**INTERNATIONAL STANDARD ISO/IEC 9945-2:2003**
TECHNICAL CORRIGENDUM 1

Published 2004-09-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

# Information technology — Portable Operating System Interface (POSIX®) —

## Part 2: System Interfaces

TECHNICAL CORRIGENDUM 1

*Technologies de l'information — Interface pour la portabilité des systèmes (POSIX®) —*

*Partie 2: Interfaces systèmes*

*RECTIFICATIF TECHNIQUE 1*

Technical Corrigendum 1 to ISO/IEC 9945-2:2003 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces.*



**ICS 35.060** **Ref. No. ISO/IEC 9945-2:2003/Cor.1:2004(E)**



Published in Switzerland

# 1 Scope

This technical corrigendum addresses issues raised in defect reports and interpretation requests submitted up to 14 August 2003, and that meet all of the following criteria:
a. They are in the scope of the approved International Standard.
b. They contain no new APIs (functions/utilities), however, they may add enumeration symbols, non-function # defines, and reserve additional namespaces.
c. They address contradictions between different parts of the International Standard, or add consistency between it and overriding International Standards, or address security-related problems.

# 2 Changes to ISO/IEC 9945-2

**Change Number: XSH/TC1/1 [XBD ERN 20]**
On Page: xxx Line: "ISO/IEC 8859" Section: Referenced Documents
Add after line starting "Part 10":
"Part 11: Latin/Thai Alphabet"
Add after line starting "Part 15":
"Part 16: Latin Alphabet No. 10"

**Change Number: XSH/TC1/2 [XSH ERN 167,TC2d5 ERN 9]**
On Page: 15,19 Line: 580-583,713-714 Section: 2.2.2
On lines 580-583
Change From:
"Implementations may add symbols to the headers shown in the following table, provided the identifiers for those symbols begin with the corresponding reserved prefixes in the following table, and do not use the reserved prefixes posix_, POSIX_, or _POSIX_."
To:
"Implementations may add symbols to the headers shown in the following table, provided the identifiers for those symbols either:
1. begin with one of the corresponding reserved prefixes in the table; or
2. have one of the corresponding complete names in the table; or
3. end in the string indicated as a reserved suffix in the table and do not use the reserved prefixes posix_, POSIX_, or _POSIX_, as long as the reserved suffix is in that part of the name considered significant by the implementation.
Symbols that use the reserved prefix _POSIX_ may be made visible by implementations in any header defined by IEEE Std 1003.1-2001."
On lines 713-714
Change From:
"1. All identifiers that begin with an underscore and either an uppercase letter or another underscore are always reserved for any use by the implementation."
To:
"1. With the exception of identifiers beginning with the prefix _POSIX_, all identifiers that begin with an underscore and either an uppercase letter or another underscore are always reserved for any use by the implementation."
Rationale:
This change permits implementations to have symbols with the prefix _POSIX_ visible in any header. A change in the 2003 edition had disallowed this.

**Change Number: XSH/TC1/3 [XSH ERN 1]**
On Page: 18 Line: 668 Section: 2.2.2
Add the following line into the table after line 668
"<math.h> FP_[A-Z]"
Rationale:
This text is added for consistency with the math.h page in the Base Definitions volume which states "Additional implementation-defined floating-point classifications, with macro definitions beginning with FP_ and an uppercase letter, may also be specified by the implementation."

**Change Number: XSH/TC1/4 [XSH ERN 99]**
On Page: 33 Line: 1337 Section: 2.4.3
Add "sockatmark()" to the table in 2.4.3 (the list of functions that shall be either reentrant or non-interruptible by signals and shall be async-signal-safe).
Rationale:
This was an omission when the sockatmark() function was added.

**Change Number: XSH/TC1/5 [XSH ERN 109]**
On Page: 55 Line: 2267 Section: 2.9.5.2
Add "fdatasync()" into the table of functions after "fcntl()".
Rationale:
This was an omission. Since fdatasync() is often implemented in terms of fsync() it should also be in this list.

**Change Number: XSH/TC1/6 [XSH ERN 76]**
On Page: 56 Line: 2277 Section: 2.9.5.2 Cancellation Points
Insert into the list of functions that "may have" a cancellation point (the second list of functions in this section) in the appropriate order:

"access

asctime

asctime_r

ctime

ctime_r

fmtmsg

fpathconf

fstat

getaddrinfo

gethostid

getnameinfo

getopt [with footnote if opterr is nonzero]

link

localtime

localtime_r

lstat

mktime

pathconf

posix_openpt

stat

strerror_r

strftime

symlink

sync

tzset

wcsftime

wordexp"

**Change Number: XSH/TC1/7 [XSH ERN 77]**
On Page: 57 Line: 2336-2353 Section: 2.9.5.3
Change From:
"Each thread maintains a list of cancellation cleanup handlers. The programmer uses the pthread_cleanup_push() and pthread_cleanup_pop() functions to place routines on and remove routines from this list. When a cancellation request is acted upon, the routines in the list are invoked one by one in LIFO sequence; that is, the last routine pushed onto the list (Last In) is the first to be invoked (First Out). The thread invokes the cancellation cleanup handler with cancellation disabled until the last cancellation cleanup handler returns. When the cancellation cleanup handler for a scope is invoked, the storage for that scope remains valid. If the last cancellation cleanup handler returns, thread execution is terminated and a status of PTHREAD_CANCELED is made available to any threads joining with the target. The symbolic constant PTHREAD_CANCELED expands to a constant expression of type (void *) whose value matches no pointer to an object in memory nor the value NULL. The cancellation cleanup handlers are also invoked when the thread calls pthread_exit().
A side effect of acting upon a cancellation request while in a condition variable wait is that the mutex is re-acquired before calling the first cancellation cleanup handler. In addition, the thread is no longer considered to be waiting for the condition and the thread shall not have consumed any pending condition signals on the condition.
A cancellation cleanup handler cannot exit via longjmp() or siglongjmp()."
To:
Each thread maintains a list of cancellation cleanup handlers. The programmer uses the pthread_cleanup_push() and pthread_cleanup_pop() functions to place routines on and remove routines from this list. When a cancellation request is acted upon, or when a thread calls pthread_exit(), the the thread first disables cancellation by setting its cancelability state to PTHREAD_CANCEL_DISABLE and its cancelability type to PTHREAD_CANCEL_DEFERRED. The cancelability state shall remain set to PTHREAD_CANCEL_DISABLE until the thread has terminated. The behavior is undefined if a cancellation cleanup handler or thread-specific data destructor routine changes the cancelability state to PTHREAD_CANCEL_ENABLE. The routines in the thread's list of cancellation cleanup handlers are invoked one by one in LIFO sequence; that is, the last routine pushed onto the list (Last In) is the first to be invoked (First Out). When the cancellation cleanup handler for a scope is invoked, the storage for that scope remains valid. If the last cancellation cleanup handler returns, thread-specific data destructors (if any) associated with thread-specific data keys for which the thread has non-NULL values will be run, in unspecified order, as described for pthread_key_create().
After all cancellation cleanup handlers and thread-specific data destructors have returned, thread execution is terminated. If the thread has terminated because of a call to pthread_exit(), the value_ptr argument is made available to any threads joining with the target. If the thread has terminated by acting on a cancellation request, a status of PTHREAD_CANCELED is made available to any threads joining with the target. The symbolic constant PTHREAD_CANCELED expands to a constant expression of type (void *) whose value matches no pointer to an object in memory nor the value NULL.
A side effect of acting upon a cancellation request while in a condition variable wait is that the mutex is re-acquired before calling the first cancellation cleanup handler. In addition, the thread is no longer considered to be waiting for the condition and the thread shall not have consumed any pending condition signals on the condition.
A cancellation cleanup handler cannot exit via longjmp() or siglongjmp()."

**Change Number: XSH/TC1/8 [XSH ERN 89]**
On Page: 58 Line: 2379 Section: Threads
Insert new section:
"2.9.8 Use of Application-Managed Thread Stacks
An "application-managed thread stack" is a region of memory allocated by the application, for example, memory returned by the malloc() or mmap() functions, and designated as a stack through the act of passing an address related to that memory as the "stackaddr" argument to pthread_attr_setstackaddr() (obsolete) or by passing the address and size of the stack, respectively, as the stackaddr and stacksize arguments to pthread_attr_setstack(). Application-managed stacks allow the application to precisely control the placement and size of a stack. The application grants to the implementation permanent ownership of and control over the application-managed stack when the attributes object in which the stack or stackaddr attribute has been set is used, either by presenting that attributes object as the 'attr' argument in a call to pthread_create() that completes successfully, or by storing a pointer to the attributes object in the sigev_notify_attributes member of a 'struct sigevent' and passing that 'struct sigevent' to a function accepting such argument that completes successfully. The application may thereafter utilize the memory within the stack only within the normal context



 

of stack usage within or properly synchronized with a thread that has been scheduled by the implementation with stack pointer value(s) that are within the range of that stack. In particular, the region of memory cannot be freed, nor can it be later specified as the stack for another thread.
When specifying an attributes object with an application-managed stack through the sigev_notify_attributes member of a 'struct sigevent', the results are undefined if the requested signal is generated multiple times (as for a repeating timer).
Until an attributes object in which the stack or stackaddr attribute has been set is used, the application retains ownership of and control over the memory allocated to the stack. It may free or reuse the memory as long as it either deletes the attributes object, or before using the attributes object replaces the stack by making an additional call to the same function, either pthread_attr_setstackaddr() or pthread_attr_setstack(), that was used originally to designate the stack. There is no mechanism to retract the reference to an application-managed stack by an existing attributes object.
Once an attributes object with an application-managed stack has been used, that attributes object cannot be used again by a subsequent call to pthread_create() or any function accepting a 'struct sigevent' with sigev_notify_attributes containing a pointer to the attributes object, without designating an unused application-managed stack by making an additional call to the function originally used to define the stack, pthread_attr_setstack() or pthread_attr_setstackaddr()."

**Change Number: XSH/TC1/9 [XSH ERN 138]**
On Page: 92 Line: 3558-3559 Section: abort
In the APPLICATION USAGE section
Change From:
"Catching the signal is intended to provide the application writer with a portable means to abort processing, free from possible interference from any implementation-defined functions."
To:
"Catching the signal is intended to provide the application writer with a portable means to abort processing, free from possible interference from any implementation-supplied functions."

**Change Number: XSH/TC1/10 [XSH ERN 2]**
On Page: 105 Line: 3919 Section: aio_cancel
In the RETURN VALUE section
Change From:
"The aio_cancel() function shall return the value AIO_CANCELED to the calling process if the requested operation(s) were canceled."
To:
"The aio_cancel() function shall return the value AIO_CANCELED if the requested operation(s) were canceled."
Rationale:
The use of the term "to the calling process" was unnecessary and incorrect.

**Change Number: XSH/TC1/11 [XSH ERN 3]**
On Page: 108 Line: 4022 Section: aio_fsync
In the RETURN VALUE section
Change From:
"The aio_fsync() function shall return the value 0 to the calling process if the I/O operation is successfully queued;"
To:
"The aio_fsync() function shall return the value 0 if the I/O operation is successfully queued;"
Rationale:
The use of the term "to the calling process" was unnecessary and incorrect.

**Change Number: XSH/TC1/12 [XSH ERN 4,5]**
On Page: 110 Line: 4065,4088 Section: aio_read
In the DESCRIPTION section
Change From:
"If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted at a priority equal to the scheduling priority of the process minus aiocbp->aio_reqprio."

To:
"[PIO] If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted at a priority equal to a base scheduling priority minus aiocbp->reqprio.
If Thread Priority Scheduling is not supported then the base scheduling priority is that of the calling process, [PIO TPS] otherwise the base scheduling priority is that of the calling thread. [/end shading]"
Rationale:
The previous wording did not take threads into account.
In the RETURN VALUE section
Change From:
"The aio_read() function shall return the value zero to the calling process if the I/O operation is successfully queued;"
To:
"The aio_read() function shall return the value zero if the I/O operation is successfully queued;"
Rationale:
The use of the term "to the calling process" was unnecessary and incorrect.

**Change Number: XSH/TC1/13 [XSH ERN 83]**
On Page: 111 Line: 4101-4103 Section: aio_read
In the ERRORS section
Change From:
"[EINVAL] The file offset value implied by aiocbp->aio_offset would be invalid, aiocbp->aio_reqprio is not a valid value, or aiocbp->aio_nbytes is an invalid value."
To:
"[EINVAL] The file offset value implied by aiocbp->aio_offset would be invalid, [PIO]aiocbp->aio_reqprio is not a valid value[/PIO], or aiocbp->aio_nbytes is an invalid value."
Rationale:
Detection of an [EINVAL] error for a invalid value of aiocbp->aio_reqprio should only be required if _POSIX_PRIORITIZED_IO is supported.

**Change Number: XSH/TC1/14 [XSH ERN 6,7]**
On Page: 116 Line: 4252,4276 Section: aio_write
In the DESCRIPTION section
Change From:
"If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted at a priority equal to the scheduling priority of the process minus aiocbp->aio_reqprio."
To:
"[PIO] If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted at a priority equal to a base scheduling priority minus aiocbp->reqprio. If Thread Priority Scheduling is not supported then the base scheduling priority is that of the calling process, [PIO TPS] otherwise the base scheduling priority is that of the calling thread. [/end shading]"
Rationale:
The previous wording did not take threads into account.
In the RETURN VALUE section
Change From:
"The aio_write () function shall return the value zero to the calling process if the I/O operation is successfully queued;"
To:
"The aio_write () function shall return the value zero if the I/O operation is successfully queued;"
Rationale:
The use of the term "to the calling process" was unnecessary and incorrect.

**Change Number: XSH/TC1/15 [XSH ERN 83]**
On Page: 117 Line: 4290-4292 Section: aio_write
In the ERRORS section
Change From:
"[EINVAL] The file offset value implied by aiocbp->aio_offset would be invalid, aiocbp->aio_reqprio is not a valid value, or aiocbp->aio_nbytes is an invalid value."

To:
"[EINVAL] The file offset value implied by aiocbp->aio_offset would be invalid, [PIO]aiocbp->aio_reqprio is not a valid value[/PIO], or aiocbp->aio_nbytes is an invalid value."
Rationale:
Detection of an [EINVAL] error for a invalid value of aiocbp->aio_reqprio should only be required if _POSIX_PRIORITIZED_IO is supported.

**Change Number: XSH/TC1/16 [XSH ERN 139]**
On Page: 119-120 Line: 4375-4376 Section: alarm
In the RATIONALE section
Change From:
"In some implementations, including 4.3 BSD, very large values of the seconds argument are silently rounded down to an implementation-defined maximum value"
To:
"In some implementations, including 4.3 BSD, very large values of the seconds argument are silently rounded down to an implementation-specific maximum value"

**Change Number: XSH/TC1/17 [XSH ERN 115]**
On Page: 122 Line: 4440 Section: asctime
In the RETURN VALUE section
Change From:
"Upon successful completion, asctime() shall return a pointer to the string."
To:
"Upon successful completion, asctime() shall return a pointer to the string. [CX] If the function is unsuccessful, it shall return NULL. [/CX]"

**Change Number: XSH/TC1/18 [XSH ERN 123]**
On Page: 132 Line: 4738 Section: atan2
In the EXAMPLES section
Change From:
"None."
To:
"Converting Cartesian to Polar coordinates system
The function below uses atan2() to convert a 2d vector expressed in cartesian coordinates (x,y) to the polar coordinates (rho,theta). There are other ways to compute the angle theta, using asin(), acos() or atan(). However, atan2() presents here two advantages:
- the angle's quadrant is automatically determined.
- the singular cases (0,y) are taken into account.
Finally, this example uses hypot() rather than sqrt() since it is better for special cases, see hypot() for more information.

```
#include <math.h>

void

cartesian_to_polar(const double x, const double y,

                   double *rho, double *theta

                   )

{

  *rho   = hypot (x,y); /* better than sqrt(x*x+y*y) */

  *theta = atan2 (y,x);

}"
```

**Change Number: XSH/TC1/19 [XSH ERN 8]**
On Page: 137 Line: 4876 Section: atexit
Add to end of the APPLICATION USAGE section:
"Since the behavior is undefined if the exit() function is called more than once, portable applications calling atexit() must ensure that the exit() function is not called at normal process termination when all functions registered by the atexit() function are called. All functions registered by the atexit() function are called at normal process termination, which occurs by a call to the exit() function or a return from main() or on the last thread termination, when the behavior is as if the implementation called exit() with a zero argument at thread termination time. If, at normal process termination, a function registered by the atexit() function is called and a portable application needs to stop further exit() processing, it must call the _exit() function or the _Exit() function or one of the functions which cause abnormal process termination."
Rationale:
Additional clarifications for application usage are added.

**Change Number: XSH/TC1/20 [XSH ERN 140]**
On Page: 142 Line: 5007-5009 Section: basename
In the DESCRIPTION section
Change From:
"If the string consists entirely of the '/' character, basename() shall return a pointer to the string "/". If the string is exactly "//", it is implementation-defined whether '/' or "//" is returned."
To:
"If the string pointed to by path consists entirely of the '/' character, basename() shall return a pointer to the string "/". If the string pointed to by path is exactly "//", it is implementation-defined whether "/" or "//" is returned."

**Change Number: XSH/TC1/21 [XSH ERN 138]**
On Page: 259 Line: 8533-8534 Section: dlopen
In the DESCRIPTION section
Change From:
"If neither RTLD_GLOBAL nor RTLD_LOCAL are specified, then an implementation-defined default behavior shall be applied."
To:
"If neither RTLD_GLOBAL nor RTLD_LOCAL are specified, then the default behavior is unspecified."

**Change Number: XSH/TC1/22 [XSH ERN 120]**
On Page: 289 Line: 9418 Section: erf
In the EXAMPLES section
Change From:
"None."
To:
"Computing the probability for a normal variate
This example shows how to use erf() to compute the probability that a normal variate assumes a value in the range [x1,x2] with x1<=x2. This example uses the constant M_SQRT1_2 which is an XSI extension.

```
#include <math.h>

double

Phi(const double x1, const double x2)

{

  return ( erf(x2*M_SQRT1_2) - erf(x1*M_SQRT1_2) ) / 2;

}"
```

**Change Number: XSH/TC1/23 [XSH ERN 132]**
On Page: 294 Line: 9526 Section: errno
In the DESCRIPTION section
Add at the end of the second paragraph:
"The setting of errno after a successful call to a function is unspecified unless the description of that function specifies that errno shall not be modified".

**Change Number: XSH/TC1/24 [XSH ERN 45,46,63]**
On Page: 297-299 Line: 9637 Section: exec
In the DESCRIPTION section
On line 9637
Change From:
"The state of the floating-point environment in the new process image shall be set to the default."
To:
"The state of the floating-point environment in the new process image [THR]or in the initial thread of the new process image[/THR] shall be set to the default."
On line 9650
Change From:
"After a successful call to any of the exec functions, any functions previously registered by atexit() are no longer registered."
To:
"After a successful call to any of the exec functions, any functions previously registered by atexit() [THR]or pthread_atfork()[/THR] are no longer registered."
On lines 9675-9677
Change From:
"[PS]For the SCHED_FIFO and SCHED_RR scheduling policies, the policy and priority settings shall not be changed by a call to an exec function. For other scheduling policies, the policy and priority settings on exec are implementation-defined.[/PS]"
To:
"When the calling process image does not use the SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC scheduling policies, the scheduling policy and parameters of the new process image and the initial thread in that new process image are implementation defined. [PS]When the calling process image uses the SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC scheduling policies, the process policy and scheduling parameter settings shall not be changed by a call to an exec function.[/PS] [TPS]The initial thread in the new process image shall inherit the process scheduling policy and parameters. It shall have the default system contention scope, but shall inherit its allocation domain from the calling process image.[/TPS]"
Insert the following after line 9701
(before the list that begins "The new process shall inherit at least..."):
"[THR]The thread id of the initial thread in the new process image is unspecified. The size and location of the stack on which the initial thread in the new process image runs is unspecified. The initial thread in the new process image shall have its cancellation type set to PTHREAD_CANCEL_DEFERRED and its cancellation state set to PTHREAD_CANCEL_ENABLED. The initial thread in the new process image shall have all thread-specific data values set to NULL and all thread-specific data keys shall be removed by the call to exec without running destructors. The initial thread in the new process image shall be joinable, as if created with the detachstate attribute set to PTHREAD_CREATE_JOINABLE.[/THR]"
Add after line 9722
"The initial thread of the new process shall inherit at least the following attributes from the calling thread:
* Signal mask (see sigprocmask() and pthread_sigmask())
* Pending signals (see sigpending())"
On lines 9723-9725
Change From:
"All other process attributes defined in this volume of IEEE Std 1003.1-2001 shall be the same in the new and old process images. The inheritance of process attributes not defined by this volume of IEEE Std 1003.1-2001 is implementation-defined."
To:
"All other process attributes defined in this volume of IEEE Std 1003.1-2001 shall be inherited in the new process image from the old process image. All other thread attributes defined in this volume of IEEE Std 1003.1-2001 shall be inherited in the initial thread in the new process image from the calling thread in the old process image. The inheritance of process or thread attributes not defined by this volume of IEEE Std 1003.1-2001 is implementation-defined."

On lines 9726-9728
Change From:
"A call to any exec function from a process with more than one thread shall result in all threads being terminated and the new executable image being loaded and executed. No destructor functions shall be called."
To:
"[THR]A call to any exec function from a process with more than one thread shall result in all threads being terminated and the new executable image being loaded and executed. No destructor functions or cleanup handlers shall be called.[/THR]"

**Change Number: XSH/TC1/25 [XSH ERN 9]**
On Page: 303 Line: 9890 Section: exec
In the RATIONALE section
Change From:
"SIG_IGN, and that the process signal mask be unchanged across an exec."
To:
"SIG_IGN, and that the new process image inherits the signal mask of the thread that called exec in the old process image"

**Change Number: XSH/TC1/26 [XSH ERN 119]**
On Page: 313 Line: 10264 Section: exp
In the EXAMPLES section
Change From:
"None."
To:
"Computing the density of the standard normal distribution
This function shows an implementation for the density of the standard normal distribution using exp(). This example uses the constant M_PI which is an XSI extension.

```
#include <math.h>

double

normal_density (double  x

                )

{

  return exp(-x*x/2) / sqrt (2*M_PI);

}"
```

**Change Number: XSH/TC1/27 [XSH ERN 121]**
On Page: 318 Line: 10437 Section: fabs
In the EXAMPLES section
Change From:
"None."
To:
" Computing the 1-norm of a floating point vector
This example shows the use of fabs() to compute the 1-norm of a vector defined as follows:
norm1(v) = |v[0]| + |v[1]| + ... + |v[n-1]|
where |x| denotes the absolute value of x, n denotes the vector's dimension, and v[i] denotes the i-th component of v (0<=i<n)

```
 #include <math.h>

 double

 norm1(const double v[], const int n )
```

```
{
  int     i;
  double  n1_v;  /* 1-norm of v */
  n1_v = 0;
  for (i=0; i<n; i++) {
    n1_v += fabs (v[i]);
  }
  return n1_v;
}"
```

**Change Number: XSH/TC1/28 [XSH ERN 11]**
On Page: 327 Line: 10726 Section: fclose
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the process would be delayed in the write operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the thread would be delayed in the write operation. [/CX]"
Rationale: It is the thread that is delayed not the process.

**Change Number: XSH/TC1/29 [XSH ERN 134]**
On Page: 332 Line: 10935 Section: fcntl
In the EXAMPLES section
Change From:
"None."
To:
"Locking and unlocking a file
The following example demonstrates how to place a lock on bytes 100 to 109 of a file and then later remove it. F_SETLK is used to perform a non-blocking lock request so that the process does not have to wait if an incompatible lock is held by another process; instead the process can take some other action.

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
int
main(int argc, char *argv[])
{
    int fd;
    struct flock fl;
    fd = open("testfile", O_RDWR);
```

```
    if (fd == -1)
        /* Handle error */;
    /* Make a non-blocking request to place a write lock
       on bytes 100-109 of testfile */
    fl.l_type = F_WRLCK;
    fl.l_whence = SEEK_SET;
    fl.l_start = 100;
    fl.l_len = 10;
    if (fcntl(fd, F_SETLK, &fl) == -1) {
        if (errno == EACCES || errno == EAGAIN) {
            printf("Already locked by another process\n");
            /* We can't get the lock at the moment */
        } else {
            /* Handle unexpected error */;
        }
    } else { /* Lock was granted... */
        /* Perform I/O on bytes 100 to 109 of file */
        /* Unlock the locked bytes */
        fl.l_type = F_UNLCK;
        fl.l_whence = SEEK_SET;
        fl.l_start = 100;
        fl.l_len = 10;
        if (fcntl(fd, F_SETLK, &fl) == -1)
            /* Handle error */;
    }
    exit(EXIT_SUCCESS);
} /* main */
```

Setting the close-on-exec flag
The following example demonstrates how to set the close on exec flag for the file descriptor fd.

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

```
#include <fcntl.h>

...

    int flags;

    flags = fcntl(fd, F_GETFD);

    if (flags == -1)

        /* Handle error */;

    flags |= FD_CLOEXEC;

    if (fcntl(fd, F_SETFD, flags) == -1)

        /* Handle error */;"
```

**Change Number: XSH/TC1/30 [XSH ERN 141]**
On Page: 343 Line: 11245-11247 Section: fdopen
In the RATIONALE section
Change From:
"The file descriptor may have been obtained from open(), creat(), pipe(), dup(), or fcntl(); inherited through fork() or exec; or perhaps obtained by implementation-defined means, such as the 4.3 BSD socket() call."
To:
The file descriptor may have been obtained from open(), creat(), pipe(), dup(), fcntl() or socket(); inherited through fork(), posix_spawn() or exec; or perhaps obtained by other means."

**Change Number: XSH/TC1/31 [XSH ERN 12]**
On Page: 360 Line: 11686 Section: fflush
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the process would be delayed in the write operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the thread would be delayed in the write operation. [/CX]"

**Change Number: XSH/TC1/32 [XSH ERN 13]**
On Page: 364 Line: 11802 Section: fgetc
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the process would be delayed in the fgetc() operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the thread would be delayed in the fgetc() operation. [/CX]"

**Change Number: XSH/TC1/33 [XSH ERN 14]**
On Page: 370 Line: 11969 Section: fgetwc
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the process would be delayed in the fgetwc() operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the thread would be delayed in the fgetwc() operation. [/CX]"

**Change Number: XSH/TC1/34 [XBD ERN 11]**
On Page: 399 Line: 12942 Section: fpathconf
In the DESCRIPTION section
Insert after the "{PIPE_BUF}" entry:
"{POSIX2_SYMLINKS} _PC_2_SYMLINKS 4"

**Change Number: XSH/TC1/35 [XSH ERN 15]**
On Page: 399-402 Line: 12943-12947,12975,13054 Section: fpathconf
In the DESCRIPTION section
Insert the number "10" into the third column for the rows beginning {POSIX_ALLOC_SIZE_MIN}, {POSIX_REC_INCR_XFER_SIZE}, {POSIX_REC_MAX_XFER_SIZE}, {POSIX_REC_MIN_XFER_SIZE}, and {POSIX_REC_XFER_ALIGN}.
Add a new bullet item after Item 9 after line 12975
"10. If path or fildes does not refer to a regular file, it is unspecified whether an implementation supports an association of the variable name with the specified file. If an implementation supports such an association for other than a regular file, the value returned is unspecified."
Add to the RATIONALE section as a new paragraph, after line 13054
"It was the intention of 1003.1d that the variables {POSIX_ALLOC_SIZE_MIN}, {POSIX_REC_INCR_XFER_SIZE}, {POSIX_REC_MAX_XFER_SIZE}, {POSIX_REC_MIN_XFER_SIZE}, and {POSIX_REC_XFER_ALIGN} only applied to regular files, but note 10 also permits implementation of the advisory semantics on other file types unique to an implementation (e.g. a character special device.) "

**Change Number: XSH/TC1/36 [XBD ERN 27]**
On Page: 400 Line: 12991,13020 Section: fpathconf
Add to the end of the RETURN VALUE section as a new paragraph:
"If the variable corresponding to name is dependent on an unsupported option the results are unspecified."
Add to the end of the APPLICATION USAGE section:
"Application writers should check whether an option, such as _POSIX_ADVISORY_INFO is supported, prior to obtaining and using values for related variables such as POSIX_ALLOC_SIZE_MIN."

**Change Number: XSH/TC1/37 [XSH ERN 16]**
On Page: 416 Line: 13661 Section: fputc
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the process would be delayed in the write operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the thread would be delayed in the write operation. [/CX]"

**Change Number: XSH/TC1/38 [XSH ERN 17]**
On Page: 420 Line: 13778 Section: fputwc
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the process would be delayed in the write operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor underlying stream and the thread would be delayed in the write operation. [/CX]"

**Change Number: XSH/TC1/39 [XSH ERN 126]**
On Page: 428 Line: 14076 Section: freeaddrinfo
In the RETURN VALUE section
Change From:
"The getaddrinfo() function shall fail and return the corresponding value if:"
To:
"The getaddrinfo() function shall fail and return the corresponding error value if:"

**Change Number: XSH/TC1/40 [XSH ERN 142]**
On Page: 430 Line: 14141-14144 Section: freopen
In the DESCRIPTION section
Change From:
"If filename is a null pointer, the freopen() function attempt to change the mode of the stream to that by mode, as if the name of the file currently with the stream had been used. It is implementation-defined which changes of mode are permitted (if any) and under what circumstances".
To:
"If filename is a null pointer, the freopen() function shall attempt to change the mode of the stream to that specified by mode, as if the name of the file currently associated with the stream had been used. In this case, the file descriptor associated with the stream need not be closed if the call to freopen() succeeds. It is implementation-defined which changes of mode are permitted (if any) and under what circumstances".

**Change Number: XSH/TC1/41 [XSH ERN 142]**
On Page: 430,431 Line: 14158,14179 Section: freopen
In the ERRORS section
Add after line 14158
"[CX][EBADF] The file descriptor underlying the stream is not a valid file descriptor when filename is a null pointer.[/CX]"
Add after line 14179
"[CX][EBADF] The mode with which the file descriptor underlying the stream was opened does not support the requested mode when filename is a null pointer.[/CX]"

**Change Number: XSH/TC1/42 [XSH ERN 18]**
On Page: 443 Line: 14598 Section: fseek
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the process would be delayed in the write operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the thread would be delayed in the write operation. [/CX]"

**Change Number: XSH/TC1/43 [XSH ERN 19]**
On Page: 445 Line: 14679 Section: fsetpos
In the ERRORS section
Change From:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the process would be delayed in the write operation. [/CX]"
To:
"[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the thread would be delayed in the write operation. [/CX]"

**Change Number: XSH/TC1/44 [XSH ERN 143]**
On Page: 452 Line: 14894-14896 Section: fsync
In the DESCRIPTION section
Change From:
"The fsync() function shall request that all data for the open file descriptor named by fildes is to be transferred to the storage device associated with the file described by fildes in an implementation-defined manner."
To:
"The fsync() function shall request that all data for the open file descriptor named by fildes is to be transferred to the storage device associated with the file described by fildes. The nature of the transfer is implementation-defined."
Rationale: This is an editorial wording change with no change intended in meaning.

**Change Number: XSH/TC1/45 [XSH ERN 53]**
On Page: 491 Line: 16190,16229 Section: getcontext
In the SYNOPSIS section
Change the margin marker from "XSI" to "OB XSI", to denote that the getcontext() and setcontext() functions are obsolescent.
In the APPLICATION USAGE section for getcontext add at the end:
"The obsolescent functions getcontext(), makecontext() and swapcontext() can be replaced using POSIX threads functions."

**Change Number: XSH/TC1/46 [XSH ERN 20]**
On Page: 554,556 Line: 18219,18266 Section: getrlimit
In the DESCRIPTION section
Change From:
"RLIMIT_STACK This is the maximum size of a process stack, in bytes."
To:
"RLIMIT_STACK This is the maximum size of the initial thread's stack, in bytes."
In the RATIONALE section
Change From:
"None."
To:
"It should be noted that RLIMIT_STACK applies "at least" to the stack of the initial thread in the process, and not to the sum of all the stacks in the process, as that would be very limiting unless the value is so big as to provide no value at all with a single thread."

**Change Number: XSH/TC1/47 [XSH ERN 21,22,23]**
On Page: 563 Line: 18492-18497 Section: getsockopt
Change From:
"If SO_LINGER is set, the system blocks the process during close() until it can transmit the data or until the end of the interval indicated by the l_linger member, whichever comes first. If SO_LINGER is not specified, and close() is issued, the system handles the call in a way that allows the process to continue as quickly as possible."
To:
"If SO_LINGER is set, the system shall block the calling thread during close() until it can transmit the data or until the end of the interval indicated by the l_linger member, whichever comes first. If SO_LINGER is not specified, and close() is issued, the system handles the call in a way that allows the calling thread to continue as quickly as possible."

**Change Number: XSH/TC1/48 [XSH ERN 48]**
On Page: 580 Line: 19104-19108 Section: gmtime
In the RETURN VALUE section
Change From:
"Upon successful completion, gmtime_r() shall return the address of the structure pointed to by the argument result. If an error is detected, gmtime_r() shall return a null pointer."
To:
"Upon successful completion, gmtime_r() shall return the address of the structure pointed to by the argument result. If an error is detected, gmtime_r() shall return a null pointer and set errno to indicate the error".
In the ERRORS section:
Change From:
"The gmtime() function shall fail if:
[EOVERFLOW] The result cannot be represented."
To:
"The gmtime() and gmtime_r() function shall fail if:
[EOVERFLOW] The result cannot be represented."
with "and gmtime_r()" shaded and margin marked TSF.

**Change Number: XSH/TC1/49 [TC2d5 ERN 3]**
On Page: 589 Line: 19374 Section: hypot
In the EXAMPLES section
Change From:
"None."
To:
"See atan2() EXAMPLES."

**Change Number: XSH/TC1/50 [XSH ERN 24]**
On Page: 648 Line: 21400 Section: isunordered
Change From:
"If x or y is NaN, 0 shall be returned."
To:
"If x or y is NaN, 1 shall be returned."
Rationale:
The previous statement is incorrect. If x is NaN, then isunordered() must return 1. Likewise if y is NaN, then isunordered() must return 1.

**Change Number: XSH/TC1/51 [XSH ERN 144]**
On Page: 669 Line: 22175-22176 Section: kill
In the RATIONALE section
Change From:
"The implementation-defined processes to which a signal cannot be sent may include the scheduler or init."
To:
"The unspecified processes to which a signal cannot be sent may include the scheduler or init."
Rationale: The RATIONALE section conflicted with the normative text in the DESCRIPTION section.

**Change Number: XSH/TC1/52 [XSH ERN 135]**
On Page: 671 Line: 22533 Section: killpg
In the EXAMPLES section
Change From:
"None."
To:
"Sending a signal to all other members of a process group
The following example shows how the calling process could send a signal to all other members of its process group. To prevent itself from receiving the signal it first makes itself immune to the signal by ignoring it.

```
#include <signal.h>

#include <unistd.h>

...

    if (signal(SIGUSR1, SIG_IGN) == SIG_ERR)

        /* Handle error */;

    if (killpg(getpgrp(), SIGUSR1) == -1)

        /* Handle error */;"
```

**Change Number: XSH/TC1/53 [XSH ERN 78]**
On Page: 686 Line: 22699 Section: lio_listio
In the DESCRIPTION section
Add after line 22699
"If the buffer pointed to by list or the aiocb structures pointed to by the elements of the array list become illegal addresses before all asynchronous I/O completed and, if necessary, the notification is sent, then the behavior is undefined. If the buffers pointed to by the aio_buf member of the aiocb structure pointed to by the elements of the array list become illegal addresses prior to the asynchronous I/O associated with that aiocb structure being completed, the behavior is undefined."
Rationale: This text is added for symmetry with the aio_read() and aio_write() functions.

**Change Number: XSH/TC1/54 [XSH ERN 79]**
On Page: 686 Line: 22719 Section: lio_listio
In the DESCRIPTION Section (after the last paragraph)
Add after line 22719
"If sig->sigev_notify is SIGEV_THREAD and sig->sigev_notify_attributes is a non-NULL pointer and the block pointed to by this pointer becomes an illegal address prior to all asynchronous I/O being completed, then the behavior is undefined."
Rationale:
This text is added to make it explicit that the user is required to keep the structure pointed to by sig->sigev_notify_attributes valid until the last asynchronous operation finished and the notification has been sent.

**Change Number: XSH/TC1/55 [XSH ERN 49]**
On Page: 702 Line: 23186-23190 Section: localtime
In the RETURN VALUE
Change From:
"Upon successful completion, localtime_r() shall return a pointer to the structure pointed to by the argument result."
To:
"Upon successful completion, localtime_r() shall return a pointer to the structure pointed to by the argument result. If an error is detected, localtime_r() shall return a null pointer and set errno to indicate the error."
In the ERRORS section
Change From:
"The localtime() function shall fail if:
[EOVERFLOW] The result cannot be represented."
To:
"The localtime() and localtime_r() functions shall fail if:
[EOVERFLOW] The result cannot be represented."
with "and localtime_r()" shaded and margin marked TSF.

**Change Number: XSH/TC1/56 [XSH ERN 84,85]**
On Page: 702,704 Line: 23181,23245 Section: localtime
In the DESCRIPTION section
Add a new paragraph following the last paragraph with TSF and XSI shading:
"If the reentrant version does not set tzname, it shall not set daylight and shall not set timezone."
In the SEE ALSO section
Add "tzset()," before "utime(),".
Rationale:
On systems supporting XSI, the daylight, timezone, and tzname variables should all be set to provide information for the same timezone.
This updates the description of localtime_r() to mention daylight and timezone as well as tzname.

**Change Number: XSH/TC1/57 [XSH ERN 53]**
On Page: 732,733 Line: 24081-24083,24143-24146 Section: makecontext
In the SYNOPSIS section
Change the "XSI" margin marker to "OB XSI"
Change the function prototype from:
"void makecontext(ucontext_t *ucp, void (*func)(void), int argc, ...);"
To:
"void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...);"
In the APPLICATION USAGE section
Change From:
"None."
To:
"The obsolescent functions getcontext(), makecontext() and swapcontext() can be replaced using POSIX threads functions."
In the RATIONALE section
Change From:
"None."
To:
"With the incorporation of ISO/IEC 9899:1999 (C99) into this specification it was found that the ISO C Standard (subclause 6.11.6) specifies that the use of function declarators with empty parentheses is an obsolescent feature. Therefore, using the function prototype:
void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...);
is making use of an obsolescent feature of ISO C. Therefore, a strictly conforming POSIX application can't use this form. Therefore, use of getcontext(), makecontext() and swapcontext() is marked obsolescent.
There is no way in ISO C to specify a non-obsolescent function prototype indicating that a function will be called with an arbitrary number (including zero) of arguments of arbitrary types (including integers, pointers to data, pointers to functions, and composite types). Replacing makecontext() with a number of ISO C compatible functions handing various numbers and types of arguments would have forced all existing uses of makecontext() to be rewritten for little or no gain.
There are very few applications today that use the *context() routines. Those that do use them are almost always using them to implement co-routines. By maintaining the XSH5 specification for makecontext(), existing applications will continue to work, although they won't be able to be classified as strictly conforming applications.
There is no way in ISO C (without using obsolescent behavior) to specify functionality that was standard, strictly conforming behavior in XSH5 using the 1990 C Standard. Threads can be used to implement the functionality provided by makecontext(), getcontext() and swapcontext() but they are more complex to use. It was felt inventing new ISO C compatible interfaces that describe what can be done with the XSH5 functions and then converting applications to use them would cause more difficulty than just converting applications that use them to use threads instead."

**Change Number: XSH/TC1/58 [XSH ERN 50]**
On Page: 768 Line: 25150-25152,25154 Section: mktime
In the RETURN VALUE section
Change From:
"The mktime() function shall return the specified time since the Epoch encoded as a value of type time_t. If the time since the Epoch cannot be represented, the function shall return the value (time_t)-1."
To:
"The mktime() function shall return the specified time since the Epoch encoded as a value of type time_t. If the time since the Epoch cannot be represented, the function shall return the value (time_t)-1 [CX] and may set errno to indicate the error.[/CX]."
In the ERRORS section
Change From:
"None."
To:
(Shaded and marked with the CX margin code):
"The mktime() function may fail if:
[EOVERFLOW] The result cannot be represented."

**Change Number: XSH/TC1/59 [XSH ERN 86]**
On Page: 769 Line: 25185 Section: mktime
In the SEE ALSO section
Add "tzset()," before "utime(), "
Rationale: This is editorial.

**Change Number: XSH/TC1/60 [XSH ERN 72]**
On Page: 774,777 Line: 25357,25469 Section: mmap
In the DESCRIPTION section
Add a new paragraph after line 25357
"If len is zero mmap() shall fail and no mapping shall be established."
In the ERRORS section (shall fail)
Add after line 25469
"[EINVAL] The value of len is zero."

**Change Number: XSH/TC1/61 [XSH ERN 47]**
On Page: 786 Line: 25817 Section: mq_getattr
In the ERRORS section
Change From:
"The mq_getattr() function shall fail if:"
To:
"The mq_getattr() function may fail if:"

**Change Number: XSH/TC1/62 [XSH ERN 87]**
On Page: 791 Line: 25933 Section: mq_open
In the DESCRIPTION section (O_CREAT mode)
Change From:
"The file permission bits shall be set to the value of mode. When bits in mode other than file permission bits are set, the effect is implementation-defined."
To:
"The permission bits of the message queue shall be set to the value of the mode argument except those set in the file mode creation mask of the process. When bits in mode other than the file permission bits are specified, the effect is unspecified."
Rationale: Consistency with shm_open(), sem_open().

**Change Number: XSH/TC1/63 [XSH ERN 25]**
On Page: 823 Line: 26879-26880 Section: nanosleep
In the RATIONALE section
Change From:
"It is common to suspend execution of a process for an interval in order to poll the status of a non-interrupting function."
To:
"It is common to suspend execution of a thread for an interval in order to poll the status of a non-interrupting function."

**Change Number: XSH/TC1/64 [XSH ERN 73]**
On Page: 829-831 Line: 27026,27069,27102,27114 Section: nftw
In the SYNOPSIS section, line 27026
Change From:
"int nftw(const char *path, int (*fn)(const char *, const struct stat *,
int, struct FTW *), int depth, int flags);"
To:
"int nftw(const char *path, int (*fn)(const char *, const struct
stat *, int, struct FTW *), int fd_limit, int flags); "
In the DESCRIPTION section, line 27069
Change From:
"The argument depth sets the maximum number of file descriptors that shall be used by nftw() while traversing the file tree."

To:
"The argument fd_limit sets the maximum number of file descriptors that shall be used by nftw() while traversing the file tree."
In the EXAMPLES section, line 27102
Change From:
"The following example walks the /tmp directory and its subdirectories, calling the nftw() function for every directory entry, to a maximum of 5 levels deep."
To:
"The following example walks the /tmp directory and its subdirectories, calling the nftw() function for every directory entry, using a maximum of 5 file descriptors."
On line 27114
Change From:

```
"int depth = 5;

int flags = FTW_CHDIR | FTW_DEPTH | FTW_MOUNT;

int ret;

ret = nftw(startpath, nftwfunc, depth, flags);"
```

To:

```
"int fd_limit = 5;

int flags = FTW_CHDIR | FTW_DEPTH | FTW_MOUNT;

int ret;

ret = nftw(startpath, nftwfunc, fd_limit, flags);"
```

**Change Number: XSH/TC1/65 [XSH ERN 127]**
On Page: 856 Line: 27866 Section: pipe
In the EXAMPLES section
Change From:
"None."
To:
"Using a pipe to pass data between a parent process and a child process
The following example demonstrates the use of a pipe to transfer data between a parent process and a child process. Error handling is excluded, but otherwise this code demonstrates good practice when using pipes: after the fork() the two processes close the unused ends of the pipe before they commence transferring data.

```
#include <stdlib.h>

#include <unistd.h>

 ...

 int fildes[2];

 const int BSIZE = 100;

 char buf[BSIZE];

 ssize_t nbytes;

 int status;

 status = pipe(fildes);
```

```
if (status == -1 ) {
   /* an error occurred */
   ....
}
switch (fork()) {
case -1: /* Handle error */
    break;
case 0:     /* Child  - reads from pipe */
    close(fildes[1]);                       /* Write end is unused */
    nbytes = read(fildes[0], buf, BSIZE);   /* Get data from pipe */
    /* At this point, a further read would see end of file.. */
    close(fildes[0]);                       /* Finished with pipe */
    exit(EXIT_SUCCESS);
default:    /* Parent - writes to pipe */
    close(fildes[0]);                       /* Read end is unused */
    write(fildes[1], "Hello world\n", 12);  /* Write data on pipe */
    close(fildes[1]);                       /* Child will see EOF */
    exit(EXIT_SUCCESS);
}"
```

**Change Number: XSH/TC1/66 [XSH ERN 145]**
On Page: 860 Line: 27995-27999 Section: poll
In the EXAMPLES section
Change From:

```
"/* Open STREAMS device. */
 fds[0].fd = open("/dev/dev0", ...);
 fds[1].fd = open("/dev/dev1", ...);
     fds[0].events = POLLOUT | POLLWRBAND;
     fds[1].events = POLLOUT | POLLWRBAND;"
```

To:
"/* Open STREAMS device. */

```
 fds[0].fd = open("/dev/dev0", ...);

 fds[1].fd = open("/dev/dev1", ...);

 fds[0].events = POLLOUT | POLLWRBAND;

 fds[1].events = POLLOUT | POLLWRBAND;"
```

**Change Number: XSH/TC1/67 [XSH ERN 128]**
On Page: 863 Line: 28077 Section: popen
In the EXAMPLES section
Change From:
"None."
To:
"Using popen() to obtain a list of files from the ls utility
The following example demonstrates the use of popen() and pclose() to execute the command "ls *" in order to obtain a list of files in the current directory:

```
#include <stdio.h>

 ...

 FILE *fp;

 int status;

 char path[PATH_MAX];

 fp = popen("ls *", "r");

 if (fp == NULL)

     /* Handle error */;

 while (fgets(path, PATH_MAX, fp) != NULL)

     printf("%s", path);

 status = pclose(fp);

 if (status == -1) {

     /* Error reported by pclose() */

     ...

 } else {

     /* Use macros described under wait() to inspect 'status' in order

        to determine success/failure of command executed by popen() */

     ...

 }"
```

**Change Number: XSH/TC1/68 [XSH ERN 146]**
On Page: 864 Line: 28122 Section: posix_fadvise
In the SYNOPSIS section
Change From:
"int posix_fadvise(int fd, off_t offset, size_t len, int advice);"
To:
"int posix_fadvise(int fd, off_t offset, off_t len, int advice);"
Rationale: The previous prototype was not large-file aware, and the standard developers felt it acceptable to make this change before implementations of the functions become widespread."

**Change Number: XSH/TC1/69 [XSH ERN 146]**
On Page: 866 Line: 28177 Section: posix_fallocate
In the SYNOPSIS section
Change From:
"int posix_fallocate(int fd, off_t offset, size_t len);"
To:
"int posix_fallocate(int fd, off_t offset, off_t len);"
Rationale: The previous prototype was not large-file aware, and the standard developers felt it acceptable to make this change before implementations of the functions become widespread."

**Change Number: XSH/TC1/70 [XSH ERN 110]**
On Page: 975 Line: 31069-31071 Section: pselect
In the DESCRIPTION section
Change From:
"If sigmask is not a null pointer, then the pselect() function shall replace the signal mask of the process by the set of signals pointed to by sigmask before examining the descriptors, and shall restore the signal mask of the process before returning."
To:
"If sigmask is not a null pointer, then the pselect() function shall replace the signal mask of the caller by the set of signals pointed to by sigmask before examining the descriptors, and shall restore the signal mask of the calling thread before returning."

**Change Number: XSH/TC1/71 [XSH ERN 147,148]**
On Page: 981 Line: 31293 Section: pthread_attr_destroy
In the ERRORS section
Add
"The pthread_attr_destroy() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object.
The pthread_attr_init() may fail if:
[EBUSY] The implementation has detected an attempt to reinitialize the thread attribute referenced by attr, a previously initialized, but not yet destroyed, thread attribute."

**Change Number: XSH/TC1/72 [XSH ERN 125]**
On Page: 984 Line: 31407 Section: pthread_attr_getdetachstate
In the EXAMPLES section
Change From:
"None."
To:
"Retrieving the detachstate attribute.
This example shows how to obtain the detachstate attribute of a thread attribute object.

```
#include <pthread.h>

pthread_attr_t thread_attr;

int            detachstate;

int            rc;
```

```
/* code initializing thread_attr */

...

rc = pthread_attr_getdetachstate (&thread_attr, &detachstate);

if (rc!=0) {

  /* handle error */

  ...

}

else {

  /* legal values for detachstate are:

   * PTHREAD_CREATE_DETACHED or PTHREAD_CREATE_JOINABLE

   */

   ...

}"
```

**Change Number: XSH/TC1/73 [XSH ERN 149]**
On Page: 984 Line: 31404 Section: pthread_attr_getdetachstate
In the ERRORS section
Add
"The pthread_attr_getdetachstate() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object.
The pthread_attr_setdetachstate() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."

**Change Number: XSH/TC1/74 [XSH ERN 150]**
On Page: 986 Line: 31461,31462 Section: pthread_attr_getdetachstate
In the ERRORS section
On line 31461 delete:
"[EINVAL] The attribute attr is invalid."
In the ERRORS section
Add:
"The pthread_attr_getguardsize() and pthread_attr_setguardsize may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."

**Change Number: XSH/TC1/75 [XSH ERN 151,152]**
On Page: 986 Line: 31500,31516,31517 Section: pthread_attr_getinheritsched
In the DESCRIPTION section
After line 31500 insert a new paragraph:
"The supported values of inheritsched shall be:"
In the ERRORS section
Insert before line 31516
"The pthread_attr_getinheritsched() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."
In the ERRORS section
Add after line 31517 :
"[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."

**Change Number: XSH/TC1/76 [XSH ERN 124]**
On Page: 987 Line: 31465 Section: pthread_attr_getguardsize
In the EXAMPLES section
Change From:
"None."
To:
"Retrieving the guardsize attribute.
This example shows how to obtain the guardsize attribute of a thread attribute object.

```
#include <pthread.h>

pthread_attr_t thread_attr;

size_t  guardsize;

int     rc;

/* code initializing thread_attr */

...

rc = pthread_attr_getguardsize (&thread_attr, &guardsize);

if (rc != 0)  {

  /* handle error */

  ...

}

else {

  if (guardsize > 0) {

    /* a guard area of at least guardsize bytes is provided */

    ...

  }

  else {

    /* no guard area provided */

    ...

  }

}"
```



**Change Number: XSH/TC1/77 [XSH ERN 55]**
On Page: 988 Line: 31524 Section: pthread_attr_getinheritsched
In the APPLICATION USAGE section
Add:
"See Section 2.9.4 for further details on thread scheduling attributes and their default settings."

**Change Number: XSH/TC1/78 [XSH ERN 154]**
On Page: 990 Line: 31567,31569 Section: pthread_attr_getschedparam
In the ERRORS section
Add after line 31567 :
"The pthread_attr_getschedparam() function may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."
On line 31569
Change From:
"[EINVAL] The value of param is not valid."
To:
"[EINVAL] The value of param is not valid, or the value specified by attr does not refer to an initialized thread attribute object."

**Change Number: XSH/TC1/79 [XSH ERN 56]**
On Page: 992 Line: 31621 Section: pthread_attr_getschedpolicy
In the APPLICATION USAGE section
Add:
"See Section 2.9.4 for further details on thread scheduling attributes and their default settings."

**Change Number: XSH/TC1/80 [XSH ERN 155]**
On Page: 992 Line: 31612,31614 Section: pthread_attr_getschedpolicy
In the ERRORS section
Add after line 31612 :
"The pthread_attr_getschedpolicy() function may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."
On line 31614
Change From:
"[EINVAL] The value of policy is not valid."
To:
"[EINVAL] The value of policy is not valid, or the value specified by attr does not refer to an initialized thread attribute object."

**Change Number: XSH/TC1/81 [XSH ERN 57]**
On Page: 994 Line: 31669 Section: pthread_attr_getscope
In the APPLICATION USAGE section
Add:
"See Section 2.9.4 for further details on thread scheduling attributes and their default settings."

**Change Number: XSH/TC1/82 [XSH ERN 157]**
On Page: 994 Line: 31660,31662 Section: pthread_attr_getscope
In the ERRORS section
Add after line 31660 :
"The pthread_attr_getscope() function may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."
On line 31662
Change From:
"[EINVAL] The value of contentionscope is not valid."
To:
"[EINVAL] The value of contentionscope is not valid, or the value specified by attr does not refer to an initialized thread attribute object."

**Change Number: XSH/TC1/83 [XSH ERN 89]**
On Page: 996 Line: 31731 Section: pthread_attr_getstack
In the APPLICATION USAGE
Add at the end:
"After a successful call to pthread_attr_setstack(), the storage area specified by the stackaddr parameter is under the control of the implementation as described in Section 2.9.8."

**Change Number: XSH/TC1/84 [XSH ERN 158]**
On Page: 996 Line: 31714,31716 Section: pthread_attr_getstack
In the ERRORS section
Add after line 31774 :
"The pthread_attr_getstack() function may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."
On line 31716
Change From:
"[EINVAL] The value of stackaddr does not have proper alignment to be used as a stack, or if ( stackaddr + stacksize) lacks proper alignment."
To:
"[EINVAL] The value of stackaddr does not have proper alignment to be used as a stack, or ( stackaddr + stacksize) lacks proper alignment, or the value specified by attr does not refer to an initialized thread attribute object."

**Change Number: XSH/TC1/85 [XSH ERN 89]**
On Page: 998 Line: 31777 Section: pthread_attr_getstackaddr
In the APPLICATION USAGE
Add at the end:
"After a successful call to pthread_attr_setstackaddr(), the storage area specified by the stackaddr parameter is under the control of the implementation as described in Section 2.9.8."

**Change Number: XSH/TC1/86 [XSH ERN 159]**
On Page: 998 Line: 31761 Section: pthread_attr_getstackaddr
In the ERRORS section
Change From:
"No errors are defined."
To:
"The pthread_attr_getstackaddr() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object.
The pthread_attr_setstackaddr() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."

**Change Number: XSH/TC1/87 [XSH ERN 160]**
On Page: 1000 Line: 31815 Section: pthread_attr_getstacksize
In the ERRORS section
Add:
"The pthread_attr_getstacksize() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object.
The pthread_attr_setstacksize() may fail if:
[EINVAL] The value specified by attr does not refer to an initialised thread attribute object."

**Change Number: XSH/TC1/88 [XSH ERN 64]**
On Page: 1024 Line: 32233 Section: pthread_cleanup_pop
In the DESCRIPTION section
Add a new final paragraph:
"The effect of the use of 'return', 'break', 'continue', and 'goto' to prematurely leave a code block described by a pair of pthread_cleanup_push() and pthread_cleanup_pop() functions calls is undefined."

**Change Number: XSH/TC1/89 [XSH ERN 90]**
On Page: 1036 Line: 32652-32655 Section: pthread_cond_timedwait
In the DESCRIPTION section
Change From:
"The effect of using more than one mutex for concurrent pthread_cond_timedwait() or pthread_cond_wait() operations on the same condition variable is undefined; that is, a condition variable becomes bound to a unique mutex when a thread waits on the condition variable, and this (dynamic) binding shall end when the wait returns."

To:
"When a thread waits on a condition variable, having specified a particular mutex to either the pthread_cond_timedwait() or the pthread_cond_wait() operation, a dynamic binding is formed between that mutex and condition variable that remains in effect as long as at least one thread is blocked on the condition variable. During this time, the effect of an attempt by any thread to wait on that condition variable using a different mutex is undefined. Once all waiting threads have been unblocked (as by the pthread_cond_broadcast() operation), the next wait operation on that condition variable shall form a new dynamic binding with the mutex specified by that wait operation. Even though the dynamic binding between condition variable and mutex may be removed or replaced between the time a thread is unblocked from a wait on the condition variable and the time that it returns to the caller or begins cancellation cleanup, the unblocked thread shall always re-acquire the mutex specified in the condition wait operation call from which it is returning."
Rationale:
This change is for consistency with pthread_cond_destroy() which says it is safe to destroy an initialized condition variable upon which no threads are currently blocked.

**Change Number: XSH/TC1/90 [XSH ERN 58]**
On Page: 1036 Line: 32656-32659 Section: pthread_cond_timedwait
In the DESCRIPTION section
Change From:
"When the cancelability enable state of a thread is set to PTHREAD_CANCEL_DEFERRED, a side effect of acting upon a cancellation request while in a condition wait is that the mutex is (in effect) re-acquired before calling the first cancellation cleanup handler."
To:
"When the cancelability type of a thread is set to PTHREAD_CANCEL_DEFERRED, a side effect of acting upon a cancellation request while in a condition wait is that the mutex is (in effect) re-acquired before calling the first cancellation cleanup handler."

**Change Number: XSH/TC1/91 [XSH ERN 26]**
On Page: 1037 Line: 32690-32693 Section: pthread_cond_timedwait
In the ERRORS section
Change From:
"The pthread_cond_timedwait() and pthread_cond_wait() functions may fail if:
[EINVAL] The value specified by cond, mutex,or abstime is invalid.
[EINVAL] Different mutexes were supplied for concurrent pthread_cond_timedwait() or pthread_cond_wait() operations on the same condition variable."
To:
"The pthread_cond_timedwait() function shall fail if:
[ETIMEDOUT] The time specified by abstime to pthread_cond_timedwait() has passed.
[EINVAL] The value specified by abstime is invalid.
The pthread_cond_timedwait() and pthread_cond_wait() functions may fail if:
[EINVAL] The value specified by cond or mutex is invalid."

**Change Number: XSH/TC1/92 [XSH ERN 118]**
On Page: 1037 Line: 32705 Section: pthread_cond_timedwait
In the RATIONALE section
Add a new second paragraph:
"The application needs to recheck the predicate on any return because it cannot be sure there is another thread waiting on the thread to handle the signal, and if there is not then the signal is lost. The burden is on the application to check the predicate."

**Change Number: XSH/TC1/93 [XSH ERN 136]**
On Page: 1050 Line: 33036,33038 Section: pthread_create
In the ERROR RETURN section
Delete the following from the "shall fail" section:
"[EINVAL] The value specified by attr is invalid."
Add after line 33038 :
"The pthread_create() function may fail if:
[EINVAL] The attributes specified by attr are invalid."

**Change Number: XSH/TC1/94 [XSH ERN 91]**
On Page: 1051 Line: 33043 Section: pthread_create
In the APPLICATION USAGE section
Change From:
"None."
To:
"There is no requirement on the implementation that the ID of the created thread be available before the newly created thread starts executing. The calling thread can obtain the ID of the created thread through the return value of the pthread_create() function, and the newly created thread can obtain its ID by a call to pthread_self()."

**Change Number: XSH/TC1/95 [XSH ERN 27]**
On Page: 1053 Line: 33123-33127 Section: pthread_detach
In the ERRORS section
Change From:
"The pthread_detach() function shall fail if:
[EINVAL] The implementation has detected that the value specified by thread does not refer to a joinable thread.
[ESRCH] No thread could be found corresponding to that specified by the given thread ID."
To:
"The pthread_detach() function may fail if:
[EINVAL] The implementation has detected that the value specified by thread does not refer to a joinable thread.
[ESRCH] No thread could be found corresponding to that specified by the given thread ID."

**Change Number: XSH/TC1/96 [XSH ERN 100]**
On Page: 1064 Line: 33451 Section: pthread_getspecific
In the ERRORS Section
Change From:
"[ENOMEM] Insufficient memory exists to associate the value with the key."
To:
"[ENOMEM] Insufficient memory exists to associate the non-NULL value with the key."

**Change Number: XSH/TC1/97 [XSH ERN 28]**
On Page: 1066 Line: 33494-33500 Section: pthread_join
Change From:
"The pthread_join() function shall fail if:
[EINVAL] The implementation has detected that the value specified by thread does not refer to a joinable thread.
[ESRCH] No thread could be found corresponding to that specified by the given thread ID.
The pthread_join() function may fail if:
[EDEADLK] A deadlock was detected or the value of thread specifies the calling thread."
To:
"The pthread_join() function shall fail if:
[ESRCH] No thread could be found corresponding to that specified by the given thread ID.
The pthread_join() function may fail if:
[EDEADLK] A deadlock was detected or the value of thread specifies the calling thread.
[EINVAL] The value specified by thread does not refer to a joinable thread."

**Change Number: XSH/TC1/98 [XSH ERN 59,65]**
On Page: 1085 Line: 34135,34150 Section: pthread_mutex_lock
In the ERRORS section
Change From:
"[EDEADLK] The current thread already owns the mutex."
To:
"[EDEADLK] A deadlock condition was detected or the current thread already owns the mutex."
In the RATIONALE section
Change From:
"For example, deadlocking on a double-lock is explicitly allowed behavior in order to avoid requiring more overhead in the basic mechanism than is absolutely necessary."

To:
"For example, on systems not supporting the XSI extended mutex types, deadlocking on a double-lock is explicitly allowed behavior in order to avoid requiring more overhead in the basic mechanism than is absolutely necessary."
Add to the end of the RATIONALE section:
"For further rationale on the XSI extended mutex types please see the Rationale Volume."

**Change Number: XSH/TC1/99 [XSH ERN 60]**
On Page: 1088 Line: 34211-34214 Section: pthread_mutex_timedlock
In the DESCRIPTION section (last paragraph)
Margin mark the following text TPI and shade:
"As a consequence of the priority inheritance rules (for mutexes initialized with the PRIO_INHERIT protocol), if a timed mutex wait is terminated because its timeout expires, the priority of the owner of the mutex shall be adjusted as necessary to reflect the fact that this thread is no longer among the threads waiting for the mutex."

**Change Number: XSH/TC1/100 [XSH ERN 65]**
On Page: 1089 Line: 34231 Section: pthread_mutex_timedlock
In the ERRORS section
Change From:
"[EDEADLK] The current thread already owns the mutex."
To:
"[EDEADLK] A deadlock condition was detected or the current thread already owns the mutex."

**Change Number: XSH/TC1/101 [XSH ERN 65]**
On Page: 1114 Line: 34988 Section: pthread_rwlock_rdlock
In the ERRORS section
Change From:
"[EDEADLK] The current thread already owns the read-write lock for writing."
To:
"[EDEADLK] A deadlock condition was detected or the current thread already owns the read-write lock for writing."

**Change Number: XSH/TC1/102 [XSH ERN 65]**
On Page: 1115 Line: 35054 Section: pthread_rwlock_timedrdlock
In the ERRORS section
Change From:
"[EDEADLK] The calling thread already holds a write lock on rwlock."
To:
"[EDEADLK] A deadlock condition was detected or the calling thread already holds a write lock on rwlock."

**Change Number: XSH/TC1/103 [XSH ERN 65]**
On Page: 1117 Line: 35112 Section: pthread_rwlock_timedwrlock
In the ERRORS section
Change From:
"[EDEADLK] The calling thread already holds the rwlock."
To:
"[EDEADLK] A deadlock condition was detected or the calling thread already holds the rwlock."

**Change Number: XSH/TC1/104 [XSH ERN 65]**
On Page: 1120 Line: 35178 Section: pthread_rwlock_trywrlock
In the ERRORS section
Change From:
"[EDEADLK] The current thread already owns the read-write lock for writing or reading."
To:
"[EDEADLK] A deadlock condition was detected or the current thread already owns the read-write lock for writing or reading."

**Change Number: XSH/TC1/105 [XSH ERN 30]**
On Page: 1139-1140 Line: 35589,35615 Section: pthread_sigmask
In the DESCRIPTION section, line 35589
Change From:
"If set is a null pointer, the value of the argument how is not significant and the process signal mask shall be unchanged; thus the call can be used to enquire about currently blocked signals."
To:
"If set is a null pointer, the value of the argument how is not significant and the thread signal mask shall be unchanged; thus the call can be used to enquire about currently blocked signals."
In the RATIONALE section, line 35615
Change From:
"When a process signal mask is changed in a signal-catching function that is installed by sigaction(), the restoration of the signal mask on return from the signal-catching function overrides that change (see sigaction())."
To:
"When a thread's signal mask is changed in a signal-catching function that is installed by sigaction(), the restoration of the signal mask on return from the signal-catching function overrides that change (see sigaction())."

**Change Number: XSH/TC1/106 [XSH ERN 114]**
On Page: 1140 Line: 35611 Section: pthread_sigmask
In the EXAMPLES section
Change From:
"None."
To:
"Signalling in multi-threaded process.
This example shows the use of pthread_sigmask() in order to deal with signals in a multi-threaded process. It provides a fairly general framework that could be easily adapted/extended by the reader.

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <signal.h>

#include <string.h>

#include <errno.h>

...

    static sigset_t   signal_mask;  /* signals to block         */

    int main (int argc, char *argv[])

    {

      pthread_t  sig_thr_id;      /* signal handler thread ID */

        int        rc;              /* return code              */

        sigemptyset (&signal_mask);

        sigaddset (&signal_mask, SIGINT);

        sigaddset (&signal_mask, SIGTERM);

        rc = pthread_sigmask (SIG_BLOCK, &signal_mask, NULL);
```

```
    if (rc != 0) {
        /* handle error */
      ...
    }
    /* any newly created threads inherit the signal mask */
    rc = pthread_create (&sig_thr_id, NULL, signal_thread, NULL);
    if (rc != 0) {
        /* handle error */
        ...
    }
    /* APPLICATION CODE */
    ...
}
void *signal_thread (void *arg)
{
    int       sig_caught;    /* signal caught       */
    int       rc;            /* returned code       */
    rc = sigwait (&signal_mask, &sig_caught);
    if (rc != 0) {
        /* handle error */
    }
    switch (sig_caught)
    {
    case SIGINT:     /* process SIGINT  */
        ...
        break;
    case SIGTERM:    /* process SIGTERM */
        ...
        break;
    default:         /* should normally not happen */
```

```
            fprintf (stderr, "\nUnexpected signal %d\n", sig_caught);
            break;
        }
    }
```

**Change Number: XSH/TC1/107 [XSH ERN 65]**
On Page: 1143 Line: 35723 Section: pthread_spin_lock
In the ERRORS section
Change From:
"[EDEADLK] The calling thread already holds the lock."
To:
"[EDEADLK] A deadlock condition was detected or the calling thread already holds the lock."

**Change Number: XSH/TC1/108 [XSH ERN 43]**
On Page: 1174 Line: 36533 Section: read
In the ERRORS section
Change From:
"[EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the process would be delayed."
To:
"[EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the thread would be delayed."

**Change Number: XSH/TC1/109 [XSH ERN 165]**
On Page: 1177 Line: 36652-36654 Section: read
In the RATIONALE section
Change From:
"IEEE Std 1003.1-2001 does not specify when an implementation that buffers read()ss actually moves the data into the user-supplied buffer, so an implementation may chose to do this at the latest possible moment."
To:
"IEEE Std 1003.1-2001 does not specify when an implementation that buffers read()s actually moves the data into the user-supplied buffer, so an implementation may chose to do this at the latest possible moment."

**Change Number: XSH/TC1/110 [XSH ERN 111]**
On Page: 1190 Line: 37082,37091 Section: realpath()
Change From:
"[ELOOP] A loop exists in symbolic links encountered during resolution of the path argument."
To:
"[ELOOP] A loop exists in symbolic links encountered during resolution of the file_name argument."
Change From:
"[ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the path argument.
To:
"[ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the file_name argument.
Rationale:
These ERROR descriptions erroneously referred to a non-existent path argument.

**Change Number: XSH/TC1/111 [XSH Errata 2003-1]**
On Page: 1247 Line: 38887 Section: select
In the SYNOPSIS section
Change from:
"#include <sys/time.h>"
To:
"#include <sys/select.h>"
Rationale: This is an editorial correction to the pointer page to match the actual manual page.

**Change Number: XSH/TC1/112 [XSH ERN 66]**
On Page: 1248 Line: 38915 Section: sem_close
In the ERRORS section
Change From:
"The sem_close() function shall fail if:"
To:
"The sem_close() function may fail if:"

**Change Number: XSH/TC1/113 [XSH ERN 67]**
On Page: 1249 Line: 38952-38955 Section: sem_destroy
In the ERRORS section
Change From:
"The sem_destroy() function shall fail if:
[EINVAL] The sem argument is not a valid semaphore.
The sem_destroy() function may fail if:
[EBUSY] There are currently processes blocked on the semaphore."
To:
"The sem_destroy() function may fail if:
[EINVAL] The sem argument is not a valid semaphore.
[EBUSY] There are currently processes blocked on the semaphore."

**Change Number: XSH/TC1/114 [XSH ERN 68]**
On Page: 1250 Line: 38993 Section: sem_getvalue
In the ERRORS section
Change From:
"The sem_getvalue() function shall fail if:"
To:
"The sem_getvalue() function may fail if:"

**Change Number: XSH/TC1/115 [XSH ERN 102]**
On Page: 1252 Line: 39025-39036 Section: sem_init
In the DESCRIPTION section
Change From:
"The sem_init() function shall initialize the unnamed semaphore referred to by sem. The value of the initialized semaphore shall be value. Following a successful call to sem_init(), the semaphore may be used in subsequent calls to sem_wait(), sem_trywait(), sem_post(), and sem_destroy(). This semaphore shall remain usable until the semaphore is destroyed.
If the pshared argument has a non-zero value, then the semaphore is shared between processes; in this case, any process that can access the semaphore sem can use sem for performing sem_wait(),sem_trywait(),sem_post(), and sem_destroy() operations.
Only sem itself may be used for performing synchronization. The result of referring to copies of sem in calls to sem_wait(),sem_trywait(),sem_post(), and sem_destroy() is undefined.
If the pshared argument is zero, then the semaphore is shared between threads of the process; any thread in this process can use sem for performing sem_wait(), sem_trywait(), sem_post(), and sem_destroy() operations. "
To:
"The sem_init() function shall initialize the unnamed semaphore referred to by sem. The value of the initialized semaphore shall be value. Following a successful call to sem_init(), the semaphore may be used in subsequent calls to sem_wait(), [TMO]sem_timedwait(),[/TMO] sem_trywait(), sem_post(), and sem_destroy(). This semaphore shall remain usable until the semaphore is destroyed.
If the pshared argument has a non-zero value, then the semaphore is shared between processes; in this case, any process that can access the semaphore sem can use sem for performing sem_wait(), [TMO]sem_timedwait(),[/TMO]sem_trywait(), sem_post(), and sem_destroy() operations.
Only sem itself may be used for performing synchronization. The result of referring to copies of sem in calls to sem_wait(), [TMO]sem_timedwait(),[/TMO]sem_trywait(), sem_post(), and sem_destroy() is undefined.
If the pshared argument is zero, then the semaphore is shared between threads of the process; any thread in this process can use sem for performing sem_wait(), [TMO]sem_timedwait(),[/TMO]sem_trywait(), sem_post(), and sem_destroy() operations."
Rationale: The sem_timedwait() function is added to the DESCRIPTION for alignment with 1003.1d-1999.

**Change Number: XSH/TC1/116 [XSH ERN 103]**
On Page: 1254 Line: 39079-39080 Section: sem_open
In the DESCRIPTION section
Change From:
"This semaphore may be used in subsequent calls to sem_wait(), sem_trywait(), sem_post(), and sem_close(). "
To:
"This semaphore may be used in subsequent calls to sem_wait(), [TMO]sem_timedwait(),[/TMO] sem_trywait(), sem_post(), and sem_close(). "
Rationale: The sem_timedwait() function is added to the DESCRIPTION for alignment with 1003.1d-1999.

**Change Number: XSH/TC1/117 [XSH ERN 97]**
On Page: 1254-1255 Line: 39115-39117 Section: sem_open
In the DESCRIPTION section
Change From:
"If a process makes multiple successful calls to sem_open() with the same value for name, the same semaphore address shall be returned for each such successful call, provided that there have been no calls to sem_unlink() for this semaphore."
To:
"If a process makes multiple successful calls to sem_open() with the same value for name, the same semaphore address shall be returned for each such successful call, provided that there have been no calls to sem_unlink() for this semaphore and at least one previous successful sem_open() call for this semaphore has not been matched with sem_close() call."

**Change Number: XSH/TC1/118 [XSH ERN 69]**
On Page: 1257 Line: 39192 Section: sem_post
In the ERRORS section
Change From:
"The sem_post() function shall fail if:"
To:
"The sem_post() function may fail if:"

**Change Number: XSH/TC1/119 [XSH ERN 70]**
On Page: 1259 Line: 39247 Section: sem_timedwait
In the ERRORS section
Move the following text on line 39247 from the "shall fail" to the "may fail" section:
"[EINVAL] The sem argument does not refer to a valid semaphore.

**Change Number: XSH/TC1/120 [XSH ERN 71]**
On Page: 1261 Line: 39298 Section: sem_trywait
In the ERRORS section
Move the following text on line 39298 from the "shall fail" to the "may fail" section:
"[EINVAL] The sem argument does not refer to a valid semaphore."

**Change Number: XSH/TC1/121 [XSH ERN 133]**
On Page: 1269 Line: 39510 Section: semget
In the DESCRIPTION section
Change From:
"The data structure associated with each semaphore in the set shall not be initialized."
To:
"The data structure associated with each semaphore in the set need not be initialized."

**Change Number: XSH/TC1/122 [XSH ERN 44]**
On Page: 1290 Line: 40250-40251 Section: seteuid
In the RATIONALE section
Change From:
"[EPERM] The process does not have appropriate privileges and uid does not match the real group ID or the saved set-group-ID."