NAME

open – gain access to a device

SYNOPSIS

Block and Character

```
#include <sys/types.h>
#include <sys/file.h>
#include <sys/errno.h>
#include <sys/open.h>
#include <sys/cred.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
```

int *prefix***open(dev_t** **devp,* **int** *flag,* **int** *otyp,* **cred_t** **cred_p***);**

STREAMS

```
#include <sys/file.h>
#include <sys/stream.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
```

int *prefix***open(queue_t** **q,* **dev_t** **devp,* **int** *oflag,* **int** *sflag,* **cred_t** **cred_p***);**

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI). This entry point is **required**, but it can be **nulldev**(9F).

ARGUMENTS

Block and Character

devp Pointer to a device number.

flag A bit field passed from the user program **open**(2) system call that instructs the driver on how to open the file. Valid settings are:

- **FEXCL** Open the device with exclusive access; fail all other attempts to open the device.
- **FNDELAY** Open the device and return immediately (do not block the open even if something is wrong).
- **FREAD** Open the device with read-only permission (if ORed with **FWRITE**, then allow both read and write access)
- **FWRITE** Open a device with write-only permission (if ORed with **FREAD**, then allow both read and write access)

otyp Parameter supplied so that the driver can determine how many times a device was opened and for what reasons.
For **OTYP_BLK** and **OTYP_CHR**, the **open( )** routine may be called many times, but the **close**(9E) routine is called only when the last reference to a device is removed. If the device is accessed through file descriptors, this is by a call to **close**(2) or **exit**(2). If the device is accessed through memory mapping, this is by a call to **munmap**(2) or **exit**(2).
For **OTYP_LYR**, there is exactly one **close**(9E) for each **open( )** called. This permits software drivers to exist above hardware drivers and removes any ambiguity from the hardware driver regarding how a device is used.

OTYP_BLK Open occurred through block interface for the device

OTYP_CHR Open occurred through the raw/character interface for the device

OTYP_LYR Open a layered process. This flag is used when one driver calls another driver's **open** or **close** (9E) routine. The calling driver will make sure that there is one layered close for each layered open. This flag applies to both block and character devices.

cred_p Pointer to the user credential structure.

STREAMS

q A pointer to the read **queue**.

devp Pointer to a device number. For STREAMS modules, *devp* always points to the device number associated with the driver at the end (tail) of the stream.

oflag Valid *oflag* values are **FEXCL**, **FNDELAY**, **FREAD**, and **FWRITEL**, the same as those listed above for *flag.* For STREAMS modules, *oflag* is always set to **0**.

sflag Valid values are as follows:

CLONEOPEN Indicates that the **open** routine is called through the clone driver. The driver should return a unique device number.

MODOPEN Modules should be called with *sflag* set to this value. Modules should return an error if they are called with *sflag* set to a different value. Drivers should return an error if they are called with *sflag* set to this value.

0 Indicates a driver is opened directly, without calling the clone driver.

cred_p Pointer to the user credential structure.

DESCRIPTION

The driver's **open( )** routine is called by the kernel during an **open**(2) or a **mount**(2) on the special file for the device. The routine should verify that the minor number component of ∗*devp* is valid, that the type of access requested by *otyp* and *flag* is appropriate for the device, and, if required, check permissions using the user credentials pointed to by *cred_p*.

The **open( )** routine is passed a pointer to a device number so that the driver can change the minor number. This allows drivers to dynamically create minor instances of the device. An example of this might be a pseudo-terminal driver that creates a new pseudo-terminal whenever it is opened. A driver that chooses the minor number dynamically, normally creates only one minor device node in **attach**(9E) with **ddi_create_minor_node**(9F), then changes the minor number component of ∗*devp* using **makedevice**(9F) and **getmajor**(9F). The driver needs to keep track of available minor numbers internally.

∗devp = makedevice(getmajor(∗devp), new_minor);

RETURN VALUES

The **open( )** routine should return 0 for success, or the appropriate error number.

SEE ALSO

exit(2), **mmap**(2), **mount**(2), **munmap**(2), **open**(2), **intro**(9E), **attach**(9E), **close**(9E), **ddi_create_minor_node**(9F), **getmajor**(9F), **getminor**(9F), **makedevice**(9F)

Writing Device Drivers

STREAMS Programming Guide

WARNINGS

Do not attempt to change the major number.