NAME

curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database

SYNOPSIS

cc [*flag* . . .] *file* . . . **–lcurses** [*library* ..]

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

int setupterm(char **term*, **int** *fildes*, **int** **errret***);**

int setterm(char **term***);**

int set_curterm(TERMINAL **nterm***);**

int del_curterm(TERMINAL **oterm***);**

int restartterm(char **term*, **int** *fildes*, **int** **errret***);**

char ***tparm(char** **str*, **long int** *p1*, **long int** *p2*, **long int** *p3*, **long int** *p4*, **long int** *p5*, **long int** *p6*, **long int** *p7*, **long int** *p8*, **long int** *p9***);**

int tputs(char **str*, **int** *affcnt*, **int (*****putc)(char));**

int putp(char **str***);**

int vidputs(chtype *attrs*, **int (*****putc)(char));**

int vidattr(chtype *attrs***);**

int mvcur(int *oldrow*, **int** *oldcol*, **int** *newrow*, **int** *newcol***);**

int tigetflag(char **capname***);**

int tigetnum(char **capname***);**

int tigetstr(char **capname***);**

MT-LEVEL

Unsafe

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**(4)). 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 headers **<curses.h>** and **<term.h>** should be included (in this order) to get the definitions for 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( )** routine 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 status of **1** in *errret* is normal, **0** means that the terminal could not be found, and –**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 compatibility and is supported at Level 2.

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*.

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 after restoring memory to a previous state. 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.

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 **putpA( )** 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 **mvcur( )** routine provides low-level cursor motion.

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

With the **tigetflag( )** routine, the value –**1** is returned if *capname* is not a boolean capability.

With the **tigetnum( )** routine, the value –**2** is returned if *capname* is not a numeric capability.

With the **tigetstr( )** routine, the value **(char** ∗**)**–**1** is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of **terminfo**(4).

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 VALUES

All routines return the integer **ERR** upon failure and 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.

SEE ALSO

curs_initscr(3X), **curs_kernel**(3X), **curs_termcap**(3X), **curses**(3X), **putc**(3S), **terminfo**(4)

NOTES

The header **<curses.h>** automatically includes the headers **<stdio.h>** and **<unctrl.h>**.

The **setupterm( )** routine should be used in place of **setterm( )**.

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