constant, Implemented
STDERR_FILENO, Constant, Implemented

### 8.2.2 Open a Stream on a File Descriptor

fdopen(), Function, Implemented

### 8.2.3 Interactions of Other FILE-Type C Functions

### 8.2.4 Operations on Files - the remove Function

### 8.2.5 Temporary File Name - the tmpnam Function

### 8.2.6 Stdio Locking Functions

flockfile(), Function, Unimplemented
ftrylockfile(), Function, Unimplemented
funlockfile(), Function, Unimplemented

### 8.2.7 Stdio With Explicit Client Locking

getc_unlocked(), Function, Unimplemented
getchar_unlocked(), Function, Unimplemented
putc_unlocked(), Function, Unimplemented
putchar_unlocked(), Function, Unimplemented

## 8.3 Other C Language Functions

### 8.3.1 Nonlocal Jumps

sigjmp_buf, Type, Implemented
sigsetjmp(), Function, Implemented
siglongjmp(), Function, Implemented

### 8.3.2 Set Time Zone

`tzset(), Function, Unimplemented`

### 8.3.3 Find String Token

`strtok_r(), Function, Implemented`

### 8.3.4 ASCII Time Representation

`asctime_r(), Function, Implemented`

### 8.3.5 Current Time Representation

`ctime_r(), Function, Implemented`

### 8.3.6 Coordinated Universal Time

`gmtime_r(), Function, Implemented`

### 8.3.7 Local Time

`localtime_r(), Function, Implemented`

### 8.3.8 Pseudo-Random Sequence Generation Functions

`rand_r(), Function, Implemented`

# 9 System Databases

## 9.1 System Databases Section

## 9.2 Database Access

### 9.2.1 Group Database Access

```
struct group, Type, Implemented
getgrgid(), Function, Implemented
getgrgid_r(), Function, Implemented
getgrname(), Function, Implemented
getgrnam_r(), Function, Implemented
```

NOTE: Creates /etc/group if none exists.

### 9.2.2 User Database Access

```
struct passwd, Type, Implemented
getpwuid(), Function, Implemented
getpwuid_r(), Function, Implemented
getpwnam(), Function, Implemented
getpwnam_r(), Function, Implemented
```

NOTE: Creates /etc/passwd if none exists.

# 10 Data Interchange Format

## 10.1 Archive/Interchange File Format

### 10.1.1 Extended tar Format

```
tar format, Type, Unimplemented
TMAGIC, Constant, Unimplemented
TMAGLEN, Constant, Unimplemented
TVERSION, Constant, Unimplemented
TVERSLEN, Constant, Unimplemented
REGTYPE, Constant, Unimplemented
AREGTYPE, Constant, Unimplemented
LNKTYPE, Constant, Unimplemented
SYMTYPE, Constant, Unimplemented
CHRTYPE, Constant, Unimplemented
BLKTYPE, Constant, Unimplemented
DIRTYPE, Constant, Unimplemented
FIFOTYPE, Constant, Unimplemented
CONTTYPE, Constant, Unimplemented
TSUID, Constant, Unimplemented
TSGID, Constant, Unimplemented
TSVTX, Constant, Unimplemented
TUREAD, Constant, Unimplemented
TUWRITE, Constant, Unimplemented
TUEXEC, Constant, Unimplemented
TGREAD, Constant, Unimplemented
TGWRITE, Constant, Unimplemented
TGEXEC, Constant, Unimplemented
TOREAD, Constant, Unimplemented
TOWRITE, Constant, Unimplemented
TOEXEC, Constant, Unimplemented
```

NOTE: Requires <tar.h> which is not in newlib.

### 10.1.2 Extended cpio Format

```
cpio format, Type, Unimplemented
C_IRUSER, Constant, Unimplemented
C_IWUSER, Constant, Unimplemented
C_IXUSER, Constant, Unimplemented
C_IRGRP, Constant, Unimplemented
C_IWGRP, Constant, Unimplemented
C_IXGRP, Constant, Unimplemented
C_IROTH, Constant, Unimplemented
C_IWOTH, Constant, Unimplemented
C_IXOTH, Constant, Unimplemented
C_ISUID, Constant, Unimplemented
```

```
C_ISGID, Constant, Unimplemented
C_ISVTX, Constant, Unimplemented
```

NOTE: POSIX does not require a header file or structure. RedHat Linux 5.0 does not have a <cpio.h> although Solaris 2.6 does.

### 10.1.3 Multiple Volumes

# 11 Synchronization

## 11.1 Semaphore Characteristics

NOTE: Semaphores are implemented but only unnamed semaphores are currently tested.

`sem_t, Type, Implemented`

## 11.2 Semaphore Functions

### 11.2.1 Initialize an Unnamed Semaphore

`sem_init(), Function, Implemented`
`SEM_FAILED, Constant, Implemented`

### 11.2.2 Destroy an Unnamed Semaphore

`sem_destroy(), Function, Implemented`

### 11.2.3 Initialize/Open a Named Semaphore

`sem_open(), Function, Implemented`

### 11.2.4 Close a Named Semaphore

`sem_close(), Function, Implemented`

### 11.2.5 Remove a Named Semaphore

`sem_unlink(), Function, Implemented`

### 11.2.6 Lock a Semaphore

`sem_wait(), Function, Implemented`
`sem_trywait(), Function, Implemented`

### 11.2.7 Unlock a Semaphore

`sem_post(), Function, Implemented`

### 11.2.8 Get the Value of a Semaphore

`sem_getvalue(), Function, Implemented`

## 11.3 Mutexes

### 11.3.1 Mutex Initialization Attributes

`pthread_mutexattr_init(), Function, Implemented`
`pthread_mutexattr_destroy(), Function, Implemented`
`pthread_mutexattr_getpshared(), Function, Implemented`
`pthread_mutexattr_setpshared(), Function, Implemented`
`PTHREAD_PROCESS_SHARED, Constant, Implemented`
`PTHREAD_PROCESS_PRIVATE, Constant, Implemented`

### 11.3.2 Initializing and Destroying a Mutex

pthread_mutex_init(), Function, Implemented
pthread_mutex_destroy(), Function, Implemented
PTHREAD_MUTEX_INITIALIZER, Constant, Implemented

### 11.3.3 Locking and Unlocking a Mutex

pthread_mutex_lock(), Function, Implemented
pthread_mutex_trylock(), Function, Implemented
pthread_mutex_unlock(), Function, Implemented

## 11.4 Condition Variables

### 11.4.1 Condition Variable Initialization Attributes

pthread_condattr_init(), Function, Implemented
pthread_condattr_destroy(), Function, Implemented
pthread_condattr_getpshared(), Function, Implemented
pthread_condattr_setpshared(), Function, Implemented

### 11.4.2 Initialization and Destroying Condition Variables

pthread_cond_init(), Function, Implemented
pthread_cond_destroy(), Function, Implemented
PTHREAD_COND_INITIALIZER, Constant, Implemented

### 11.4.3 Broadcasting and Signaling a Condition

pthread_cond_signal(), Function, Implemented
pthread_cond_broadcast(), Function, Implemented

### 11.4.4 Waiting on a Condition

pthread_cond_wait(), Function, Implemented
pthread_cond_timedwait(), Function, Implemented

# 12 Memory Management

## 12.1 Memory Locking Functions

### 12.1.1 Lock/Unlock the Address Space of a Process

mlockall(), Function, Unimplemented
munlockall(), Function, Unimplemented
MCL_CURRENT, Constant, Unimplemented
MCL_FUTURE, Constant, Unimplemented

### 12.1.2 Lock/Unlock a Rand of Process Address Space

mlock(), Function, Unimplemented
munlock(), Function, Unimplemented

## 12.2 Memory Mapping Functions

### 12.2.1 Map Process Addresses to a Memory Object

mmap(), Function, Unimplemented
PROT_READ, Constant, Unimplemented
PROT_WRITE, Constant, Unimplemented
PROT_EXEC, Constant, Unimplemented
PROT_NONE, Constant, Unimplemented
MAP_SHARED, Constant, Unimplemented
MAP_PRIVATE, Constant, Unimplemented
MAP_FIXED, Constant, Unimplemented

### 12.2.2 Unmap Previously Mapped Addresses

munmap(), Function, Unimplemented

### 12.2.3 Change Memory Protection

mprotect(), Function, Unimplemented

### 12.2.4 Memory Object Synchronization

msync(), Function, Unimplemented, Unimplemented
MS_ASYNC, Constant, Unimplemented
MS_SYNC, Constant, Unimplemented
MS_INVALIDATE, Constant, Unimplemented

## 12.3 Shared Memory Functions

### 12.3.1 Open a Shared Memory Object

shm_open(), Function, Unimplemented

### 12.3.2 Remove a Shared Memory Object

shm_unlink(), Function, Unimplemented

# 13 Execution Scheduling

## 13.1 Scheduling Parameters

struct sched_param, Type, Implemented

## 13.2 Scheduling Policies

SCHED_FIFO, Constant, Implemented
SCHED_RR, Constant, Implemented
SCHED_OTHER, Constant, Implemented

NOTE: RTEMS adds SCHED_SPORADIC.

### 13.2.1 SCHED_FIFO

### 13.2.2 SCHED_RR

### 13.2.3 SCHED_OTHER

## 13.3 Process Scheduling Functions

### 13.3.1 Set Scheduling Parameters

sched_setparam(), Function, Dummy Implementation

### 13.3.2 Get Scheduling Parameters

sched_getparam(), Function, Dummy Implementation

### 13.3.3 Set Scheduling Policy and Scheduling Parameters

sched_setscheduler(), Function, Dummy Implementation

### 13.3.4 Get Scheduling Policy

sched_getscheduler(), Function, Dummy Implementation

### 13.3.5 Yield Processor

sched_yield(), Function, Implemented

### 13.3.6 Get Scheduling Parameter Limits

sched_get_priority_max(), Function, Implemented
sched_get_priority_min(), Function, Implemented
sched_get_priority_rr_get_interval(), Function, Implemented

## 13.4 Thread Scheduling

### 13.4.1 Thread Scheduling Attributes

PTHREAD_SCOPE_PROCESS, Constant, Implemented
PTHREAD_SCOPE_SYSTEM, Constant, Implemented

### 13.4.2 Scheduling Contention Scope

### 13.4.3 Scheduling Allocation Domain

### 13.4.4 Scheduling Documentation

## 13.5 Thread Scheduling Functions

### 13.5.1 Thread Creation Scheduling Attributes

pthread_attr_setscope(), Function, Implemented
pthread_attr_getscope(), Function, Implemented
pthread_attr_setinheritsched(), Function, Implemented
pthread_attr_getinheritsched(), Function, Implemented
pthread_attr_setschedpolicy(), Function, Implemented
pthread_attr_getschedpolicy(), Function, Implemented
pthread_attr_setschedparam(), Function, Implemented
pthread_attr_getschedparam(), Function, Implemented
PTHREAD_INHERIT_SCHED, Constant, Implemented
PTHREAD_EXPLICIT_SCHED, Constant, Implemented

### 13.5.2 Dynamic Thread Scheduling Parameters Access

pthread_setschedparam(), Function, Implemented
pthread_getschedparam(), Function, Implemented

## 13.6 Synchronization Scheduling

### 13.6.1 Mutex Initialization Scheduling Attributes

pthread_mutexattr_setprotocol(), Function, Implemented
pthread_mutexattr_getprotocol(), Function, Implemented
pthread_mutexattr_setprioceiling(), Function, Implemented
pthread_mutexattr_getprioceiling(), Function, Implemented
PTHREAD_PRIO_NONE, Constant, Implemented
PTHREAD_PRIO_INHERIT, Constant, Implemented
PTHREAD_PRIO_PROTECT, Constant, Implemented

### 13.6.2 Change the Priority Ceiling of a Mutex

pthread_mutex_setprioceiling(), Function, Implemented
pthread_mutex_getprioceiling(), Function, Implemented

# 14 Clocks and Timers

## 14.1 Data Definitions for Clocks and Timers

### 14.1.1 Time Value Specification Structures

struct timespec, Type, Implemented
struct itimerspec, Type, Implemented

### 14.1.2 Timer Event Notification Control Block

### 14.1.3 Type Definitions

clockid_t, Type, Implemented
timerid_t, Type, Implemented

### 14.1.4 Timer Event Notification Manifest Constants

CLOCK_REALTIME, Constant, Implemented
TIMER_ABSTIME, Constant, Implemented

## 14.2 Clock and Timer Functions

### 14.2.1 Clocks

clock_settime(), Function, Partial Implementation
clock_gettime(), Function, Partial Implementation
clock_getres(), Function, Implemented

### 14.2.2 Create a Per-Process Timer

timer_create(), Function, Implemented

### 14.2.3 Delete a Per-Process Timer

timer_delete(), Function, Implemented

### 14.2.4 Per-Process Timers

timer_settime(), Function, Implemented
timer_gettime(), Function, Implemented
timer_getoverrun(), Function, Implemented

### 14.2.5 High Resolution Sleep

nanosleep(), Function, Implemented

# 15 Message Passing

## 15.1 Data Definitions for Message Queues

### 15.1.1 Data Structures

NOTE: Semaphores are implemented but only unnamed semaphores are currently tested.

```
mqd_t, Type, Implemented
struct mq_attr, Type, Implemented
```

## 15.2 Message Passing Functions

### 15.2.1 Open a Message Queue

```
mq_open(), Function, Implemented
```

### 15.2.2 Close a Message Queue

```
mq_close(), Function, Implemented
```

### 15.2.3 Remove a Message Queue

```
mq_unlink(), Function, Implemented
```

### 15.2.4 Send a Message to a Message Queue

```
mq_send(), Function, Implemented
```

### 15.2.5 Receive a Message From a Message Queue

```
mq_receive(), Function, Implemented
```

### 15.2.6 Notify Process That a Message is Available on a Queue

```
mq_notify(), Function, Implemented
```

### 15.2.7 Set Message Queue Attributes

```
mq_setattr(), Function, Implemented
```

### 15.2.8 Get Message Queue Attributes

```
mq_getattr(), Function, Implemented
```

# 16 Thread Management

## 16.1 Threads

## 16.2 Thread Functions

### 16.2.1 Thread Creation Attributes

pthread_attr_init(), Function, Implemented
pthread_attr_destroy(), Function, Implemented
pthread_attr_setstacksize(), Function, Implemented
pthread_attr_getstacksize(), Function, Implemented
pthread_attr_setstackaddr(), Function, Implemented
pthread_attr_getstackaddr(), Function, Implemented
pthread_attr_setdetachstate(), Function, Implemented
pthread_attr_getdetachstate(), Function, Implemented
PTHREAD_CREATE_JOINABLE, Constant, Implemented
PTHREAD_CREATE_DETACHED, Constant, Implemented

### 16.2.2 Thread Creation

pthread_create(), Function, Implemented

### 16.2.3 Wait for Thread Termination

pthread_join(), Function, Implemented

### 16.2.4 Detaching a Thread

pthread_detach(), Function, Implemented

### 16.2.5 Thread Termination

pthread_exit(), Function, Implemented

### 16.2.6 Get Thread ID

pthread_self(), Function, Implemented

### 16.2.7 Compare Thread IDs

pthread_equal(), Function, Implemented

### 16.2.8 Dynamic Package Initialization

pthread_once(), Function, Implemented
PTHREAD_ONCE_INIT, Constant, Implemented

# 17 Thread-Specific Data

## 17.1 Thread-Specific Data Functions

### 17.1.1 Thread-Specific Data Key Creation

`pthread_key_create(), Function, Implemented`

### 17.1.2 Thread-Specific Data Management

`pthread_key_setspecific(), Function, Implemented`
`pthread_key_getspecific(), Function, Implemented`

### 17.1.3 Thread-Specific Data Key Deletion

`pthread_key_delete(), Function, Implemented`

# 18 Thread Cancellation

## 18.1 Thread Cancellation Overview

### 18.1.1 Cancelability States

PTHREAD_CANCEL_DISABLE, Constant, Implemented
PTHREAD_CANCEL_ENABLE, Constant, Implemented
PTHREAD_CANCEL_ASYNCHRONOUS, Constant, Implemented
PTHREAD_CANCEL_DEFERRED, Constant, Implemented

### 18.1.2 Cancellation Points

### 18.1.3 Thread Cancellation Cleanup Handlers

PTHREAD_CANCELED, Constant, Unimplemented

### 18.1.4 Async-Cancel Safety

## 18.2 Thread Cancellation Functions

### 18.2.1 Canceling Execution of a Thread

pthread_cancel(), Function, Implemented

### 18.2.2 Setting Cancelability State

pthread_setcancelstate(), Function, Implemented
pthread_setcanceltype(), Function, Implemented
pthread_testcancel(), Function, Implemented

### 18.2.3 Establishing Cancellation Handlers

pthread_cleanup_push(), Function, Implemented
pthread_cleanup_pop(), Function, Implemented

## 18.3 Language-Independent Cancellation Functionality

### 18.3.1 Requesting Cancellation

### 18.3.2 Associating Cleanup Code With Scopes

### 18.3.3 Controlling Cancellation Within Scopes

### 18.3.4 Defined Cancellation Sequence

### 18.3.5 List of Cancellation Points

# 19 Compliance Summary

## 19.1 General Chapter

```
Functions:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 21
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

**FEATURE FLAG COUNTS DO NOT ADD UP!!**

```
Constants:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.2 Terminology and General Requirements Chapter

```
Functions:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 19
    Implemented     : 19
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 32
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

**FEATURE FLAG COUNTS DO NOT ADD UP!!**

```
Constants:
    Total Number    : 126
    Implemented     : 124
    Unimplemented   : 2
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.3 Process Primitives Chapter

```
Functions:
    Total Number    : 36
    Implemented     : 20
    Unimplemented   : 0
    Unimplementable : 16
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 5
    Implemented     : 5
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 40
    Implemented     : 32
    Unimplemented   : 6
    Unimplementable : 2
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.4 Process Environment Chapter

```
Functions:
    Total Number    : 23
    Implemented     : 21
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 2
    Untested        : 0

Data Types:
    Total Number    : 2
    Implemented     : 2
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 53
    Implemented     : 51
    Unimplemented   : 2
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.5 Files and Directories Chapter

```
Functions:
    Total Number    : 35
    Implemented     : 30
    Unimplemented   : 3
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 1
```

**FUNCTION COUNTS DO NOT ADD UP!!**

```
Data Types:
    Total Number    : 3
    Implemented     : 3
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 39
    Implemented     : 37
    Unimplemented   : 2
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.6 Input and Output Primitives Chapter

```
Functions:
    Total Number    : 19
    Implemented     : 9
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 9
    Untested        : 0
```

**FUNCTION COUNTS DO NOT ADD UP!!**

```
Data Types:
    Total Number    : 2
    Implemented     : 1
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 1

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 24
    Implemented     : 24
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.7 Device- and Class-Specific Functions Chapter

```
Functions:
    Total Number    : 12
    Implemented     : 8
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 4
    Untested        : 0

Data Types:
    Total Number    : 3
    Implemented     : 3
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 77
    Implemented     : 76
    Unimplemented   : 1
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.8 Language-Specific Services for the C Programming Language Chapter

```
Functions:
    Total Number    : 125
    Implemented     : 117
    Unimplemented   : 8
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 2
    Implemented     : 2
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 11
    Implemented     : 11
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.9 System Databases Chapter

```
Functions:
    Total Number    : 8
    Implemented     : 8
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 2
    Implemented     : 2
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.10 Data Interchange Format Chapter

```
Functions:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 2
    Implemented     : 0
    Unimplemented   : 2
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 37
    Implemented     : 0
    Unimplemented   : 37
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.11 Synchronization Chapter

```
Functions:
    Total Number    : 28
    Implemented     : 28
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 1
    Implemented     : 1
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 5
    Implemented     : 5
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.12 Memory Management Chapter

```
Functions:
    Total Number    : 10
    Implemented     : 0
    Unimplemented   : 10
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 12
    Implemented     : 0
    Unimplemented   : 12
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.13 Execution Scheduling Chapter

```
Functions:
    Total Number    : 24
    Implemented     : 20
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 4
    Untested        : 0

Data Types:
    Total Number    : 1
    Implemented     : 1
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 10
    Implemented     : 10
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.14 Clocks and Timers Chapter

```
Functions:
    Total Number    : 9
    Implemented     : 7
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 2
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 4
    Implemented     : 4
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 2
    Implemented     : 2
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.15 Message Passing Chapter

```
Functions:
    Total Number    : 8
    Implemented     : 8
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 2
    Implemented     : 2
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.16 Thread Management Chapter

```
Functions:
    Total Number    : 15
    Implemented     : 15
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 3
    Implemented     : 3
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.17 Thread-Specific Data Chapter

```
Functions:
    Total Number    : 4
    Implemented     : 4
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.18 Thread Cancellation Chapter

```
Functions:
    Total Number    : 6
    Implemented     : 6
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Data Types:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Feature Flags:
    Total Number    : 0
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0

Constants:
    Total Number    : 5
    Implemented     : 4
    Unimplemented   : 1
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

## 19.19 Overall Summary

```
Functions:
    Total Number    : 362
    Implemented     : 301
    Unimplemented   : 21
    Unimplementable : 16
    Partial         : 2
    Dummy           : 19
    Untested        : 1
```

**FUNCTION COUNTS DO NOT ADD UP!!**

```
Data Types:
    Total Number    : 48
    Implemented     : 45
    Unimplemented   : 2
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 1

Feature Flags:
    Total Number    : 53
    Implemented     : 0
    Unimplemented   : 0
    Unimplementable : 0
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

**FEATURE FLAG COUNTS DO NOT ADD UP!!**

```
Constants:
    Total Number    : 444
    Implemented     : 379
    Unimplemented   : 63
    Unimplementable : 2
    Partial         : 0
    Dummy           : 0
    Untested        : 0
```

# Command and Variable Index

There are currently no Command and Variable Index entries.

# Concept Index

There are currently no Concept Index entries.