NAME

tty – interface to low speed asynchronous devices including typewriters

DESCRIPTION

This section describes the general nature of the interface to low speed asynchrounous devices including the character processing that takes place in UNIX interaction with typewriters. At this level, the discussion applies also to MERT systems.

The low speed asynchronous devices all use the same general software interface, no matter what hardware is involved. The *kl, dc* and *dh* writeups (IV) describe peculiarities of the individual devices; the remainder of this section describes common features of the overall interface.

Typewriter Interface

Typewriter files are opened by *getty* and become a user's input and output file (see *init* (VIII)). The very first process to open a typewriter file defines the process group with which the typewriter is associated. The process group plays a special role in handling quit and interrupt signals, as discussed below. The process group is inherited by a child process during a fork.

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely choked, which is rare, or when the user has accumulated the maximum allowed number of input characters which have not yet been read by some program. Currently this limit is 256 characters. When the input limit is reached all the saved characters are thrown away without notice.

There are a number of modes which can be changed by use of the *stty* system call (II). When first opened, the interface mode is 300 baud; either parity accepted; character echoing and newline as an end of line delimiter. Modes that can be changed by *stty* include the interface speed (if the hardware permits); acceptance of even parity, odd parity, or both; a raw mode in which all characters may be read one at a time; a carriage return (CR) mode in which CR is mapped into newline on input, and either CR or line feed (LF) cause echoing of the sequence LF-CR; mapping of upper case letters into lower case; suppression of echoing; a variety of delays after function characters; and the printing of tabs as spaces. See *getty* (VIII) for the way that terminal speed and type are detected.

Normally, typewriter input is processed in units of lines. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not however necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, erase and kill processing is normally done. The character '#' erases the last character typed, except that it will not erase beyond the beginning of a line or an EOT. The character '@' kills the entire line up to the point where it was typed, but not beyond an EOT. Both these characters operate on a keystroke basis independently of any backspacing or tabbing that may have been done. Either '@' or '#' may be entered literally by preceding it by '\'; the erase or kill character remains, but the '\' disappears.

When desired, all upper-case letters are mapped into the corresponding lower-case letter. The upper-case letter may be generated by preceding it by '\'. In addition, the following escape se-

quences are generated on output and accepted on input:

for	use
`	\'
|	\!
~	\^
{	\(
}	\)

In raw mode, the program reading is awakened on each character. No erase or kill processing is done; and the EOT, quit and interrupt characters are not treated specially. The input parity bit is passed back to the reader, but parity is still generated for output characters.

The ASCII EOT (control-D) character may be used to generate an end of file from a typewriter. When an EOT is received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line, and the EOT is discarded. Thus if there are no characters waiting, which is to say the EOT occurred at the beginning of a line, zero characters will be passed back, and this is the standard end-of-file indication. The EOT is passed back unchanged in raw mode.

When the carrier signal from the dataset drops (usually because the user has hung up his terminal) a *hangup* signal is sent to all processes in the process group associated with that typewriter. Unless other arrangements have been made, this signal causes the processes to terminate. If the hangup signal is ignored, any read returns with an end-of-file indication. Thus programs which read a typewriter and test for end-of-file on their input can terminate appropriately when hung up on.

Three other characters have a special meaning when typed. The ASCII DEL character (sometimes called 'rubout') is not passed to a program but generates an *interrupt* signal which is sent to all processes with the associated control typewriter in the process group associated with that typewriter. Normally each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed-upon location. See *signal* (II).

The ASCII character FS generates the *quit* signal. Its treatment is identical to the interrupt signal except that unless a receiving process has made other arrangements it will not only be terminated but a core image file will be generated. If you find it hard to type this character, try control-\ or control-shift-L.

When one or more characters are written, they are actually transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. When a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold the program is resumed. Even parity is always generated on output. The EOT character is not transmitted (except in raw mode) to prevent terminals which respond to it from hanging up.

FILES

/dev/ttys defines treatment of lines by *init*-VIII /dev/tty? (special) file names of terminals

SEE ALSO

dc (IV), dh (IV), getty (VIII), stty (I, II), gtty (II), signal (II)

DIAGNOSTICS

The error bit (c-bit) is set if the file descriptor does not refer to a device of the proper type or the command is rejected by the line discipline. From C, a negative value indicates an error.

BUGS

On raw-mode output with standard typewriters, parity should be transmitted as specified in the characters written.

TTY (IV)

This page has been left blank intentionally.