NAME

mapdev_access – device mapping access entry point

SYNOPSIS

#include <sys/sunddi.h>

int *prefix* **mapdev_access(ddi_mapdev_handle_t** *handle,* **void** **devprivate,* **off_t** *offset***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

handle An opaque pointer to a device mapping.

devprivate Driver private mapping data from **ddi_mapdev**(9F).

offset The offset within device memory at which the access occurred.

DESCRIPTION

mapdev_access( ) is called when an access is made to a mapping that has either been newly created with **ddi_mapdev**(9F) or that has been enabled with a call to **ddi_mapdev_intercept**(9F).

mapdev_access( ) is passed the *handle* of the mapped object on which an access has occurred. This handle uniquely identifies the mapping and is used as an argument to **ddi_mapdev_intercept**(9F) or **ddi_mapdev_nointercept**(9F) to control whether or not future accesses to the mapping will cause **mapdev_access( )** to be called. In general, **mapdev_access( )** should call **ddi_mapdev_intercept( )** on the mapping that is currently in use and then call **ddi_mapdev_nointercept( )** on the mapping that generated this call to **mapdev_access( ).** This will ensure that a call to **mapdev_access( )** will be generated for the current mapping next time it is accessed.

mapdev_access( ) must at least call **ddi_mapdev_nointercept( )** with *offset* passed in in order for the access to succeed. A request to allow accesses affects the entire page containing the *offset.*

Accesses to portions of mappings that have been disabled by a call to **ddi_mapdev_nointercept( )** will not generate a call to **mapdev_access( ).** A subsequent call to **ddi_mapdev_intercept( )** will enable **mapdev_access( )** to be called again.

A non-zero return value from **mapdev_access( )** will cause the corresponding operation to fail. The failure may result in a **SIGSEGV** or **SIGBUS** signal being delivered to the process.

RETURN VALUES

mapdev_access( ) should return **0** on success, **-1** if there was a hardware error, or the return value from **ddi_mapdev_intercept( )** or **ddi_mapdev_nointercept( )**.

CONTEXT

This function is called from user context only.

EXAMPLES

The following shows an example of managing a device context that is one page in length.

```
ddi_mapdev_handle_t cur_hdl;

static int
xxmapdev_access(ddi_mapdev_handle_t handle, void *devprivate,
   off_t offset)
{
        int     err;

        /* enable calls to mapdev_access for the current mapping */
        if (cur_hdl != NULL) {
                if ((err = ddi_mapdev_intercept(cur_hdl, off, 0)) != 0)
                        return (err);
        }

        /* Switch device context - device dependent*/
        ...

        /* Make handle the new current mapping */
        cur_hdl = handle;

        /*
         * Disable callbacks and complete the access for the
         * mapping that generated this callback.
         */

        return (ddi_mapdev_nointercept(handle, off, 0));
}
```

SEE ALSO

mmap(2), **mapdev_dup**(9E), **mapdev_free**(9E), **segmap**(9E), **ddi_mapdev**(9F), **ddi_mapdev_intercept**(9F), **ddi_mapdev_nointercept**(9F), **ddi_mapdev_ctl**(9S),

Writing Device Drivers