NAME

copyout – copy data from a driver to a user program

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ddi.h>

int copyout(caddr_t driverbuf, caddr_t userbuf, size_t cn);
```

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

ARGUMENTS

driverbuf	Source address in the driver from which the data is transferred.
userbuf	Destination address in the user program to which the data is transferred.
cn	Number of bytes moved.

DESCRIPTION

copyout( ) copies data from driver buffers to user data space.

Addresses that are word-aligned are moved most efficiently. However, the driver developer is not obligated to ensure alignment. This function automatically finds the most efficient move algorithm according to address alignment.

RETURN VALUES

Under normal conditions a **0** is returned to indicate a successful copy. Otherwise, a **-1** is returned if one of the following occurs:

- paging fault; the driver tried to access a page of memory for which it did not have read or write access
- invalid user address, such as a user area or stack area
- invalid address that would have resulted in data being copied into the user block

If a **-1** is returned to the caller, driver entry point routines should return **EFAULT**.

CONTEXT

copyout( ) can be called from user context only.

EXAMPLES

A driver **ioctl**(9E) routine (line 9) can be used to get or set device attributes or registers. In the **XX_GETREGS** condition (line 17), the driver copies the current device register values to a user data area (line 18). If the specified argument contains an invalid address, an error code is returned.

```
 1 struct device {                      /* layout of physical device registers */
 2       int       control;             /* physical device control word */
 3       int       status;              /* physical device status word */
 4       short     recv_char;           /* receive character from device */
 5       short     xmit_char;           /* transmit character to device */
 6 }; /* end device */
 7
 8 extern struct device xx_addr[];       /* phys. device regs. location */
    . . .
 9 xx_ioctl(dev, cmd, arg, mode, cred_p, rval_p)
10       dev_t     dev;
```

```
11      int         cmd, arg;
12              ...
13  {
14      register struct device *rp = &xx_addr[getminor(dev) >> 4];
15      switch (cmd)  {
16
17          case XX_GETREGS:                    /* copy device regs. to user program */
18              if (copyout((caddr_t)rp, (caddr_t)arg, sizeof(struct device)))
19                  return(EFAULT);
21              break;
```

SEE ALSO

bcopy(9F), **copyin**(9F), **ddi_copyin**(9F), **ddi_copyout**(9F), **uiomove**(9F).

Writing Device Drivers

NOTES

Driver writers who intend to support layered ioctls in their **ioctl**(9E) routines should use **ddi_copyout**(9F) instead.

Driver defined locks should not be held across calls to this function.