NAME

ddi_segmap, ddi_segmap_setup – Set up a user mapping using seg_dev

SYNOPSIS

#include <sys/conf.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

int ddi_segmap(dev_t *dev*, **off_t** *offset*, **struct as** **asp*, **caddr_t** **addrp*, **off_t** *len*, **u_int** *prot*, **u_int** *maxprot,* **u_int** *flags,* **cred_t** **credp*);

int ddi_segmap_setup(dev_t *dev*, **off_t** *offset*, **struct as** **asp*, **caddr_t** **addrp*, **off_t** *len*, **u_int** *prot*, **u_int** *maxprot*, **u_int** *flags*, **cred_t** **credp*, **ddi_device_acc_attr_t** *dev_acc_attr***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

dev The device whose memory is to be mapped.

offset The offset within device memory at which the mapping begins.

asp An opaque pointer to the user address space into which the device memory should be mapped.

addrp Pointer to the starting address within the user address space to which the device memory should be mapped.

len Length (in bytes) of the memory to be mapped.

prot A bit field that specifies the protections. Some combinations of possible settings are:

- **PROT_READ** Read access is desired.
- **PROT_WRITE** Write access is desired.
- **PROT_EXEC** Execute access is desired.
- **PROT_USER** User-level access is desired (the mapping is being done as a result of a **mmap**(2) system call).
- **PROT_ALL** All access is desired.

maxprot *maxprot* Maximum protection flag possible for attempted mapping (the **PROT_WRITE** bit may be masked out if the user opened the special file read-only). If **(maxprot & prot) != prot** then there is an access violation.

flags Flags indicating type of mapping. Possible values are (other bits may be set):

- **MAP_PRIVATE** Changes are private.
- **MAP_SHARED** Changes should be shared.
- **MAP_FIXED** The user specified an address in **addrp* rather than letting the system pick and address.

credp Pointer to user credential structure.

ddi_segmap_setup( )

dev_acc_attr Pointer to a **ddi_device_acc_attr**(9S) structure which contains the device access attributes to apply to this mapping.

DESCRIPTION

ddi_segmap( ) and **ddi_segmap_setup( )** set up user mappings to device space. When setting up the mapping, the **ddi_segmap**( ) and **ddi_segmap_setup**( ) routines call the **mmap**(9E) entry point to validate the range to be mapped. When a user process accesses the mapping, the drivers **mmap**( ) entry point is again called to retrieve the page frame number that needs to be loaded. The mapping translations for that page are then loaded on behalf of the driver by the DDI framework.

ddi_segmap( ) is typically used as the **segmap**(9E) entry in the **cb_ops**(9S) structure for those devices that do not choose to provide their own **segmap**(9E) entry point. However, some drivers may have their own **segmap**(9E) entry point to do some initial processing on the parameters
and then call **ddi_segmap( )** to establish the default memory mapping.

ddi_segmap_setup( ) is used in the drivers **segmap( )** entry point to set up the mapping and assign device access attributes to that mapping. See **ddi_device_acc_attr**(9S) for details what device access attributes are available.

ddi_segmap_setup( ) cannot be used directly in the **cb_ops**(9S) structure and requires a driver to have a **segmap( )** entry point.

RETURN VALUES

ddi_segmap( ) and **ddi_segmap_setup( )** return:

0 on success.

non-zero on failure. In particular, they return **ENXIO** if the range to be mapped is invalid.

CONTEXT

ddi_segmap( ) and **ddi_segmap_setup( )** can be called from user or kernel context only.

SEE ALSO

mmap(2), **mmap**(9E), **segmap**(9E), **ddi_mapdev**(9F), **cb_ops**(9S), **ddi_device_acc_attr**(9S)

Writing Device Drivers

NOTES

If driver notification of user accesses to the mappings is required, the driver should use **ddi_mapdev**(9F) instead.