



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*.

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 "socketmark()" 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 socketmark() function was added.

Change Number: XSH/TC1/5 [XSH ERN 109]

On Page: 55 Line: 2267 Section: 2.9.5.2

Add "fdatsync()" into the table of functions after "fcntl()".

Rationale:

This was an omission. Since fdatsync() 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 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 `aio_cbp->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->aiio_offset would be invalid, [PIO]aiocbp->aiio_reqprio is not a valid value[/PIO], or aiocbp->aiio_nbytes is an invalid value."

Rationale:

Detection of an [EINVAL] error for a invalid value of aiocbp->aiio_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:

$\text{norm1}(v) = |v[0]| + |v[1]| + \dots + |v[n-1]|$

where $|x|$ denotes the absolute value of x , n denotes the vector's dimension, and $v[i]$ denotes the i -th component of v ($0 \leq i < n$)

```
#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 files 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:"