man7.org > Linux > man-pages   **Linux/UNIX system programming training**

NAME | SYNOPSIS | DESCRIPTION | RETURN VALUE | ATTRIBUTES | CONFORMING TO |
NOTES | SEE ALSO | COLOPHON

Search online pages

TERMIOS(3)   Linux Programmer's Manual   TERMIOS(3)

NAME   top

       termios, tcgetattr, tcsetattr, tcsendbreak, tcdrain, tcflush, tcflow,
       cfmakeraw, cfgetospeed, cfgetispeed, cfsetispeed, cfsetospeed, cfset‐
       speed - get and set terminal attributes, line control, get and set
       baud rate

SYNOPSIS   top

       #include <termios.h>
       #include <unistd.h>

       int tcgetattr(int *fd*, **struct termios ****termios_p***);**

       int tcsetattr(int *fd*, **int** *optional_actions*,
                     const struct termios **termios_p***);**

       int tcsendbreak(int *fd*, **int** *duration***);**

       int tcdrain(int *fd***);**

       int tcflush(int *fd*, **int** *queue_selector***);**

       int tcflow(int *fd*, **int** *action***);**

       void cfmakeraw(struct termios **termios_p***);**

       speed_t cfgetispeed(const struct termios **termios_p***);**

       speed_t cfgetospeed(const struct termios **termios_p***);**

       int cfsetispeed(struct termios *termios_p, **speed_t** *speed***);**

       int cfsetospeed(struct termios *termios_p, **speed_t** *speed***);**

       int cfsetspeed(struct termios *termios_p, **speed_t** *speed***);**

    Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       cfsetspeed(), cfmakeraw():
           Since glibc 2.19:
               _DEFAULT_SOURCE
           Glibc 2.19 and earlier:
               _BSD_SOURCE

DESCRIPTION   top

>   The termios functions describe a general terminal interface that is
>   provided to control asynchronous communications ports.

The termios structure
>   Many of the functions described here have a *termios_p* argument that
>   is a pointer to a *termios* structure. This structure contains at
>   least the following members:

```
            tcflag_t c_iflag;       /* input modes */
            tcflag_t c_oflag;       /* output modes */
            tcflag_t c_cflag;       /* control modes */
            tcflag_t c_lflag;       /* local modes */
            cc_t     c_cc[NCCS];    /* special characters */
```

>   The values that may be assigned to these fields are described below.
>   In the case of the first four bit-mask fields, the definitions of
>   some of the associated flags that may be set are exposed only if a
>   specific feature test macro (see feature_test_macros(7)) is defined,
>   as noted in brackets ("[]").

>   In the descriptions below, "not in POSIX" means that the value is not
>   specified in POSIX.1-2001, and "XSI" means that the value is speci-
>   fied in POSIX.1-2001 as part of the XSI extension.

>   *c_iflag* flag constants:

>   **IGNBRK** Ignore BREAK condition on input.

>   **BRKINT** If **IGNBRK** is set, a BREAK is ignored. If it is not set but
>   **BRKINT** is set, then a BREAK causes the input and output queues
>   to be flushed, and if the terminal is the controlling terminal
>   of a foreground process group, it will cause a **SIGINT** to be
>   sent to this foreground process group. When neither **IGNBRK**
>   nor **BRKINT** are set, a BREAK reads as a null byte ('\0'),
>   except when **PARMRK** is set, in which case it reads as the
>   sequence \377 \0 \0.

>   **IGNPAR** Ignore framing errors and parity errors.

>   **PARMRK** If this bit is set, input bytes with parity or framing errors
>   are marked when passed to the program. This bit is meaningful
>   only when **INPCK** is set and **IGNPAR** is not set. The way erro-
>   neous bytes are marked is with two preceding bytes, \377 and
>   \0. Thus, the program actually reads three bytes for one
>   erroneous byte received from the terminal. If a valid byte
>   has the value \377, and **ISTRIP** (see below) is not set, the
>   program might confuse it with the prefix that marks a parity
>   error. Therefore, a valid byte \377 is passed to the program
>   as two bytes, \377 \377, in this case.

>   If neither **IGNPAR** nor **PARMRK** is set, read a character with a

parity error or framing error as \0.

INPCK   Enable input parity checking.

ISTRIP Strip off eighth bit.

INLCR   Translate NL to CR on input.

IGNCR   Ignore carriage return on input.

ICRNL   Translate carriage return to newline on input (unless **IGNCR** is set).

IUCLC   (not in POSIX) Map uppercase characters to lowercase on input.

IXON   Enable XON/XOFF flow control on output.

IXANY   (XSI) Typing any character will restart stopped output. (The default is to allow just the START character to restart output.)

IXOFF   Enable XON/XOFF flow control on input.

IMAXBEL
        (not in POSIX) Ring bell when input queue is full. Linux does not implement this bit, and acts as if it is always set.

IUTF8 (since Linux 2.6.4)
        (not in POSIX) Input is UTF8; this allows character-erase to be correctly performed in cooked mode.

c_oflag flag constants:

OPOST   Enable implementation-defined output processing.

OLCUC   (not in POSIX) Map lowercase characters to uppercase on output.

ONLCR   (XSI) Map NL to CR-NL on output.

OCRNL   Map CR to NL on output.

ONOCR   Don't output CR at column 0.

ONLRET Don't output CR.

OFILL   Send fill characters for a delay, rather than using a timed delay.

OFDEL   Fill character is ASCII DEL (0177). If unset, fill character is ASCII NUL ('\0'). (Not implemented on Linux.)

NLDLY   Newline delay mask. Values are **NL0** and **NL1**. [requires **_BSD_SOURCE** or **_SVID_SOURCE** or **_XOPEN_SOURCE**]

CRDLY   Carriage return delay mask. Values are **CR0**, **CR1**, **CR2**, or **CR3**.
        [requires **_BSD_SOURCE** or **_SVID_SOURCE** or **_XOPEN_SOURCE**]

TABDLY Horizontal tab delay mask. Values are **TAB0**, **TAB1**, **TAB2**, **TAB3**
        (or **XTABS**). A value of TAB3, that is, XTABS, expands tabs to
        spaces (with tab stops every eight columns). [requires
        _BSD_SOURCE or **_SVID_SOURCE** or **_XOPEN_SOURCE**]

BSDLY Backspace delay mask. Values are **BS0** or **BS1**. (Has never been
        implemented.) [requires **_BSD_SOURCE** or **_SVID_SOURCE** or
        _XOPEN_SOURCE]

VTDLY Vertical tab delay mask. Values are **VT0** or **VT1**.

FFDLY Form feed delay mask. Values are **FF0** or **FF1**. [requires
        _BSD_SOURCE or **_SVID_SOURCE** or **_XOPEN_SOURCE**]

c_cflag flag constants:

CBAUD (not in POSIX) Baud speed mask (4+1 bits). [requires
        _BSD_SOURCE or **_SVID_SOURCE**]

CBAUDEX
        (not in POSIX) Extra baud speed mask (1 bit), included in
        CBAUD. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

        (POSIX says that the baud speed is stored in the *termios*
        structure without specifying where precisely, and provides
        cfgetispeed() and **cfsetispeed**() for getting at it. Some sys-
        tems use bits selected by **CBAUD** in *c_cflag*, other systems use
        separate fields, for example, *sg_ispeed* and *sg_ospeed*.)

CSIZE Character size mask. Values are **CS5**, **CS6**, **CS7**, or **CS8**.

CSTOPB Set two stop bits, rather than one.

CREAD Enable receiver.

PARENB Enable parity generation on output and parity checking for
        input.

PARODD If set, then parity for input and output is odd; otherwise
        even parity is used.

HUPCL Lower modem control lines after last process closes the device
        (hang up).

CLOCAL Ignore modem control lines.

LOBLK (not in POSIX) Block output from a noncurrent shell layer.
        For use by **shl** (shell layers). (Not implemented on Linux.)

CIBAUD (not in POSIX) Mask for input speeds. The values for the

CIBAUD bits are the same as the values for the **CBAUD** bits, shifted left **IBSHIFT** bits. [requires **_BSD_SOURCE** or **_SVID_SOURCE**] (Not implemented on Linux.)

CMSPAR (not in POSIX) Use "stick" (mark/space) parity (supported on certain serial devices): if **PARODD** is set, the parity bit is always 1; if **PARODD** is not set, then the parity bit is always 0. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

CRTSCTS
         (not in POSIX) Enable RTS/CTS (hardware) flow control. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

c_lflag flag constants:

ISIG   When any of the characters INTR, QUIT, SUSP, or DSUSP are received, generate the corresponding signal.

ICANON Enable canonical mode (described below).

XCASE (not in POSIX; not supported under Linux) If **ICANON** is also set, terminal is uppercase only. Input is converted to lowercase, except for characters preceded by \. On output, uppercase characters are preceded by \ and lowercase characters are converted to uppercase. [requires **_BSD_SOURCE** or **_SVID_SOURCE** or **_XOPEN_SOURCE**]

ECHO   Echo input characters.

ECHOE If **ICANON** is also set, the ERASE character erases the preceding input character, and WERASE erases the preceding word.

ECHOK If **ICANON** is also set, the KILL character erases the current line.

ECHONL If **ICANON** is also set, echo the NL character even if ECHO is not set.

ECHOCTL
         (not in POSIX) If **ECHO** is also set, terminal special characters other than TAB, NL, START, and STOP are echoed as **^X**, where X is the character with ASCII code 0x40 greater than the special character. For example, character 0x08 (BS) is echoed as **^H**. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

ECHOPRT
         (not in POSIX) If **ICANON** and **ECHO** are also set, characters are printed as they are being erased. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

ECHOKE (not in POSIX) If **ICANON** is also set, KILL is echoed by erasing each character on the line, as specified by **ECHOE** and **ECHOPRT**. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

DEFECHO
> (not in POSIX) Echo only when a process is reading. (Not
> implemented on Linux.)

FLUSHO (not in POSIX; not supported under Linux) Output is being
> flushed. This flag is toggled by typing the DISCARD charac-
> ter. [requires **_BSD_SOURCE** or **_SVID_SOURCE**]

NOFLSH Disable flushing the input and output queues when generating
> signals for the INT, QUIT, and SUSP characters.

TOSTOP Send the **SIGTTOU** signal to the process group of a background
> process which tries to write to its controlling terminal.

PENDIN (not in POSIX; not supported under Linux) All characters in
> the input queue are reprinted when the next character is read.
> (bash(1) handles typeahead this way.) [requires **_BSD_SOURCE**
> or **_SVID_SOURCE**]

IEXTEN Enable implementation-defined input processing. This flag, as
> well as **ICANON** must be enabled for the special characters
> EOL2, LNEXT, REPRINT, WERASE to be interpreted, and for the
> **IUCLC** flag to be effective.

The *c_cc* array defines the terminal special characters. The symbolic
indices (initial values) and meaning are:

VDISCARD
> (not in POSIX; not supported under Linux; 017, SI, Ctrl-O)
> Toggle: start/stop discarding pending output. Recognized when
> **IEXTEN** is set, and then not passed as input.

VDSUSP (not in POSIX; not supported under Linux; 031, EM, Ctrl-Y)
> Delayed suspend character (DSUSP): send **SIGTSTP** signal when
> the character is read by the user program. Recognized when
> **IEXTEN** and **ISIG** are set, and the system supports job control,
> and then not passed as input.

VEOF   (004, EOT, Ctrl-D) End-of-file character (EOF). More pre-
> cisely: this character causes the pending tty buffer to be
> sent to the waiting user program without waiting for end-of-
> line. If it is the first character of the line, the read(2)
> in the user program returns 0, which signifies end-of-file.
> Recognized when **ICANON** is set, and then not passed as input.

VEOL   (0, NUL) Additional end-of-line character (EOL). Recognized
> when **ICANON** is set.

VEOL2 (not in POSIX; 0, NUL) Yet another end-of-line character
> (EOL2). Recognized when **ICANON** is set.

VERASE (0177, DEL, rubout, or 010, BS, Ctrl-H, or also #) Erase char-
> acter (ERASE). This erases the previous not-yet-erased char-
> acter, but does not erase past EOF or beginning-of-line. Rec-

ognized when **ICANON** is set, and then not passed as input.

VINTR (003, ETX, Ctrl-C, or also 0177, DEL, rubout) Interrupt char-
acter (INTR). Send a **SIGINT** signal. Recognized when **ISIG** is
set, and then not passed as input.

VKILL (025, NAK, Ctrl-U, or Ctrl-X, or also @) Kill character
(KILL). This erases the input since the last EOF or begin-
ning-of-line. Recognized when **ICANON** is set, and then not
passed as input.

VLNEXT (not in POSIX; 026, SYN, Ctrl-V) Literal next (LNEXT). Quotes
the next input character, depriving it of a possible special
meaning. Recognized when **IEXTEN** is set, and then not passed
as input.

VMIN Minimum number of characters for noncanonical read (MIN).

VQUIT (034, FS, Ctrl-\) Quit character (QUIT). Send **SIGQUIT** signal.
Recognized when **ISIG** is set, and then not passed as input.

VREPRINT
(not in POSIX; 022, DC2, Ctrl-R) Reprint unread characters
(REPRINT). Recognized when **ICANON** and **IEXTEN** are set, and
then not passed as input.

VSTART (021, DC1, Ctrl-Q) Start character (START). Restarts output
stopped by the Stop character. Recognized when **IXON** is set,
and then not passed as input.

VSTATUS
(not in POSIX; not supported under Linux; status request: 024,
DC4, Ctrl-T). Status character (STATUS). Display status
information at terminal, including state of foreground process
and amount of CPU time it has consumed. Also sends a **SIGINFO**
signal (not supported on Linux) to the foreground process
group.

VSTOP (023, DC3, Ctrl-S) Stop character (STOP). Stop output until
Start character typed. Recognized when **IXON** is set, and then
not passed as input.

VSUSP (032, SUB, Ctrl-Z) Suspend character (SUSP). Send **SIGTSTP**
signal. Recognized when **ISIG** is set, and then not passed as
input.

VSWTCH (not in POSIX; not supported under Linux; 0, NUL) Switch char-
acter (SWTCH). Used in System V to switch shells in *shell
layers*, a predecessor to shell job control.

VTIME Timeout in deciseconds for noncanonical read (TIME).

VWERASE
(not in POSIX; 027, ETB, Ctrl-W) Word erase (WERASE). Recog-

nized when **ICANON** and **IEXTEN** are set, and then not passed as
input.

An individual terminal special character can be disabled by setting
the value of the corresponding *c_cc* element to **_POSIX_VDISABLE**.

The above symbolic subscript values are all different, except that
VTIME, **VMIN** may have the same value as **VEOL**, **VEOF**, respectively. In
noncanonical mode the special character meaning is replaced by the
timeout meaning. For an explanation of **VMIN** and **VTIME**, see the
description of noncanonical mode below.

Retrieving and changing terminal settings

tcgetattr() gets the parameters associated with the object referred
by *fd* and stores them in the *termios* structure referenced by
termios_p. This function may be invoked from a background process;
however, the terminal attributes may be subsequently changed by a
foreground process.

tcsetattr() sets the parameters associated with the terminal (unless
support is required from the underlying hardware that is not avail-
able) from the *termios* structure referred to by *termios_p*.
optional_actions specifies when the changes take effect:

TCSANOW
      the change occurs immediately.

TCSADRAIN
      the change occurs after all output written to *fd* has been
      transmitted. This option should be used when changing parame-
      ters that affect output.

TCSAFLUSH
      the change occurs after all output written to the object
      referred by *fd* has been transmitted, and all input that has
      been received but not read will be discarded before the change
      is made.

Canonical and noncanonical mode

The setting of the **ICANON** canon flag in *c_lflag* determines whether
the terminal is operating in canonical mode (**ICANON** set) or noncanon-
ical mode (**ICANON** unset). By default, **ICANON** is set.

In canonical mode:

* Input is made available line by line. An input line is available
  when one of the line delimiters is typed (NL, EOL, EOL2; or EOF at
  the start of line). Except in the case of EOF, the line delimiter
  is included in the buffer returned by read(2).

* Line editing is enabled (ERASE, KILL; and if the **IEXTEN** flag is
  set: WERASE, REPRINT, LNEXT). A read(2) returns at most one line
  of input; if the read(2) requested fewer bytes than are available
  in the current line of input, then only as many bytes as requested

are read, and the remaining characters will be available for a
future read(2).

* The maximum line length is 4096 chars (including the terminating
  newline character); lines longer than 4096 chars are truncated.
  After 4095 characters, input processing (e.g., **ISIG** and **ECHO*** pro-
  cessing) continues, but any input data after 4095 characters up to
  (but not including) any terminating newline is discarded. This
  ensures that the terminal can always receive more input until at
  least one line can be read.

In noncanonical mode input is available immediately (without the user
having to type a line-delimiter character), no input processing is
performed, and line editing is disabled. The read buffer will only
accept 4095 chars; this provides the necessary space for a newline
char if the input mode is switched to canonical. The settings of MIN
(*c_cc[VMIN]*) and TIME (*c_cc[VTIME]*) determine the circumstances in
which a read(2) completes; there are four distinct cases:

MIN == 0, TIME == 0 (polling read)
        If data is available, read(2) returns immediately, with the
        lesser of the number of bytes available, or the number of
        bytes requested. If no data is available, read(2) returns 0.

MIN > 0, TIME == 0 (blocking read)
        read(2) blocks until MIN bytes are available, and returns up
        to the number of bytes requested.

MIN == 0, TIME > 0 (read with timeout)
        TIME specifies the limit for a timer in tenths of a second.
        The timer is started when read(2) is called. read(2) returns
        either when at least one byte of data is available, or when
        the timer expires. If the timer expires without any input
        becoming available, read(2) returns 0. If data is already
        available at the time of the call to read(2), the call behaves
        as though the data was received immediately after the call.

MIN > 0, TIME > 0 (read with interbyte timeout)
        TIME specifies the limit for a timer in tenths of a second.
        Once an initial byte of input becomes available, the timer is
        restarted after each further byte is received. read(2)
        returns when any of the following conditions is met:

        * MIN bytes have been received.

        * The interbyte timer expires.

        * The number of bytes requested by read(2) has been received.
           (POSIX does not specify this termination condition, and on
           some other implementations read(2) does not return in this
           case.)

        Because the timer is started only after the initial byte
        becomes available, at least one byte will be read. If data is

already available at the time of the call to read(2), the call
behaves as though the data was received immediately after the
call.

POSIX does not specify whether the setting of the **O_NONBLOCK** file
status flag takes precedence over the MIN and TIME settings. If
O_NONBLOCK is set, a read(2) in noncanonical mode may return immedi-
ately, regardless of the setting of MIN or TIME. Furthermore, if no
data is available, POSIX permits a read(2) in noncanonical mode to
return either 0, or -1 with *errno* set to **EAGAIN.**

Raw mode

cfmakeraw() sets the terminal to something like the "raw" mode of the
old Version 7 terminal driver: input is available character by char-
acter, echoing is disabled, and all special processing of terminal
input and output characters is disabled. The terminal attributes are
set as follows:

```
termios_p->c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP
                    | INLCR | IGNCR | ICRNL | IXON);
termios_p->c_oflag &= ~OPOST;
termios_p->c_lflag &= ~(ECHO | ECHONL | ICANON | ISIG | IEXTEN);
termios_p->c_cflag &= ~(CSIZE | PARENB);
termios_p->c_cflag |= CS8;
```

Line control

tcsendbreak() transmits a continuous stream of zero-valued bits for a
specific duration, if the terminal is using asynchronous serial data
transmission. If *duration* is zero, it transmits zero-valued bits for
at least 0.25 seconds, and not more that 0.5 seconds. If *duration* is
not zero, it sends zero-valued bits for some implementation-defined
length of time.

If the terminal is not using asynchronous serial data transmission,
tcsendbreak() returns without taking any action.

tcdrain() waits until all output written to the object referred to by
fd has been transmitted.

tcflush() discards data written to the object referred to by *fd* but
not transmitted, or data received but not read, depending on the
value of *queue_selector*:

TCIFLUSH
        flushes data received but not read.

TCOFLUSH
        flushes data written but not transmitted.

TCIOFLUSH
        flushes both data received but not read, and data written but
        not transmitted.

tcflow() suspends transmission or reception of data on the object

referred to by *fd*, depending on the value of *action*:

TCOOFF suspends output.

TCOON restarts suspended output.

TCIOFF transmits a STOP character, which stops the terminal device
       from transmitting data to the system.

TCION transmits a START character, which starts the terminal device
       transmitting data to the system.

The default on open of a terminal file is that neither its input nor
its output is suspended.

Line speed
The baud rate functions are provided for getting and setting the val-
ues of the input and output baud rates in the *termios* structure. The
new values do not take effect until **tcsetattr**() is successfully
called.

Setting the speed to **B0** instructs the modem to "hang up". The actual
bit rate corresponding to **B38400** may be altered with **setserial**(8).

The input and output baud rates are stored in the *termios* structure.

cfgetospeed() returns the output baud rate stored in the *termios*
structure pointed to by *termios_p*.

cfsetospeed() sets the output baud rate stored in the *termios* struc-
ture pointed to by *termios_p* to *speed*, which must be one of these
constants:

    B0
    B50
    B75
    B110
    B134
    B150
    B200
    B300
    B600
    B1200
    B1800
    B2400
    B4800
    B9600
    B19200
    B38400
    B57600
    B115200
    B230400

The zero baud rate, **B0**, is used to terminate the connection. If B0

is specified, the modem control lines shall no longer be asserted.
Normally, this will disconnect the line. **CBAUDEX** is a mask for the
speeds beyond those defined in POSIX.1 (57600 and above). Thus,
B57600 & **CBAUDEX** is nonzero.

cfgetispeed() returns the input baud rate stored in the *termios*
structure.

cfsetispeed() sets the input baud rate stored in the *termios* struc‐
ture to *speed*, which must be specified as one of the **Bnnn** constants
listed above for **cfsetospeed**(). If the input baud rate is set to
zero, the input baud rate will be equal to the output baud rate.

cfsetspeed() is a 4.4BSD extension. It takes the same arguments as
cfsetispeed(), and sets both input and output speed.

RETURN VALUE   top

cfgetispeed() returns the input baud rate stored in the *termios*
structure.

cfgetospeed() returns the output baud rate stored in the *termios*
structure.

All other functions return:

0   on success.

-1   on failure and set *errno* to indicate the error.

Note that **tcsetattr**() returns success if *any* of the requested changes
could be successfully carried out. Therefore, when making multiple
changes it may be necessary to follow this call with a further call
to **tcgetattr**() to check that all changes have been performed
successfully.

ATTRIBUTES   top

For an explanation of the terms used in this section, see
attributes(7).

Interface	Attribute	Value
tcgetattr(), **tcsetattr**(), **tcdrain**(), **tcflush**(), **tcflow**(), **tcsendbreak**(), **cfmakeraw**(), **cfgetispeed**(), **cfgetospeed**(), **cfsetispeed**(), **cfsetospeed**(), **cfsetspeed**()	Thread safety	MT-Safe

CONFORMING TO   top

tcgetattr(), tcsetattr(), tcsendbreak(), tcdrain(), tcflush(),
tcflow(), cfgetispeed(), cfgetospeed(), cfsetispeed(), and
cfsetospeed() are specified in POSIX.1–2001.

cfmakeraw() and cfsetspeed() are nonstandard, but available on the
BSDs.

NOTES   top

UNIX V7 and several later systems have a list of baud rates where
after the fourteen values B0, ..., B9600 one finds the two constants
EXTA, EXTB ("External A" and "External B"). Many systems extend the
list with much higher baud rates.

The effect of a nonzero *duration* with **tcsendbreak**() varies. SunOS
specifies a break of *duration* * *N* seconds, where *N* is at least 0.25,
and not more than 0.5. Linux, AIX, DU, Tru64 send a break of
duration milliseconds. FreeBSD and NetBSD and HP–UX and MacOS ignore
the value of *duration*. Under Solaris and UnixWare, **tcsendbreak**()
with nonzero *duration* behaves like **tcdrain**().

SEE ALSO   top

reset(1), setterm(1), stty(1), tput(1), tset(1), tty(1),
ioctl_console(2), ioctl_tty(2), setserial(8)

COLOPHON   top

This page is part of release 4.13 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/.

Linux   2017–09–15   TERMIOS(3)

Pages that refer to this page: _exit(2), ioctl_console(2), ioctl_tty(2), setpgid(2), getpass(3), stdin(3),
tty(4), attributes(7), credentials(7), pty(7), signal-safety(7), termio(7), agetty(8)

Copyright and license for this manual page