NAME

tty ioctl − ioctls for terminals and serial lines

SYNOPSIS

#include <termios.h>

int ioctl(int *fd***, int** *cmd***, ...);**

DESCRIPTION

The **ioctl**() call for terminals and serial ports accepts many possible command arguments. Most require a third argument, of varying type, here called *argp* or *arg*.

Use of *ioctl* makes for non-portable programs. Use the POSIX interface described in **termios**(3) whenever possible.

Get and Set Terminal Attributes

TCGETS **struct termios ***_argp_

Equivalent to *tcgetattr(fd, argp)*.
Get the current serial port settings.

TCSETS **const struct termios ***_argp_

Equivalent to *tcsetattr(fd, TCSANOW, argp)*.
Set the current serial port settings.

TCSETSW **const struct termios ***_argp_

Equivalent to *tcsetattr(fd, TCSADRAIN, argp)*.
Allow the output buffer to drain, and set the current serial port settings.

TCSETSF **const struct termios ***_argp_

Equivalent to *tcsetattr(fd, TCSAFLUSH, argp)*.
Allow the output buffer to drain, discard pending input, and set the current serial port settings.

The following four ioctls are just like **TCGETS**, **TCSETS**, **TCSETSW**, **TCSETSF**, except that they take a *struct termio* * instead of a *struct termios* *.

TCGETA **struct termio ***_argp_

TCSETA **const struct termio ***_argp_

TCSETAW **const struct termio ***_argp_

TCSETAF **const struct termio ***_argp_

Locking the termios structure

The *termios* structure of a terminal can be locked. The lock is itself a *termios* structure, with non-zero bits or fields indicating a locked value.

TIOCGLCKTRMIOS **struct termios ***_argp_

Gets the locking status of the *termios* structure of the terminal.

TIOCSLCKTRMIOS **const struct termios ***_argp_

Sets the locking status of the *termios* structure of the terminal. Only root (more precisely: a process with the **CAP_SYS_ADMIN** capability) can do this.

Get and Set Window Size

Window sizes are kept in the kernel, but not used by the kernel (except in the case of virtual consoles, where the kernel will update the window size when the size of the virtual console changes, for example, by loading a new font).

The following constants and structure are defined in *<sys/ioctl.h>*.

TIOCGWINSZ **struct winsize ***_argp_

Get window size.

TIOCSWINSZ **const struct winsize ****argp*
Set window size.

The struct used by these ioctls is defined as

```
struct winsize {
    unsigned short ws_row;
    unsigned short ws_col;
    unsigned short ws_xpixel;   /* unused */
    unsigned short ws_ypixel;   /* unused */
};
```

When the window size changes, a **SIGWINCH** signal is sent to the foreground process group.

Sending a Break

TCSBRK **int** *arg*
Equivalent to *tcsendbreak(fd, arg)*.
If the terminal is using asynchronous serial data transmission, and *arg* is zero, then send a break (a stream of zero bits) for between 0.25 and 0.5 seconds. If the terminal is not using asynchronous serial data transmission, then either a break is sent, or the function returns without doing anything. When *arg* is non-zero, nobody knows what will happen.

(SVr4, UnixWare, Solaris, Linux treat *tcsendbreak(fd,arg)* with non-zero *arg* like *tcdrain(fd)*. SunOS treats *arg* as a multiplier, and sends a stream of bits *arg* times as long as done for zero *arg*. DG/UX and AIX treat *arg* (when non-zero) as a time interval measured in milliseconds. HP-UX ignores *arg*.)

TCSBRKP **int** *arg*
So-called "POSIX version" of **TCSBRK**. It treats non-zero *arg* as a timeinterval measured in deciseconds, and does nothing when the driver does not support breaks.

TIOCSBRK **void**
Turn break on, that is, start sending zero bits.

TIOCCBRK **void**
Turn break off, that is, stop sending zero bits.

Software flow control

TCXONC **int** *arg*
Equivalent to *tcflow(fd, arg)*.
See **tcflow**(3) for the argument values **TCOOFF**, **TCOON**, **TCIOFF**, **TCION**.

Buffer count and flushing

FIONREAD **int ****argp*
Get the number of bytes in the input buffer.

TIOCINQ **int ****argp*
Same as **FIONREAD**.

TIOCOUTQ **int ****argp*
Get the number of bytes in the output buffer.

TCFLSH **int** *arg*
Equivalent to *tcflush(fd, arg)*.
See **tcflush**(3) for the argument values **TCIFLUSH**, **TCOFLUSH**, **TCIOFLUSH**.

Faking input

TIOCSTI **const char ****argp*
Insert the given byte in the input queue.

Redirecting console output

TIOCCONS void

Redirect output that would have gone to */dev/console* or */dev/tty0* to the given terminal. If that was a pseudo-terminal master, send it to the slave. In Linux before version 2.6.10, anybody can do this as long as the output was not redirected yet; since version 2.6.10, only root (a process with the **CAP_SYS_ADMIN** capability) may do this. If output was redirected already **EBUSY** is returned, but redirection can be stopped by using this ioctl with *fd* pointing at */dev/console* or */dev/tty0*.

Controlling terminal

TIOCSCTTY int *arg*

Make the given terminal the controlling terminal of the calling process. The calling process must be a session leader and not have a controlling terminal already. If this terminal is already the controlling terminal of a different session group then the ioctl fails with **EPERM**, unless the caller is root (more precisely: has the **CAP_SYS_ADMIN** capability) and *arg* equals 1, in which case the terminal is stolen, and all processes that had it as controlling terminal lose it.

TIOCNOTTY void

If the given terminal was the controlling terminal of the calling process, give up this controlling terminal. If the process was session leader, then send **SIGHUP** and **SIGCONT** to the foreground process group and all processes in the current session lose their controlling terminal.

Process group and session ID

**TIOCGPGRP pid_t ***_argp_

When successful, equivalent to **argp = tcgetpgrp(fd)*.
Get the process group ID of the foreground process group on this terminal.

**TIOCSPGRP const pid_t ***_argp_

Equivalent to *tcsetpgrp(fd, *argp)*.
Set the foreground process group ID of this terminal.

**TIOCGSID pid_t ***_argp_

Get the session ID of the given terminal. This will fail with **ENOTTY** in case the terminal is not a master pseudo-terminal and not our controlling terminal. Strange.

Exclusive mode

TIOCEXCL void

Put the terminal into exclusive mode. No further **open**(2) operations on the terminal are permitted. (They will fail with **EBUSY**, except for root, that is, a process with the **CAP_SYS_ADMIN** capability.)

TIOCNXCL void

Disable exclusive mode.

Line discipline

**TIOCGETD int ***_argp_

Get the line discipline of the terminal.

**TIOCSETD const int ***_argp_

Set the line discipline of the terminal.

Pseudo-terminal ioctls

**TIOCPKT const int ***_argp_

Enable (when **argp* is non-zero) or disable packet mode. Can be applied to the master side of a pseudo-terminal only (and will return **ENOTTY** otherwise). In packet mode, each subsequent **read**(2) will return a packet that either contains a single non-zero control byte, or has a single byte containing zero ('\0') followed by data written on the slave side of the pseudo-terminal. If the first byte is not **TIOCPKT_DATA** (0), it is an OR of one or more of the following bits:

TIOCPKT_FLUSHREAD The read queue for the terminal is flushed.

TIOCPKT_FLUSHWRITE The write queue for the terminal is flushed.
TIOCPKT_STOP Output to the terminal is stopped.
TIOCPKT_START Output to the terminal is restarted.
TIOCPKT_DOSTOP The start and stop characters are **^S**/**^Q**.
TIOCPKT_NOSTOP The start and stop characters are not **^S**/**^Q**.

While this mode is in use, the presence of control status information to be read from the master side may be detected by a **select**(2) for exceptional conditions.

This mode is used by **rlogin**(1) and **rlogind**(8) to implement a remote-echoed, locally **^S**/**^Q** flow-controlled remote login.

The BSD ioctls **TIOCSTOP**, **TIOCSTART**, **TIOCUCNTL**, **TIOCREMOTE** have not been implemented under Linux.

Modem control

TIOCMGET **int ****argp*
get the status of modem bits.

TIOCMSET **const int ****argp*
set the status of modem bits.

TIOCMBIC **const int ****argp*
clear the indicated modem bits.

TIOCMBIS **const int ****argp*
set the indicated modem bits.

Bits used by these four ioctls:

TIOCM_LE DSR (data set ready/line enable)
TIOCM_DTR DTR (data terminal ready)
TIOCM_RTS RTS (request to send)
TIOCM_ST Secondary TXD (transmit)
TIOCM_SR Secondary RXD (receive)
TIOCM_CTS CTS (clear to send)
TIOCM_CAR DCD (data carrier detect)
TIOCM_CD see TIOCM_CAR
TIOCM_RNG RNG (ring)
TIOCM_RI see TIOCM_RNG
TIOCM_DSR DSR (data set ready)

Marking a line as local

TIOCGSOFTCAR **int ****argp*
("Get software carrier flag") Get the status of the CLOCAL flag in the c_cflag field of the *termios* structure.

TIOCSSOFTCAR **const int ****argp*
("Set software carrier flag") Set the CLOCAL flag in the *termios* structure when **argp* is non-zero, and clear it otherwise.

If the **CLOCAL** flag for a line is off, the hardware carrier detect (DCD) signal is significant, and an **open**(2) of the corresponding terminal will block until DCD is asserted, unless the **O_NONBLOCK** flag is given. If **CLOCAL** is set, the line behaves as if DCD is always asserted. The software carrier flag is usually turned on for local devices, and is off for lines with modems.

Linux-specific

For the **TIOCLINUX** ioctl, see **console_ioctl**(4).

Kernel debugging

#include <linux/tty.h>

TIOCTTYGSTRUCT **struct tty_struct ****argp*

Get the *tty_struct* corresponding to *fd*.

RETURN VALUE

The **ioctl**() system call returns 0 on success. On error it returns −1 and sets *errno* appropriately.

ERRORS

EINVAL

Invalid command parameter.

ENOIOCTLCMD

Unknown command.

ENOTTY

Inappropriate *fd*.

EPERM

Insufficient permission.

EXAMPLE

Check the condition of DTR on the serial port.

```
#include <termios.h>
#include <fcntl.h>
#include <sys/ioctl.h>

int
main(void)
{
   int fd, serial;

   fd = open("/dev/ttyS0", O_RDONLY);
   ioctl(fd, TIOCMGET, &serial);
   if (serial & TIOCM_DTR)
      puts("TIOCM_DTR is not set");
   else
      puts("TIOCM_DTR is set");
   close(fd);
}
```

SEE ALSO

ioctl(2), **termios**(3), **console_ioctl**(4), **pty**(7)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.