NAME

stty, gtty − set and retrieve terminal modes

SYNOPSIS

#include <sys/sgtty.h>

stty (fildes, arg)
struct SGBUF *arg;

gtty (fildes, arg)
struct SGBUF *arg;

DESCRIPTION

Stty and *gtty* are used to set and get various characteristics of a character device referred to by *fildes*. *Fildes* usually refers to a typewriter line but may also refer to certain special devices such as named pipes. The second argument, *arg*, should be a pointer to the SGTTY structure which is defined in the include file **<sys/sgtty.h>** . A copy of this header file is included here for reference:

```
/*              @(#)sgtty.h     3.2             */

/*
 * stty and gtty structure layouts
 *
 * All structures are 6 bytes.
 * For each given command, doing a stty
 * sets the information into the operating
 * system. Doing a gtty retrieves it.
 */


/*
 * Command 0 -- set modes and speeds.
 *              Wait for output to drain and flush any input.
 * Command 1 -- set modes and speeds.
 *              Don't wait or flush.
 */
#define         STTY_MODES  0
#define         STTY_NFMODES                1
struct SGBUF {
                char            sm_ispeed;      /* Input speed */
                char            sm_ospeed;      /* Output speed, data and stop bits */
                char            sm_cmd;                         /* Command = 0 or 1 */
                char            sm_fill;
                int             sm_modes;       /* See below */
};

/*
 * Modes
 */
#define         NCDELAY         0000001                 /* no carriage return delay */
#define         XTABS           0000002                 /* map tabs to spaces on output */
#define         LCASE           0000004                 /* upper case only terminal */
#define         ECHO            0000010                 /* echo all received chars */
#define         CRMOD           0000020                 /* map CR->LF;echo CR or LF as CR-LF*/
#define         RAW             0000040                 /* raw character input */
#define         ODDP            0000100                 /* odd parity rcvd/xmtd */
#define         EVENP           0000200                 /* even parity rcvd/xmtd */
#define         ANYP            0000300                 /* any parity mask */
#define         HDPLX           0000400                 /* Half duplex line */
#define         NOHUP           0001000                 /* don't drop DTR on last close */
```

```
#define     XCLUDE      0002000     /* disallow future opens */
#define     NOSLEEP     0004000     /* dont sleep if nothing is ready */
#define     NTDELAY     0010000     /* no tab delay flag */
#define     NLDELAY     0020000     /* no newline delay flag */
#define     TANDEM      0040000     /* xon/xoff enabled */
#define     STDTTY      0100000     /* non-std tty escapes and kills */

/*
 * Speeds
 */
#define     B0          0
#define     B50         1
#define     B75         2
#define     B110        3
#define     B134        4
#define     B150        5
#define     B200        6
#define     B300        7
#define     B600        8
#define     B1200       9
#define     B1800       10
#define     B2400       11
#define     B4800       12
#define     B9600       13
#define     EXTA        14
#define     EXTB        15

/*
 * Character length and stop bits.
 * Character length does not include parity or stop bits.
 * Ored with sm_ospeeed.
 */
#define     SETSTOP     0200        /* set to change stop or length bits */
#define     ONESTOP     0000
#define     TWOSTOP     0100        /* 1.5 stop bits at 75 baud */
#define     BITS5       0000
#define     BITS6       0020
#define     BITS7       0040
#define     BITS8       0060
#define     SLBITS      0160        /* Mask of stop and length bits */


/*
 * Command 2 -- set line
 * discipline of a line
 */
#define     STTY_LTYPE  2

/*
 * standard line discipline
 */
#define     STDLTYPE    0
struct {
            int         sl_fil1;
            char        sl_cmd;                 /* Command = 2 */
            char        sl_ltype;   /* Line discipline number = 0 */
            int         sl_fil2;
};

/*
 * line disciplines 1 and 2 reserved for
 * project specific line disciplines
 */
```

```
#define         PRJ1LTYPE       1
#define         PRJ2LTYPE       2

/*
 * transparent line discipline
 */
#define         TRSLTYPE        3
struct {
                char            ts_quanta;      /* Sleep quanta */
                char            ts_fil1;
                char            ts_cmd;                         /* Command = 2 */
                char            ts_ltype;       /* Line discipline number = 3 */
                char            ts_brk0;        /* First break character */
                char            ts_brk1;        /* Second break character */
};

/*
 * Half Duplex line discipline
 */
#define         HFLTYPE         4
struct {
                int             sl_fil1;
                char            sl_cmd;                         /* Command = 2 */
                char            sl_ltype;       /* Line discipline number = 4 */
                int             sl_fil2;
};

/*
 * Line disciplines 5 through 9 reserved for
 * future common line disciplines
 */
#define         RSV5LTYPE       5
#define         RSV6LTYPE       6
#define         RSV7LTYPE       7
#define         RSV8LTYPE       8
#define         RSV9LTYPE       9

/*
 * Command 3 -- set terminal type
 */
#define         STTY_TERM       3
struct {
                char            st_flgs;        /* terminal flags (see below) */
                char            st_fil1;
                char            st_cmd;                         /* Command = 3 */
                char            st_term;        /* Terminal type */
                int             st_fil2;
};

/*
 * Terminal types
 */
#define         TERM_NONE       0               /* tty */
#define         TERM_TEC        1               /* TEC Scope */
#define         TERM_V61        2               /* DEC VT61 */
#define         TERM_V10        3               /* DEC VT100 */
#define         TERM_TEX        4               /* Tektronix 4023 */
#define         TERM_D40        5               /* TTY Mod 40/1 */
#define         TERM_H45        6               /* Hewlitt-Packard 45 */
#define         TERM_D42        7               /* TTY Mod 40/2B */

/*
 * Terminal flags
```

```
 */
#define         TM_NONE                 0               /* use default flags */
#define         TM_SNL                  1               /* special newline flag */
#define         TM_ANL                  2               /* auto newline on column 80 */
#define         TM_LCF                  4               /* last col of last row special */
#define         TM_CECHO        010             /* echo terminal cursor control */
#define         TM_CINVIS       020             /* do not send esc sequences to user */
#define         TM_SET                          0200            /* must be on to set/reset flags */


/*
 * Command 4 -- set variable portion
 * of crt screen
 */
#define         STTY_SCREEN     4
struct {
                char            ss_crow;        /* cursor's row */
                                                        /* ignored on stty */
                char            ss_fil1;
                char            ss_cmd;                 /* Command = 4 */
                char            ss_vrow;        /* variable row */
                int             ss_fil2;
};


/*
 * Command 0377 -- enable spy
 */
#define         STTY_SPY        0377
struct {
                int             sy_fil1;
                char            sy_cmd;                 /* Command = 0377 */
                char            sy_scmd;        /* 0=>delete spy; 1=>initiate spy */
                int             sy_fil2;
};


/*
 * stty info for named pipes ONLY
 */
#define         STTY_NPIPE      0376
struct {
                int             sp_rflg;        /* read flag; 0 => nosleep */
                char            sp_cmd;                 /* Command = 0376 */
                char            sp_fil1;
                int             sp_wflg;        /* write flag; 0 => nosleep */
};
```

Notice that the format of the SGTTY structure may be different for various *stty*/*gtty* commands. The only byte which is always used is the command byte. This byte appears in all the structure definitions and must be filled in by the user before utilizing the *stty* or *gtty* system calls. The user should declare any *stty* or *gtty* structures using the structure tag-name **SGBUF**. Note, however, that references to the structure may be made using the **SGBUF** structure or any of the untagged structures defined above.

If the command byte is **STTY_MODES** or **STTY_NFMODES** the system call will set or get the input speed, output speed, number of data and stop bits, and the teletype modes. If an attempt is made to change the speed of a nonprogrammable device (e.g., DJ-11) or change the speed to a unsupported speed (e.g., **B4800** on a DC-11) the present speed is left unchanged.

Certain modes require further explaination:

LCASE Map upper case to lower case on input; map lower case to uppercase on output. Map | to \!; ‘ to \’; { to \(; } to \); ~ to \^; and map \<C> to upper case input where <C> is any upper case character.

RAW In raw mode, every character is passed immediately to the program without waiting for a full line to be typed. No input characters have special meaning. (e.g., The interrupt character (DEL) will not cause the program to be interrupted but will be sent to the program as a character.) **LCASE** and **CRMOD** will still cause input mapping. Output character processing is unaffected. If the transmitter has been stopped by the ESC key, setting **RAW** will release it. Note, however, that this can only be effective if the **STTY_NFMODES** command is utilized. Otherwise the program will wait for the ESC key to be depressed again. Input and output data width is eight bits, but the eighth bit may be a parity bit depending upon the setting of **ODDP** and **EVENP**.

ODD, EVENP

For the standard line discipline a character will be rejected unless its parity matches that expected. If both bits are set either parity is accepted and even parity is transmitted. If both bits are set and **RAW** is set the parity is visible to and supplied by the user on input and output. If neither bit is set no characters are accepted and even parity is transmitted.

HDPLX For those communications controllers with the capability, disable reception during transmission.

XCLUDE When set, no one may open the line. Cleared upon the last close.

NOSLEEP

Return a zero if a read is performed and no characters are present. Don't wait to flush output on *close* or *stty*. Don't wait for carrier in the first *read* or *write* after an *open* if carrier is not up. Normally a process will block when waiting for carrier to come up after an *open*. This roadblock will take place in the first *read* or *write* not the *open*.

STDTTY Change the erase character from # to _ and the delete line character from @ to $. In addition to CR and LF, wake up on / and !, and generate an interrupt upon reception of & or DEL.

It is also possible for the user to set the number of data and stop bits if the defaults are not satisfactory. The default is **TWOSTOP** at **B75** and **B110**, **ONESTOP** otherwise; and **BITS5** for **B75**, **BITS7** plus one even parity bit otherwise. In order to set these bits the **SETSTOP** bit must also be set.

Normally a *stty* will wait for output to flush before doing anything. This can be circumvented by using the command **STTY_NFMODES**.

The **STTY_LTYPE** command may be used to change the line discipline (protocol) used on a line. The normal CB-UNIX line discipline is **STDLTYPE**. Also commonly supported is the half duplex line discipline **HFLTYPE**, and the transparent line discipline **TRSLTYPE**. Different line disciplines expect different format in the *stty/ gtty* structure. **STLDTYPE** and **HFLTYPE** require no additional information.

TRSLTYPE is a line discipline that allows the user full eight bit transparency on input and output with or without parity. For this line discipline a *write* will perform no mapping. A read will return upon the occurrence of the first of three conditions as specified by the user:

1) The requested number of characters have arrived.

2) The number of seconds, *ts_quanta*, has elapsed.

3) A break character has arrived.

If *ts_quanta* is zero timing is disabled, otherwise *ts_quanta* is the maximum wait time in seconds. If *ts_brk0* and *ts_brk1* are both zero no break characters will awaken the process. If *ts_brk1* is 0377 then *ts_brk0* is taken as a single break character. Otherwise both break characters are assumed valid. **NCDELAY**, **XTABS**, **LCASE**, **ECHO**, **CRMOD**, **RAW**, **NTDELAY**, **NLDELAY**, and **STDTTY** have no meaning for this line discipline.

The **STTY_TERM** command is used to specify the type of CRT connected to a line. **TERM_NONE** is the standard, non-CRT, type. If a type other than **TERM_NONE** is specified input and output mapping will occur for the CRT language defined in the header file **<crtctl.h>**. In this case the ESC character takes on special meaning, escaping the subsequent characters on input and output. The terminal flags *st_flag* and modes *st_modes* are given a default set of values when a terminal type is set. The modes may be subsequently changed with a **STTY_MODES** command. The flags may be changed by setting the **TM_SET** bit when changing the terminal type and specifying the flag bits. The flag bits require further clarification:

TM_SNL Handle new lines specially if the terminal driver is so equipped.

TM_ANL Provide a carriage return and newline when writing beyond column eighty.

TM_LCF Immediately before placing a character in the last column and last row, delete the top line, print the character in the last column of the now second to last row, and then move the cursor to column one of the new last line. This function is required for terminal that move the cursor to "bad" places when printing in the last position.

TM_CECHO
Echo the control sequences such as cursor up when received.

TM_CINVIS
Do not pass the cursor control characters to the user program on input.

The **STTY_SCREEN** command is also used to set or get information about CRT terminals. It is used to set or get the variable row for split screen operation and to get the current row number of the cursor.

The **STTY_SPY** command will cause any output directed to the terminal specified by *fildes* to be copied to the controlling terminal of the program performing the *stty*. Only one spy operation may be active in the entire system at any time. The spy continues until explicitly turned off. Currently spy is only effective on lines using the **STDLTYPE** line discipline.

Finally, the **STTY_NPIPE** command can be used on named pipes to prevent *read*s or *write*s to named pipes from roadblocking. If *sp_rflg* is nonzero then a reader of the named pipe will roadblock when a *read* is performed with no data in the pipe, otherwise a zero is returned immediately. Similarly if *sp_wflg* is nonzero a *write* will roadblock if the pipe is full. When a named pipe is first opened *sp_rflg* is set to one and *sp_wflg* is zero.

Stty has been replaced by *ioctl*(2) in the new implementation of the library.

SEE ALSO

stty(1), ioctl(2)

ASSEMBLER

(stty = 31.)
(file descriptor in r0)
sys stty; arg

(gtty = 32.)
(file descriptor in r0)
sys gtty; arg