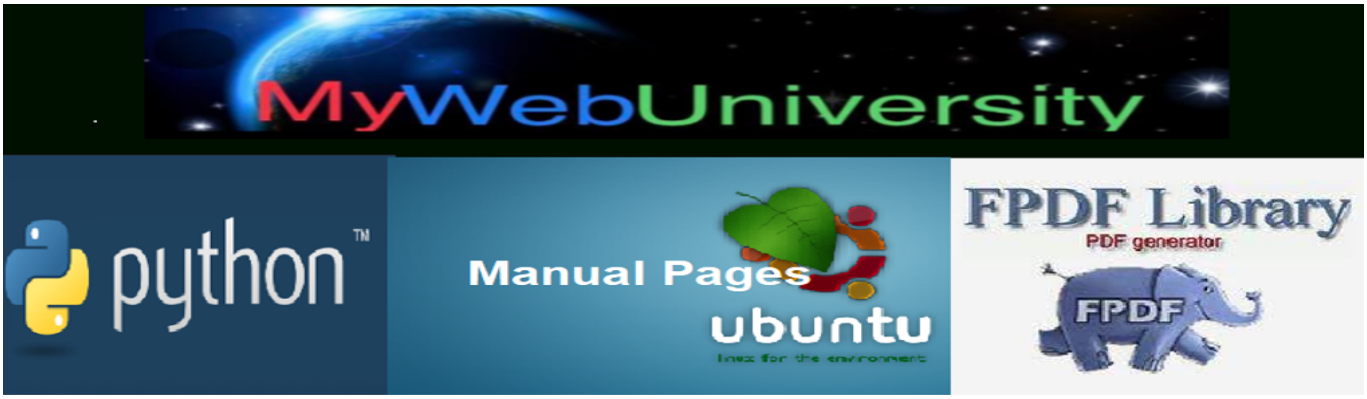

Rocky Enterprise Linux 9.2 Manual Pages on command 'tty_ioctl.4'

$ man tty_ioctl.4

IOCTL_TTY(2) Linux Programmer's Manual IOCTL_TTY(2)

NAME

ioctl_tty - ioctls for terminals and serial lines

SYNOPSIS

#include <termios.h>

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

DESCRIPTION

The ioctl(2) 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 nonportable 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 struc? ture, with nonzero 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 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 vir? tual 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 nonzero, nobody knows what will happen.

(SVr4, UnixWare, Solaris, Linux treat tcsendbreak(fd,arg) with nonzero 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 nonzero) as a time interval measured in milliseconds. HP-UX ignores arg.)

TCSBRKP int arg

So-called "POSIX version" of TCSBRK. It treats nonzero arg as a time interval mea? sured 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 ter?
minal. If that was a pseudoterminal 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 a process with the CAP_SYS_ADMIN capability may do this.
If output was redirected already, then 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 call?
ing process must be a session leader and not have a controlling terminal already.
For this case, arg should be specified as zero.
If this terminal is already the controlling terminal of a different session group,
then the ioctl fails with EPERM, unless the caller 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 fails with the error ENOTTY if the terminal is not a master pseudoterminal and not our controlling terminal. Strange.

Exclusive mode

TIOCEXCL void

Put the terminal into exclusive mode. No further open(2) operations on the termi? nal are permitted. (They fail with EBUSY, except for a process with the CAP_SYS_ADMIN capability.)

TIOCGEXCL int *argp

(since Linux 3.8) If the terminal is currently in exclusive mode, place a nonzero value in the location pointed to by argp; otherwise, place zero in *argp.

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.

Pseudoterminal ioctls

TIOCPKT const int *argp

Enable (when *argp is nonzero) or disable packet mode. Can be applied to the mas? ter side of a pseudoterminal only (and will return ENOTTY otherwise). In packet mode, each subsequent read(2) will return a packet that either contains a single nonzero control byte, or has a single byte containing zero ('\0') followed by data written on the slave side of the pseudoterminal. If the first byte is not TI? OCPKT_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 packet 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 or a

poll(2) for the POLLPRI event.

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

TIOCGPKT const int *argp

(since Linux 3.8) Return the current packet mode setting in the integer pointed to by argp.

TIOCSPTLCK int *argp

Set (if *argp is nonzero) or remove (if *argp is zero) the lock on the pseudotermi? nal slave device. (See also unlockpt(3).)

TIOCGPTLCK int *argp

(since Linux 3.8) Place the current lock state of the pseudoterminal slave device in the location pointed to by argp.

TIOCGPTPEER int flags

(since Linux 4.13) Given a file descriptor in fd that refers to a pseudoterminal master, open (with the given open(2)-style flags) and return a new file descriptor that refers to the peer pseudoterminal slave device. This operation can be per? formed regardless of whether the pathname of the slave device is accessible through the calling process's mount namespace.

Security-conscious programs interacting with namespaces may wish to use this opera? tion rather than open(2) with the pathname returned by ptsname(3), and similar li? brary functions that have insecure APIs. (For example, confusion can occur in some cases using ptsname(3) with a pathname where a devpts filesystem has been mounted in a different mount namespace.)

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.

The following bits are used by the above 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)

TIOCMIWAIT int arg

Wait for any of the 4 modem bits (DCD, RI, DSR, CTS) to change. The bits of inter?

est are specified as a bit mask in arg, by ORing together any of the bit values,

TIOCM_RNG, TIOCM_DSR, TIOCM_CD, and TIOCM_CTS. The caller should use TIOCGICOUNT

to see which bit has changed.

TIOCGICOUNT struct serial_icounter_struct *argp

Get counts of input serial line interrupts (DCD, RI, DSR, CTS). The counts are

written to the serial_icounter_struct structure pointed to by argp.

Note: both 1->0 and 0->1 transitions are counted, except for RI, where only 0->1

transitions are counted.

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 nonzero, and clear it otherwise.

If the CLOCAL flag for a line is off, the hardware carrier detect (DCD) signal is signifi?

cant, and an open(2) of the corresponding terminal will block until DCD is asserted, un?

less 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 ioctl_console(2).

Kernel debugging

#include <linux/tty.h>

TIOCTTYGSTRUCT struct tty_struct *argp

Get the tty_struct corresponding to fd. This command was removed in Linux 2.5.67.

RETURN VALUE

The ioctl(2) system call returns 0 on success. On error, it returns -1 and sets errno ap? propriately.

ERRORS

EINVAL Invalid command parameter.

ENOIOCTLCMD

Unknown command.

ENOTTY Inappropriate fd.

EPERM Insufficient permission.

EXAMPLES

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 set");
    else
        puts("TIOCM_DTR is not set");
    close(fd);
```

```
}
```

SEE ALSO

ldattach(1), ioctl(2), ioctl_console(2), termios(3), pty(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.