NAME

shm_map, **shm_unmap** **-** map shared memory objects into the kernel's address space

SYNOPSIS

#include <sys/types.h>
#include <sys/mman.h>

int
shm_map(*struct file *fp*, *size_t size*, *off_t offset*, *void **memp*);

int
shm_unmap(*struct file *fp*, *void *mem*, *size_t size*);

DESCRIPTION

The **shm_map**() and **shm_unmap**() functions provide an API for mapping shared memory objects into the kernel. Shared memory objects are created by shm_open(2). These objects can then be passed into the kernel via file descriptors.

A shared memory object cannot be shrunk while it is mapped into the kernel. This is to avoid invalidating any pages that may be wired into the kernel's address space. Shared memory objects can still be grown while mapped into the kernel.

To simplify the accounting needed to enforce the above requirement, callers of this API are required to unmap the entire region mapped by **shm_map**() when calling **shm_unmap**(). Unmapping only a portion of the region is not permitted.

The **shm_map**() function locates the shared memory object associated with the open file *fp*. It maps the region of that object described by *offset* and *size* into the kernel's address space. If it succeeds, **memp* will be set to the start of the mapping. All pages for the range will be wired into memory upon successful return.

The **shm_unmap**() function unmaps a region previously mapped by **shm_map**(). The *mem* argument should match the value previously returned in **memp*, and the *size* argument should match the value passed to **shm_map**().

Note that **shm_map**() will not hold an extra reference on the open file *fp* for the lifetime of the mapping. Instead, the calling code is required to do this if it wishes to use **shm_unmap**() on the region in the future.

RETURN VALUES

The **shm_map**() and **shm_unmap**() functions return zero on success or an error on failure.

EXAMPLES

The following function accepts a file descriptor for a shared memory object. It maps the first sixteen kilobytes of the object into the kernel, performs some work on that address, and then unmaps the address before returning.

```
int
shm_example(int fd)
{
        struct file *fp;
        void *mem;
        int error;

        error = fget(curthread, fd, CAP_MMAP, &fp);
        if (error)
                return (error);
        error = shm_map(fp, 16384, 0, &mem);
        if (error) {
                fdrop(fp, curthread);
                return (error);
        }

        /* Do something with ’mem’. */

        error = shm_unmap(fp, mem, 16384);
        fdrop(fp, curthread);
        return (error);
}
```

ERRORS

The **shm_map**() function returns the following errors on failure:

[EINVAL]	The open file *fp* is not a shared memory object.
[EINVAL]	The requested region described by *offset* and *size* extends beyond the end of the shared memory object.
[ENOMEM]	Insufficient address space was available.

[EACCES]	The shared memory object could not be mapped due to a protection error.
[EINVAL]	The shared memory object could not be mapped due to some other VM error.

The **shm_unmap**() function returns the following errors on failure:

[EINVAL]	The open file *fp* is not a shared memory object.
[EINVAL]	The address range described by *mem* and *size* is not a valid address range.
[EINVAL]	The address range described by *mem* and *size* is not backed by the shared memory object associated with the open file *fp*, or the address range does not cover the entire mapping of the object.

SEE ALSO

shm_open(2)

HISTORY

This API was first introduced in FreeBSD 10.0.