NAME

mem, **kmem** - memory files

SYNOPSIS

device mem

DESCRIPTION

The special file */dev/mem* is an interface to the physical memory of the computer. Byte offsets in this file are interpreted as physical memory addresses. Reading and writing this file is equivalent to reading and writing memory itself. Only offsets within the bounds of */dev/mem* are allowed.

Kernel virtual memory is accessed through the interface */dev/kmem* in the same manner as */dev/mem*. Only kernel virtual addresses that are currently mapped to memory are allowed.

On ISA the I/O memory space begins at physical address 0x000a0000 and runs to 0x00100000. The per-process data size for the current process is UPAGES long, and ends at virtual address 0xf0000000.

IOCTL INTERFACE

Address Properties

The MEM_EXTRACT_PADDR ioctl can be used to look up the physical address and NUMA domain of a given virtual address in the calling process' address space. The request is described by

```
struct mem_extract {
        uint64_t  me_vaddr;       /* input */
        uint64_t  me_paddr;       /* output */
        int               me_domain;      /* output */
        int               me_state; /* output */
};
```

The ioctl returns an error if the address is not valid. The information returned by MEM_EXTRACT_PADDR may be out of date by the time that the ioctl call returns. Specifically, concurrent system calls, page faults, or system page reclamation activity may have unmapped the virtual page or replaced the backing physical page before the ioctl call returns. Wired pages, e.g., those locked by mlock(2), will not be reclaimed by the system.

The *me_state* field provides information about the state of the virtual page:

ME_STATE_INVALID
    The virtual address is invalid.

ME_STATE_VALID

The virtual address is valid but is not mapped at the time of the ioctl call.

ME_STATE_MAPPED

The virtual address corresponds to a physical page mapping, and the *me_paddr* and *me_domain* fields are valid.

Memory Ranges

Several architectures allow attributes to be associated with ranges of physical memory. These attributes can be manipulated via **ioctl**() calls performed on */dev/mem*. Declarations and data types are to be found in <*sys/memrange.h*>.

The specific attributes, and number of programmable ranges may vary between architectures. The full set of supported attributes is:

MDF_UNCACHEABLE

The region is not cached.

MDF_WRITECOMBINE

Writes to the region may be combined or performed out of order.

MDF_WRITETHROUGH

Writes to the region are committed synchronously.

MDF_WRITEBACK

Writes to the region are committed asynchronously.

MDF_WRITEPROTECT

The region cannot be written to.

Memory ranges are described by

```
struct mem_range_desc {
        uint64_t  mr_base; /* physical base address */
        uint64_t  mr_len;          /* physical length of region */
        int                mr_flags; /* attributes of region */
        char               mr_owner[8];
};
```

In addition to the region attributes listed above, the following flags may also be set in the *mr_flags* field:

MDF_FIXBASE
: The region's base address cannot be changed.

MDF_FIXLEN
: The region's length cannot be changed.

MDF_FIRMWARE
: The region is believed to have been established by the system firmware.

MDF_ACTIVE
: The region is currently active.

MDF_BOGUS
: We believe the region to be invalid or otherwise erroneous.

MDF_FIXACTIVE
: The region cannot be disabled.

MDF_BUSY
: The region is currently owned by another process and may not be altered.

Operations are performed using

```
struct mem_range_op {
        struct mem_range_desc    *mo_desc;
        int                      mo_arg[2];
};
```

The MEMRANGE_GET ioctl is used to retrieve current memory range attributes. If *mo_arg[0]* is set to 0, it will be updated with the total number of memory range descriptors. If greater than 0, the array at *mo_desc* will be filled with a corresponding number of descriptor structures, or the maximum, whichever is less.

The MEMRANGE_SET ioctl is used to add, alter and remove memory range attributes. A range with the MDF_FIXACTIVE flag may not be removed; a range with the MDF_BUSY flag may not be removed or updated.

mo_arg[0] should be set to MEMRANGE_SET_UPDATE to update an existing or establish a new range, or to MEMRANGE_SET_REMOVE to remove a range.

Live Kernel Dumps

The MEM_KERNELDUMP ioctl will initiate a kernel dump against the running system, the contents of which will be written to a process-owned file descriptor. The resulting dump output will be in minidump format. The request is described by

```
struct mem_livedump_arg {
        int       fd;             /* input */
        int       flags           /* input */
        uint8_t   compression     /* input */
};
```

The *fd* field is used to pass the file descriptor.

The *flags* field is currently unused and must be set to zero.

The *compression* field can be used to specify the desired compression to be applied to the dump output. The supported values are defined in <*sys/kerneldump.h*>; that is, KERNELDUMP_COMP_NONE, KERNELDUMP_COMP_GZIP, or KERNELDUMP_COMP_ZSTD.

Kernel dumps taken against the running system may have inconsistent kernel data structures due to allocation, deallocation, or modification of memory concurrent to the dump procedure. Thus, the resulting core dump is not guaranteed to be usable. A system under load is more likely to produce an inconsistent result. Despite this, live kernel dumps can be useful for offline debugging of certain types of kernel bugs, such as deadlocks, or in inspecting a particular part of the system's state.

RETURN VALUES

MEM_EXTRACT_PADDR

The MEM_EXTRACT_PADDR ioctl always returns a value of zero.

MEMRANGE_GET/MEMRANGE_SET

[EOPNOTSUPP] Memory range operations are not supported on this architecture.

[ENXIO] No memory range descriptors are available (e.g., firmware has not enabled any).

[EINVAL] The memory range supplied as an argument is invalid or overlaps another range in a fashion not supported by this architecture.

[EBUSY] An attempt to remove or update a range failed because the range is busy.

[ENOSPC] An attempt to create a new range failed due to a shortage of hardware resources (e.g., descriptor slots).

[ENOENT] An attempt to remove a range failed because no range matches the descriptor base/length supplied.

[EPERM] An attempt to remove a range failed because the range is permanently enabled.

MEM_KERNELDUMP

[EOPNOTSUPP] Kernel minidumps are not supported on this architecture.

[EPERM] An attempt to begin the kernel dump failed because the calling thread lacks the

[EBADF] The supplied file descriptor was invalid, or does not have write permission.

[EBUSY] An attempt to begin the kernel dump failed because one is already in progress.

[EINVAL] An invalid or unsupported value was specified in *flags*.

[EINVAL] An invalid or unsupported compression type was specified.
PRIV_KMEM_READ privilege.

FILES

/dev/mem
/dev/kmem

SEE ALSO

kvm(3), memcontrol(8)

HISTORY

The **mem** and **kmem** files appeared in Version 6 AT&T UNIX. The ioctl interface for memory range attributes was added in FreeBSD 3.2.

BUGS

Busy range attributes are not yet managed correctly.

This device is required for all users of kvm(3) to operate.