tputs(3) - Linux man page

Name

del_curterm, **mvcur**, **putp**, **restartterm**, **set_curterm**, **setterm**, **setupterm**, **tigetflag**, **tigetnum**, **tigetstr**, **tparm**, **tputs**, **vid_attr**, **vid_puts**, **vidattr**, **vidputs** - **curses** interfaces to terminfo database

Synopsis

```
#include <curses.h>
#include <term.h>

int setupterm(char *term, int fildes, int *errret);
int setterm(char *term);
TERMINAL *set_curterm(TERMINAL *nterm);
int del_curterm(TERMINAL *oterm);
int restartterm(char *term, int fildes, int *errret);
char *tparm(char *str, ...);
int tputs(const char *str, int affcnt, int (*putc)(int));
int putp(const char *str);
int vidputs(chtype attrs, int (*putc)(int));
int vidattr(chtype attrs);
int vid_puts(attr_t attrs, short pair, void *opts, int (*putc)(char));
int vid_attr(attr_t attrs, short pair, void *opts);
int mvcur(int oldrow, int oldcol, int newrow, int newcol);
int tigetflag(char *capname);
int tigetnum(char *capname);
char *tigetstr(char *capname);
```

Description

These low-level routines must be called by programs that have to deal directly with the **terminfo** database to handle certain terminal capabilities, such as programming function keys. For all other functionality, **curses** routines are more suitable and their use is recommended.

Initially, **setupterm** should be called. Note that **setupterm** is automatically called by **initscr** and **newterm**. This defines the set of terminal-dependent variables [listed in **terminfo**(5)]. The **terminfo** variables **lines** and **columns** are initialized by **setupterm** as follows:

> If **use_env(FALSE)** has been called, values for **lines** and **columns** specified in **terminfo** are used.
>
> Otherwise, if the environment variables **LINES** and **COLUMNS** exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for **lines** and **columns** specified in the **terminfo** database are used.

The header files **curses.h** and **term.h** should be included (in this order) to get the defi these strings, numbers, and flags. Parameterized strings should be passed through **tparm** to instantiate them. All **terminfo** strings [including the output of **tparm**] should be printed with **tputs** or **putp**. Call the **reset_shell_mode** to restore the tty modes before exiting [see **curs_kernel**(3X)]. Programs which use cursor addressing should output **enter_ca_mode** upon startup and should output **exit_ca_mode** before exiting. Programs desiring shell escapes should call

reset_shell_mode and output **exit_ca_mode** before the shell is called and should output **enter_ca_mode** and call **reset_prog_mode** after returning from the shell.

The **setupterm** routine reads in the **terminfo** database, initializing the **terminfo** structures, but does not set up the output virtualization structures used by **curses**. The terminal type is the character string *term*; if *term* is null, the environment variable **TERM** is used. All output is to file descriptor **fildes** which is initialized for output. If *errret* is not null, then **setupterm** returns **OK** or **ERR** and stores a status value in the integer pointed to by *errret*. A return value of **OK** combined with status of **1** in *errret* is normal. If **ERR** is returned, examine *errret*:

1

means that the terminal is hardcopy, cannot be used for curses applications.

0

means that the terminal could not be found, or that it is a generic type, having too little information for curses applications to run.

-1

means that the **terminfo** database could not be found.

If *errret* is null, **setupterm** prints an error message upon finding an error and exits. Thus, the simplest call is:

setupterm((char *)0, 1, (int *)0);,

which uses all the defaults and sends the output to **stdout**.

The **setterm** routine is being replaced by **setupterm**. The call:

setupterm(*term***, 1, (int *)0)**

provides the same functionality as **setterm(***term***)**. The **setterm** routine is included here for BSD compatibility, and is not recommended for new programs.

The **set_curterm** routine sets the variable **cur_term** to *nterm*, and makes all of the **terminfo** boolean, numeric, and string variables use the values from *nterm*. It returns the old value of **cur_term**.

The **del_curterm** routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as **cur_term**, references to any of the **terminfo** boolean, numeric, and string variables thereafter may refer to invalid memory locations until another **setupterm** has been called.

The **restartterm** routine is similar to **setupterm** and **initscr**, except that it is called afte memory to a previous state (for example, when reloading a game saved as a core image dump). It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different. Accordingly, it saves various tty state bits, calls **setupterm**, and then restores the bits.

The **tparm** routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The **tputs** routine applies padding information to the string *str* and outputs it. The *str* must be a terminfo string variable or the return value from **tparm**, **tgetstr**, or **tgoto**. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a **putchar**-like routine to which the characters are passed, one at a time.

The **putp** routine calls **tputs(***str***, 1, putchar)**. Note that the output of **putp** always goes to **stdout**, not to the *fildes* specified in **setupterm**.

The **vidputs** routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in **curses**(3X). The characters are passed to the **putchar**-like routine *putc*.

The **vidattr** routine is like the **vidputs** routine, except that it outputs through **putchar**.

The **vid_attr** and **vid_puts** routines correspond to vidattr and vidputs, respectively. They use a set of arguments for representing the video attributes plus color, i.e., one of type attr_t for the attributes and one of short for the color_pair number. The **vid_attr** and **vid_puts** routines are designed to use the attribute constants with the *WA_* prefix. The opts argument is reserved for future use. Currently, applications must provide a null pointer for that argument.

The **mvcur** routine provides low-level cursor motion. It takes effect immediately (rather than at the next refresh).

The **tigetflag**, **tigetnum** and **tigetstr** routines return the value of the capability corresponding to the **terminfo** *capname* passed to them, such as **xenl**.

The **tigetflag** routine returns the value **-1** if *capname* is not a boolean capability, or **0** if it is canceled or absent from the terminal description.

The **tigetnum** routine returns the value **-2** if *capname* is not a numeric capability, or **-1** if it is canceled or absent from the terminal description.

The **tigetstr** routine returns the value **(char *)-1** if *capname* is not a string capability, or **0** if it is canceled or absent from the terminal description.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of **terminfo**(5).

char *boolnames[], ***boolcodes[]**, ***boolfnames[]**

char *numnames[], ***numcodes[]**, ***numfnames[]**

char *strnames[], ***strcodes[]**, ***strfnames[]**

These null-terminated arrays contain the *capnames*, the **termcap** codes, and the full C names, for each of the **terminfo** variables.

Return Value

Routines that return an integer return **ERR** upon failure and **OK** (SVr4 only specifies "an integer value other than **ERR**") upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return **NULL** on error.

X/Open defines no error conditions. In this implementation

del_curterm
returns an error if its terminal parameter is null.
putp

calls **tputs**, returning the same error-codes.

restartterm
returns an error if the associated call to **setupterm** returns an error.
setupterm
returns an error if it cannot allocate enough memory, or create the initial windows (stdscr, curscr, newscr). Other error conditions are documented above.
tputs
returns an error if the string parameter is null. It does not detect I/O errors: X/Open states that **tputs** ignores the return value of the output function *putc*.

Notes

The **setupterm** routine should be used in place of **setterm**. It may be useful when you want to test for terminal capabilities without committing to the allocation of storage involved in **initscr**.

Note that **vidattr** and **vidputs** may be macros.

Portability

The function **setterm** is not described by X/Open and must be considered non-portable. All other functions are as described by X/Open.

setupterm copies the terminal name to the array **ttytype**. This is not part of X/Open Curses, but is assumed by some applications.

In System V Release 4, **set_curterm** has an **int** return type and returns **OK** or **ERR**. We have chosen to implement the X/Open Curses semantics.

In System V Release 4, the third argument of **tputs** has the type **int (*putc)(char)**.

At least one implementation of X/Open Curses (Solaris) returns a value other than OK/**tputs**. That returns the length of the string, and does no error-checking.

X/Open Curses prototypes **tparm** with a fixed number of parameters, rather than a variable argument list. This implementation uses a variable argument list. Portable applications should provide 9 parameters after the format; zeroes are fine for this purpose.

X/Open notes that after calling **mvcur**, the curses state may not match the actual terminal state, and that an application should touch and refresh the window before resuming normal curses calls. Both ncurses and System V Release 4 curses implement **mvcur** using the SCREEN data allocated in either **initscr** or **newterm**. So though it is documented as a terminfo function, **mvcur** is really a curses function which is not well specified.

X/Open states that the old location must be given for **mvcur**. This implementation allows the caller to use -1's for the old ordinates. In that case, the old location is unknown.

Extended terminal capability names, e.g., as defined by **tic -x**, are not stored in the arrays described in this section.

See Also

curses(3X), **curs_initscr**(3X), **curs_kernel**(3X), **curs_termcap**(3X), **putc**(3), **terminfo**(5)