Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'posix_typed_mem_open.3p' command

$ man posix_typed_mem_open.3p

POSIX_TYPED_MEM_OPEN(3P) POSIX Programmer's Manual POSIX_TYPED_MEM_OPEN(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

posix_typed_mem_open ? open a typed memory object (ADVANCED REALTIME)

SYNOPSIS

#include <sys/mman.h>

int posix_typed_mem_open(const char *name, int oflag, int tflag);

DESCRIPTION

The posix_typed_mem_open() function shall establish a connection be? tween the typed memory object specified by the string pointed to by name and a file descriptor. It shall create an open file description that refers to the typed memory object and a file descriptor that refers to that open file description. The file descriptor shall be al? located as described in Section 2.14, File Descriptor Allocation and can be used by other functions to refer to that typed memory object. It is unspecified whether the name appears in the file system and is visi? ble to other functions that take pathnames as arguments. The name argu? ment conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash>

character in name is implementation-defined, and that the length limits for the name argument are implementation-defined and need not be the same as the pathname limits {PATH_MAX} and {NAME_MAX}. If name begins with the <slash> character, then processes calling posix_typed_mem_open() with the same value of name shall refer to the same typed memory object. If name does not begin with the <slash> char? acter, the effect is implementation-defined.

Each typed memory object supported in a system shall be identified by a name which specifies not only its associated typed memory pool, but also the path or port by which it is accessed. That is, the same typed memory pool accessed via several different ports shall have several different corresponding names. The binding between names and typed mem? ory objects is established in an implementation-defined manner. Unlike shared memory objects, there is no way within POSIX.1?2008 for a pro? gram to create a typed memory object.

The value of tflag shall determine how the typed memory object behaves when subsequently mapped by calls to mmap(). At most, one of the fol? lowing flags defined in <sys/mman.h> may be specified:

POSIX_TYPED_MEM_ALLOCATE

Allocate on mmap().

POSIX_TYPED_MEM_ALLOCATE_CONTIG

Allocate contiguously on mmap().

POSIX_TYPED_MEM_MAP_ALLOCATABLE

Map on mmap(), without affecting allocatability.

If tflag has the flag POSIX_TYPED_MEM_ALLOCATE specified, any subse? quent call to mmap() using the returned file descriptor shall result in allocation and mapping of typed memory from the specified typed memory pool. The allocated memory may be a contiguous previously unallocated area of the typed memory pool or several non-contiguous previously un? allocated areas (mapped to a contiguous portion of the process address space). If tflag has the flag POSIX_TYPED_MEM_ALLOCATE_CONTIG speci? fied, any subsequent call to mmap() using the returned file descriptor shall result in allocation and mapping of a single contiguous previ?

ously unallocated area of the typed memory pool (also mapped to a con?
tiguous portion of the process address space). If tflag has none of the
flags POSIX_TYPED_MEM_ALLOCATE or POSIX_TYPED_MEM_ALLOCATE_CONTIG spec?
ified, any subsequent call to mmap() using the returned file descriptor
shall map an application-chosen area from the specified typed memory
pool such that this mapped area becomes unavailable for allocation un?
til unmapped by all processes. If tflag has the flag
POSIX_TYPED_MEM_MAP_ALLOCATABLE specified, any subsequent call to
mmap() using the returned file descriptor shall map an application-cho?
sen area from the specified typed memory pool without an effect on the
availability of that area for allocation; that is, mapping such an ob?
ject leaves each byte of the mapped area unallocated if it was unallo?
cated prior to the mapping or allocated if it was allocated prior to
the mapping. Appropriate privileges to specify the
POSIX_TYPED_MEM_MAP_ALLOCATABLE flag are implementation-defined.
If successful, posix_typed_mem_open() shall return a file descriptor
for the typed memory object. The open file description is new, and
therefore the file descriptor shall not share it with any other pro?
cesses. It is unspecified whether the file offset is set. The
FD_CLOEXEC file descriptor flag associated with the new file descriptor
shall be cleared.
The behavior of msync(), ftruncate(), and all file operations other
than mmap(), posix_mem_offset(), posix_typed_mem_get_info(), fstat(),
dup(), dup2(), and close(), is unspecified when passed a file descrip?
tor connected to a typed memory object by this function.
The file status flags of the open file description shall be set accord?
ing to the value of oflag. Applications shall specify exactly one of
the three access mode values described below and defined in the <fc?
ntl.h> header, as the value of oflag.
O_RDONLY Open for read access only.
O_WRONLY Open for write access only.
O_RDWR Open for read or write access.

RETURN VALUE

Upon successful completion, the posix_typed_mem_open() function shall return a non-negative integer representing the file descriptor. Otherwise, it shall return -1 and set errno to indicate the error.

ERRORS

The posix_typed_mem_open() function shall fail if:

EACCES The typed memory object exists and the permissions specified by oflag are denied.

EINTR The posix_typed_mem_open() operation was interrupted by a signal.

EINVAL The flags specified in tflag are invalid (more than one of POSIX_TYPED_MEM_ALLOCATE, POSIX_TYPED_MEM_ALLOCATE_CONTIG, or POSIX_TYPED_MEM_MAP_ALLOCATABLE is specified).

EMFILE All file descriptors available to the process are currently open.

ENFILE Too many file descriptors are currently open in the system.

ENOENT The named typed memory object does not exist.

EPERM The caller lacks appropriate privileges to specify the POSIX_TYPED_MEM_MAP_ALLOCATABLE flag in the tflag argument.

The posix_typed_mem_open() function may fail if:

ENAMETOOLONG

The length of the name argument exceeds {_POSIX_PATH_MAX} on systems that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI systems, or has a pathname component that is longer than {_POSIX_NAME_MAX} on systems that do not support the XSI option or longer than {_XOPEN_NAME_MAX} on XSI systems.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.14, File Descriptor Allocation, close(), dup(), exec, fc?ntl(), fstat(), ftruncate(), mmap(), msync(), posix_mem_offset(), posix_typed_mem_get_info(), umask()

The Base Definitions volume of POSIX.1?2017, <fcntl.h>, <sys_mman.h>

COPYRIGHT