NAME

ioctl – control a character device

SYNOPSIS

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

int *prefix***ioctl(dev_t** *dev,* **int** *cmd,* **int** *arg,* **int** *mode,* **cred_t** **cred_p,* **int** **rval_p***);**

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI). This entry point is **optional**.

ARGUMENTS

dev Device number.

cmd Command argument the driver **ioctl** routine interprets as the operation to be performed.

arg Passes parameters between a user program and the driver. When used with terminals, the argument is the address of a user program structure containing driver or hardware settings. Alternatively, the argument may be an integer that has meaning only to the driver. The interpretation of the argument is driver dependent and usually depends on the command type; the kernel does not interpret the argument.

mode Contains values set when the device was opened. Use of this mode is optional. However, the driver may use it to determine if the device was opened for reading or writing. The driver can make this determination by checking the **FREAD** or **FWRITE** flags. See the *flag* argument description of the **open( )** routine for further values for the **ioctl** routine's *mode* argument.

In some circumstances, *mode* is used to provide address space information about the *arg* argument. See below.

cred_p Pointer to the user credential structure.

rval_p Pointer to return value for calling process. The driver may elect to set the value which is valid only if the **ioctl( )** succeeds.

DESCRIPTION

ioctl( ) provides character-access drivers with an alternate entry point that can be used for almost any operation other than a simple transfer of characters in and out of buffers. Most often, **ioctl( )** is used to control device hardware parameters and establish the protocol used by the driver in processing data.

The kernel determines that this is a character device, and looks up the entry point routines in **cb_ops** (9S). The kernel then packages the user request and arguments as integers and passes them to the driver's **ioctl( )** routine. The kernel itself does no processing of the passed command, so it is up to the user program and the driver to agree on what the arguments mean.

I/O control commands are used to implement the terminal settings passed from **ttymon**(1M) and **stty**(1), to format disk devices, to implement a trace driver for debugging, and to clean up character queues. Since the kernel does not interpret the command type that defines the operation, a driver is free to define its own commands.

Drivers that use an **ioctl( )** routine typically have a command to ''read'' the current **ioctl( )** settings, and at least one other that sets new settings. Drivers can use the mode argument to determine if the device unit was opened for reading or writing, if necessary, by checking the **FREAD** or **FWRITE** setting.

If the third argument, *arg*, is a pointer to a user buffer, the driver can call the **copyin**(9F) and **copyout**(9F) functions to transfer data between kernel and user space.

Other kernel subsystems may need to call into the drivers **ioctl** routine. Drivers that intend to allow their **ioctl( )** routine to be used in this way should publish the **ddi-kernel-ioctl** property on the associated devinfo node(s).

When the **ddi-kernel-ioctl** property is present, the *mode* argument is used to pass address space information about *arg* through to the driver. If the driver expects *arg* to contain a buffer address, and the **FKIOCTL** flag is set in *mode*, then the driver should assume that it is being handed a kernel buffer address. Otherwise, *arg* may be the address of a buffer from a user program. The driver can use **ddi_copyin**(9F) and **ddi_copyout**(9F) perform the correct type of copy operation for either kernel or user address spaces. See the example on **ddi_copyout**(9F).

To implement I/O control commands for a driver the following two steps are required:

1. Define the I/O control command names and the associated value in the driver's header and comment the commands.
2. Code the **ioctl** routine in the driver that defines the functionality for each I/O control command name that is in the header.

The **ioctl** routine is coded with instructions on the proper action to take for each command. It is commonly a **switch** statement, with each **case** definition corresponding to an **ioctl** name to identify the action that should be taken. However, the command passed to the driver by the user process is an integer value associated with the command name in the header.

RETURN VALUES

ioctl( ) should return **0** on success, or the appropriate error number. The driver may also set the value returned to the calling process through *rval_p*.

SEE ALSO

stty(1), **ttymon**(1M), **dkio**(7I), **fbio**(7I), **termio**(7I), **open**(9E), **put**(9E), **srv**(9E), **copyin**(9F), **copyout**(9F), **ddi_copyin**(9F), **ddi_copyout**(9F) **cb_ops**(9S)

Writing Device Drivers

WARNINGS

Non-STREAMS driver **ioctl( )** routines must make sure that user data is copied into or out of the kernel address space explicitly using **copyin**(9F), **copyout**(9F), **ddi_copyin**(9F), or **ddi_copyout**(9F), as appropriate.

It is a severe error to simply dereference pointers to the user address space, even when in user context.

Failure to use the appropriate copying routines can result in panics under load on some platforms, and reproducible panics on others.

NOTES

STREAMS drivers do not have **ioctl** routines. The stream head converts I/O control commands to **M_IOCTL** messages, which are handled by the driver's **put**(9E) or **srv**(9E) routine.