NAME

mmap – check virtual mapping for memory mapped device

SYNOPSIS

#include <sys/types.h>
#include <sys/cred.h>
#include <sys/mman.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

int *prefix***mmap(dev_t** *dev,* **off_t** *off,* **int** *prot***);**

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

ARGUMENTS

dev Device whose memory is to be mapped.

off Offset within device memory at which mapping begins.

prot A bit field that specifies the protections this page of memory will receive. Possible settings are:

PROT_READ	Read access will be granted.
PROT_WRITE	Write access will be granted.
PROT_EXEC	Execute access will be granted.
PROT_USER	User-level access will be granted.
PROT_ALL	All access will be granted.

DESCRIPTION

The **mmap**(9E) entry point is a required entry point for character drivers supporting memory-mapped devices. A memory mapped device has memory that can be mapped into a process's address space. The **mmap**(2) system call, when applied to a character special file, allows this device memory to be mapped into user space for direct access by the user application.

The **mmap**(9E) entry point is called as a result of an **mmap**(2) system call, and also as a result of a page fault. **mmap**(9E) is called to translate the offset *off* in device memory to the corresponding physical page frame number.

The **mmap**(9E) entry point checks if the offset *off* is within the range of pages exported by the device. For example, a device that has 512 bytes of memory that can be mapped into user space should not support offsets greater than 512. If the offset does not exist, then **-1** is returned. If the offset does exist, **mmap**(9E) returns the value returned by **hat_getkpfnum**(9F) for the physical page in device memory containing the offset *off*.

hat_getkpfnum(9F) accepts a kernel virtual address as an argument. A kernel virtual address can be obtained by calling **ddi_map_regs**(9F) in the driver's **attach**(9E) routine. The corresponding **ddi_unmap_regs**(9F) call can be made in the driver's **detach**(9E) routine. Refer to the **EXAMPLES** section below for more information.

mmap(9E) should only be supported for memory-mapped devices. See the **segmap**(9E) and **ddi_mapdev**(9F) reference pages for further information on memory-mapped device drivers.

RETURN VALUES

If the protection and offset are valid for the device, the driver should return the value returned by **hat_getkpfnum**(9F), for the page at offset *off* in the device's memory. If not, **-1** should be returned.

EXAMPLES

The following is an example of the **mmap**(9E) entry point. If offset *off* is valid, **hat_getkpfnum**(9F) is called to obtain the page frame number corresponding to this offset in the device's memory. In this example, **xsp→regp→csr** is a kernel virtual address which maps to device memory. **ddi_map_regs**(9F) can be used to obtain this address. For example, **ddi_map_regs**(9F) can be called in the driver's **attach**(9E) routine. The resulting kernel virtual address is stored in the **xxstate** structure (see **ddi_soft_state**(9F)), which is accessible from the driver's **mmap**(9E) entry point. The corresponding **ddi_unmap_regs**(9F) call can be made in the driver's **detach**(9E) routine.

```
struct reg {
        char    csr;
        char    data;
};

struct xxstate {
        . . .
        struct reg        *regp
        . . .
};

struct xxstate *xsp;
. . .

static int
xxmmap(dev_t dev, off_t off, int prot)
{
      int           instance;
      struct xxstate *xsp;

      /* No write access */
      if (prot & PROT_WRITE)
            return (-1);

      instance = getminor(dev);
      xsp = ddi_get_soft_state(statep, instance);
      if (xsp == NULL)
            return (-1);

      /* check for a valid offset */
      if ( off is invalid )
            return (-1);
      return (hat_getkpfnum (xsp->regp->csr + off));
```

```
}
```

SEE ALSO

mmap(2), **attach**(9E), **ddi_map_regs**(9F), **ddi_unmap_regs**(9F), **detach**(9E), **segmap**(9E), **ddi_btop**(9F), **ddi_get_soft_state**(9F), **ddi_mapdev**(9F), **getminor**(9F), **hat_getkpfnum**(9F)

Writing Device Drivers

NOTES

For some devices, mapping device memory in the driver's **attach**(9E) routine and unmapping device memory in the driver's **detach**(9E) routine is a sizeable drain on system resources. This is especially true for devices with a large amount of physical address space.

One alternative is to create a mapping for only the first page of device memory in **attach**(9E). If the device memory is contiguous, a kernel page frame number may be obtained by calling **hat_getkpfnum**(9F) with the kernel virtual address of the first page of device memory and adding the desired page offset to the result. The page offset may be obtained by converting the byte offset *off* to pages (see **ddi_btop**(9F)).

Another alternative is to call **ddi_map_regs**(9F) and **ddi_unmap_regs**(9F) in **mmap**. These function calls would bracket the call to **hat_getkpfnum**(9F).

However, note that the above alternatives may not work in all cases. The existence of intermediate nexus devices with memory management unit translation resources which are not locked down may cause unexpected and undefined behavior.