NAME

tty, termios − terminals

DESCRIPTION

The **tty** driver family takes care of all user input and output. It governs the keyboard, the console, the serial lines, and pseudo ttys. Input on any of these devices undergoes "input processing", and output undergoes "output processing" according to the standard termios terminal interface.

Input processing

Each terminal device has an input queue. This queue is used to store preprocessed input characters, and to perform the backspacing and erase functions. Some special characters like a newline make the contents of the queue available to a process reading from the terminal. Characters up to and including the newline, or another so-called "line break", may be read by a process. The process need not read all characters at once. An input line may be read byte by byte if one wants to. A line break just makes characters available for reading, thats all.

When data is made available depends on whether the tty is in canonical mode or not. In canonical mode the terminal processes input line by line. A line ends with a newline (**NL**), end-of-file (**EOF**), or end-of-line (**EOL**). Characters that have not been delimited by such a line break may be erased one by one with the **ERASE** character or all at once with the **KILL** character. Once a line break is typed the characters become available to a reading process and can no longer be erased. Once read they are removed from the input queue. Several lines may be gathered in the input queue if no reader is present to read them, but a new reader will only receive one line. Two line breaks are never returned in one read call. The input queue has a maximum length of **MAX_CANON** characters. Any more characters are discarded. One must use **ERASE** or **KILL** to make the terminal functioning again if the input queue fills up. If nonblocking I/O is set then −1 is returned with **errno** set to **EAGAIN** if the reader would otherwise be blocked.

In non-canonical mode (raw mode for short) all characters are immediately available to the reader in principle. One may however tune the terminal to bursty input with the **MIN** and **TIME** parameters, see the raw I/O parameters section below. In raw mode no characters are discarded if the input queue threatens to overflow if the device supports flow control.

Output processing

Characters written to a terminal device may undergo output processing, which is usually just inserting a carriage returns before newlines. A writer may return before all characters are output if the characters can be stored in the output buffers. If not then the writer may be blocked until space is available. If non-blocking I/O is set then only the count of the number of bytes that can be processed immediately is returned. If no characters can be written at all then −1 is returned with **errno** set to **EAGAIN**.

Special characters

Some characters have special functions in some of the terminal modes. These characters are as follows, with the Minix defaults shown in parentheses:

INTR (^?)
: Special input character that is recognized if **ISIG** is set. (For **ISIG** and other flags see the various modes sections below.) It causes a **SIGINT** signal to be sent to all processes in the terminal process group. (See the section on session leaders below.)

QUIT (^\)
: Special input character if **ISIG** is set. Causes a **SIGQUIT** signal to be sent to the terminal process group.

ERASE (^H)
: Special input character if **ICANON** is set. Erases the last character in the current line.

KILL (^U)
: Special input character if **ICANON** is set. Erases the entire line.

EOF (^D)
: Special input character if **ICANON** is set. It is a line break character that is not itself returned to a reader. If EOF is typed with no input present then the read returns zero, which normally causes

the reader to assume that end-of-file is reached.

CR (ˆM)
Special input character if **IGNCR** or **ICRNL** is set. It is a carriage return ('\r'). If **IGNCR** is set then **CR** is discarded. If **ICRNL** is set and **IGNCR** is not set then **CR** is changed into an **NL** and has the same function as **NL.**

NL (ˆJ)
Special input character if **ICANON** is set. It is both a newline ('\n') and a line break.
Special output character if **OPOST** and **ONLCR** are set. A **CR NL** sequence is output instead of just **NL**. (Minix specific, but almost mandatory on any UNIX-like system.)

TAB (ˆI)
Special character on output if **OPOST** and **XTABS** are set. It is transformed into the number of spaces necessary to reach a column position that is a multiple of eight. (Only needed for terminals without hardware tabs.)

EOL (undefined)
Special input character if **ICANON** is set. It is an additional line break.

SUSP (ˆZ)
Special input character if job control is implemented and **ISIG** is set. It causes a **SIGTSTP** signal to be send to the terminal process group. (Minix does not have job control.)

STOP (ˆS)
Special input character if **IXON** is set. It suspends terminal output and is then discarded.

START (ˆQ)
Special output character if **IXON** is set. It starts terminal output if suspended and is then discarded. If **IXANY** is also set then any other character also starts terminal output, but they are not discarded.

REPRINT (ˆR)
Special input character if **IEXTEN** and **ECHO** are set. Reprints the input queue from the last line break onwards. A reprint also happens automatically if the echoed input has been messed up by other output and **ERASE** is typed.

LNEXT (ˆV)
Special input character if **IEXTEN** is set. It is the "literal next" character that causes the next character to be input without any special processing.

DISCARD (ˆO)
Special input character if **IEXTEN** is set. Causes output to be discarded until it is typed again. (Implemented only under Minix-vmd.)

All of these characters except **CR**, **NL** and **TAB** may be changed or disabled under Minix. (Changes to **START** and **STOP** may be ignored under other termios implementations.) The **REPRINT** and **LNEXT** characters are Minix extensions that are commonly present in other implementations. POSIX is unclear on whether **IEXTEN, IGNCR** and **ICRNL** should be active in non-canonical mode, but under Minix they are.

Terminal attributes

The attributes of a terminal, such as whether the mode should be canonical or non-canonical, are controlled by routines that use the **termios** structure as defined in **<termios.h>**:

```
struct termios {
      tcflag_t    c_iflag;        /* input modes */
      tcflag_t    c_oflag;        /* output modes */
      tcflag_t    c_cflag;        /* control modes */
      tcflag_t    c_lflag;        /* local modes */
      speed_t     c_ispeed;       /* input speed */
      speed_t     c_ospeed;       /* output speed */
      cc_t        c_cc[NCCS];     /* control characters */
```

};

The types **tcflag**, **speed_t** and **cc_t** are defined in **<termios.h>** as unsigned integral types.

Input Modes

The **c_iflag** field contains the following single bit flags that control input processing:

ICRNL
Map **CR** to **NL** on input.

IGNCR
Ignore **CR** on input. This flag overrides **ICRNL**.

INLCR
Map **NL** to **CR** on input. This is done after the **IGNCR** check.

IXON
Enable start/stop output control.

IXOFF
Enable start/stop input control. (Not implemented.)

IXANY
Allow any character to restart output. (Minix specific.)

ISTRIP
Strip characters to seven bits.

IGNPAR
Ignore characters with parity errors. (Not implemented.)

INPCK
Enable input parity checking. (Not implemented.)

PARMRK
Mark parity errors by preceding the faulty character with '\377', '\0'. The character '\377' is preceded by another '\377' to avoid ambiguity. (Not implemented.)

BRKINT
Send the signal **SIGINT** to the terminal process group when receiving a break condition. (Not implemented.)

IGNBRK
Ignore break condition. If neither **BRKINT** or **IGNBRK** is set a break is input as a single '\0', or if **PARMRK** is set as '\377', '\0', '\0'. (Breaks are always ignored.)

Output Modes

The **c_oflag** field contains the following single bit flags that control output processing:

OPOST
Perform output processing. This flag is the "main switch" on output processing. All other flags are Minix specific.

ONLCR
Transform an **NL** to a **CR NL** sequence on output. Note that a key labeled "RETURN" or "ENTER" usually sends a **CR**. In line oriented mode this is normally transformed into **NL** by **ICRNL**. **NL** is the normal UNIX line delimiter ('\n'). On output an **NL** is transformed into the **CR NL** sequence that is necessary to reach the first column of the next line. (This is a common output processing function for UNIX-like systems, but not always separately switchable by an **ONLCR** flag.)

XTABS
Transform a **TAB** into the number of spaces necessary to reach a column position that is a multiple of eight.

ONOEOT
Discard **EOT** (^D) characters. (Minix-vmd only.)

Control Modes

The **c_cflag** field contains the following single bit flags and bit field for basic hardware control:

CLOCAL
: Ignore modem status lines.

CREAD
: Enable receiver. (The receiver is always enabled.)

CSIZE
: Number of bits per byte. **CSIZE** masks off the values **CS5**, **CS6**, **CS7** and **CS8** that indicate that 5, 6, 7 or 8 bits are used.

CSTOPB
: Send two stop bits instead of one. Two stop bits are normally used at 110 baud or less.

PARENB
: Enable parity generation.

PARODD
: Generate odd parity if parity is generated, otherwise even parity.

HUPCL
: Drop the modem control lines on the last close of the terminal line. (Not implemented.)

Local Modes

The **c_lflag** field contains the following single bit flags that control various functions:

ECHO
: Enable echoing of input characters. Most input characters are echoed as they are. Control characters are echoed as ^*X* where *X* is the letter used to say that the control character is **CTRL**−*X*. The **CR**, **NL** and **TAB** characters are echoed with their normal effect unless they are escaped by **LNEXT**.

ECHOE
: If **ICANON** and **ECHO** are set then echo **ERASE** and **KILL** as one or more backspace-space-backspace sequences to wipe out the last character or the entire line, otherwise they are echoed as they are.

ECHOK
: If **ICANON** and **ECHO** are set and **ECHOE** is not set then output an **NL** after the **KILL** character. (For hardcopy terminals it is best to unset **ECHOE** and to set **ECHOK**.)

ECHONL
: Echo **NL** even if **ECHO** is not set, but **ICANON** is set.

ICANON
: Canonical input. This enables line oriented input and erase and kill processing.

IEXTEN
: Enable implementation defined input extensions.

ISIG
: Enable the signal characters **INTR**, **QUIT** and **SUSP**.

NOFLSH
: Disable the flushing of the input and output queues that is normally done if a signal is sent.

TOSTOP
: Send a **SIGTTOU** signal if job control is implemented and a background process tries to write. (Minix has no job control.)

Input and output speed

The input and output speed are encoded into the **c_ispeed** and **c_ospeed** fields. **<termios.h>** defines the symbols **B0**, **B50**, **B75**, **B110**, **B134**, **B150**, **B200**, **B300**, **B600**, **B1200**, **B1800**, **B2400**, **B4800**, **B9600**, **B19200**, **B38400**, **B57600** and **B115200** as values used to indicate the given baud rates. The zero baud rate, **B0**, if used for the input speed causes the input speed to be equal to the output speed. Setting the output speed to zero hangs up the line. One should use the functions **cfgetispeed()**, **cfgetospeed()**, **cfsetispeed()** and **cfsetospeed()** to get or set a speed, because the **c_ispeed** and

c_ospeed fields may not be visible under other implementations. (The **c_ispeed** and **c_ospeed** fields and the **B57600** and **B115200** symbols are Minix specific.)

Special characters

The **c_cc** array contains the special characters that can be modified. The array has length **NCCS** and is subscripted by the symbols **VEOF**, **VEOL**, **VERASE**, **VINTR**, **VKILL**, **VMIN**, **VQUIT**, **VTIME**, **VSUSP**, **VSTART**, **VSTOP**, **VREPRINT**, **VLNEXT** and **VDISCARD**. All these symbols are defined in **<termios.h>**. Some implementations may give the same values to the **VMIN** and **VTIME** subscripts and the **VEOF** and **VEOL** subscripts respectively, and may ignore changes to **START** and **STOP**. (Under Minix all special characters have their own *c_cc* slot and can all be modified.)

Raw I/O Parameters

The **MIN** and **TIME** parameters can be used to adjust a raw connection to bursty input. **MIN** represents a minimum number of bytes that must be received before a read call returns. **TIME** is a timer of 0.1 second granularity that can be used to time out a read. Setting either of these parameters to zero has special meaning, which leads to the following four possibilities:

MIN > 0, TIME > 0

TIME is an inter-byte timer that is started (and restarted) when a byte is received. A read succeeds when either the minimum number of characters is received or the timer expires. Note that the timer starts **after** the first character, so the read returns at least one byte.

MIN > 0, TIME = 0

Now the timer is disabled, and a reader blocks indefinitely until at least **MIN** characters are received.

MIN = 0, TIME > 0

TIME is now a read timer that is started when a read is executed. The read will return if the read timer expires or if at least one byte is input. (Note that a value of zero may be returned to the reader.)

MIN = 0, TIME = 0

The bytes currently available are returned. Zero is returned if no bytes are available.

User Level Functions

Termios attributes are set or examined, and special functions can be performed by using the functions described in **termios**(3).

Session Leaders and Process Groups

With the use of the **setsid()** function can a process become a session leader. A session leader forms a process group with a process group id equal to the process id of the session leader. If a session leader opens a terminal device file then this terminal becomes the controlling tty of the session leader. Unless the terminal is already the controlling tty of another process, or unless the **O_NOCTTY** flag is used to prevent the allocation of a controlling tty. The process group of the session leader is now remembered as the terminal process group for signals sent by the terminal driver. All the children and grandchildren of the session leader inherit the controlling terminal and process group until they themselves use **setsid()**.

The controlling tty becomes inaccessible to the children of the session leader when the session leader exits, and a hangup signal is sent to all the members of the process group. The input and output queues are flushed on the last close of a terminal and all attributes are reset to the default state.

A special device **/dev/tty** is a synonym for the controlling tty of a process. It allows a process to reach the terminal even when standard input, output and error are redirected. Opening this device can also be used as a test to see if a process has a controlling tty or not.

For Minix a special write-only device **/dev/log** exists for processes that want to write messages to the system console. Unlike the console this device is still accessible when a session leader exits.

Minix-vmd also has a **/dev/log** device, but this device is read-write. All messages written to the log device or to the console when X11 is active can be read from **/dev/log**. The system tries to preserve the log buffer over a reboot so that panic messages reappear in the log if the system happens to crash.

Pseudo Terminals

Pseudo ttys allow a process such as a remote login daemon to set up a terminal for a remote login session. The login session uses a device like **/dev/ttyp0** for input and output, and the remote login daemon uses the device **/dev/ptyp0** to supply input to or take output from the login session and transfer this to or from the originating system. So the character flow may be: Local user input sent to the remote system is written to **/dev/ptyp0** by the remote login daemon, undergoes input processing and appears on **/dev/ttyp0** as input to the login session. Output from the login session to **/dev/ttyp0** undergoes output processing, is read from **/dev/ptyp0** by the remote login daemon and is send over to the local system to be displayed for the user. (So there are only four data streams to worry about in a pseudo terminal.)

A pseudo terminal can be allocated by trying to open all the controlling devices **/dev/pty***nn* one by one until it succeeds. Further opens will fail once a pty is open. The process should now fork, the child should become session leader, open the tty side of the pty and start a login session.

If the tty side is eventually closed down then reads from the pty side will return zero and writes return −1 with **errno** set to **EIO**. If the pty side is closed first then a **SIGHUP** signal is sent to the session leader and further reads from the tty side return zero and writes return −1 with **errno** set to **EIO**. (Special note: A line erase may cause up to three times the size of the tty input queue to be sent to the pty reader as backspace overstrikes. Some of this output may get lost if the pty reader cannot accept it all at once in a single read call.)

Backwards compatibility

The **TIOCGETP**, **TIOCSETP**, **TIOCGETC** and **TIOCSETC** ioctl functions that are used by the old **sgtty** terminal interface are still supported by the terminal driver by emulation. Note that these old functions cannot control all termios attributes, so the terminal must be in a relatively sane state to avoid problems.

FILES

The list below shows all devices that Minix and Minix-vmd have. Not all of these devices are configured in by default, as indicated by the numbers (i/j/k, l/m/n) that tell the minimum, default and maximum possible number of these devices for Minix (i/j/k) and Minix-vmd (l/m/n).

/dev/console	System console.
/dev/ttyc[1-7]	Virtual consoles. (0/1/7, 0/1/7)
/dev/log	Console log device.
/dev/tty0[0-3]	Serial lines. (0/2/2, 4/4/4)
/dev/tty[p-w][0-f]	Pseudo ttys. (0/0/64, 1/32/128)
/dev/pty[p-w][0-f]	Associated pseudo tty controllers.

SEE ALSO

stty(1), **termios**(3), **setsid**(2), **read**(2), **write**(2).

BUGS

A fair number of flags are not implemented under Minix (yet). Luckily they are very limited utility and only apply to RS-232, not to the user interface.

AUTHOR

Kees J. Bot (kjb@cs.vu.nl)