NAME

def_prog_mode, **def_shell_mode**, **reset_prog_mode**, **reset_shell_mode**, **resetty**, **savetty**, **getsyx**, **setsyx**, **curs_set**, **mvcur**, **napms**, **ripoffline** − low-level *curses* routines

SYNOPSIS

```
#include <curses.h>

int def_prog_mode(void);
int def_shell_mode(void);

int reset_prog_mode(void);
int reset_shell_mode(void);

int resetty(void);
int savetty(void);

void getsyx(int y, int x);
void setsyx(int y, int x);

int curs_set(int visibility);
int mvcur(int oldrow, int oldcol, int newrow, int newcol);
int napms(int ms);
int ripoffline(int line, int (*init)(WINDOW *, int));
```

DESCRIPTION

The following routines give low-level access to various **curses** capabilities. These routines typically are used inside library routines.

def_prog_mode, def_shell_mode

The **def_prog_mode** and **def_shell_mode** routines save the current terminal modes as the "program" (in **curses**) or "shell" (not in **curses**) state for use by the **reset_prog_mode** and **reset_shell_mode** routines. This is done automatically by **initscr**. There is one such save area for each screen context allocated by **newterm**.

reset_prog_mode, reset_shell_mode

The **reset_prog_mode** and **reset_shell_mode** routines restore the terminal to "program" (in **curses**) or "shell" (out of**curses**) state. These are done automatically by **end win**(3X) and, after an **endwin**, by **doupdate**, so they normally are not called.

resetty, savetty

The **resetty** and **savetty** routines save and restore the state of the terminal modes. **sa vetty** saves the current state in a buffer and **resetty** restores the state to what it was at the last call to **savetty**.

getsyx

getsyx stores the coordinates of virtual screen (**newscr**) cursor in *y* and *x*. If**newscr**' s **leaveok**(3X) output option is **TRUE**, **getsyx** stores **−1** in both *y* and *x*. If lines have been removed from the top of the screen using **ripoffline**, *y* includes these lines; therefore, *y* and *x* populated by **getsyx** should be used only as arguments for **setsyx**.

Few applications use this feature; most call **getyx**(3X) instead.

setsyx

setsyx sets the virtual screen (**newscr**) cursor location to (*y*, *x*). **setsyx(−1, −1)** is equivalent to **leaveok(newscr, TRUE)**.

getsyx and **setsyx** are designed to be used by a function that manipulates *curses* windows but seeks to avoid changing the cursor position. Such a function would first call **getsyx**, modify its windows' content, call **wnoutrefresh**(3X) on them, call **setsyx**, then call **doupdate**(3X).

Few applications use this feature; most call **wmove**(3X) instead.

curs_set

curs_set adjusts the cursor visibility to "invisible", "visible", "very visible", as its argument is **0**, **1**, or **2**, respectively. It returns the previous *visibility* if the requested one is supported, and **ERR** otherwise.

mvcur

mvcur provides low-level cursor motion. It takes effect immediately, rather than at the next refresh. Unlike the other low-level output functions, which either write to the standard output stream or are passed a function pointer to perform output, **mvcur** uses a file descriptor derived from the output stream parameter of **newterm**(3X).

One application of **mvcur** accompanies the temporary use of another program to write to the terminal screen. For example, first call **refresh**(3X) to ensure that the screen and the library's model of it are up to date; then call **reset_shell_mode**; write to the screen with the external application; call **reset_prog_mode**; and finally call **mvcur(**. . ., . . ., **−1, −1)** to move the terminal cursor to where *curses* thinks it is, since the library has no knowledge of how the external application moved it.

napms

napms sleeps for *ms* milliseconds. If *ms* exceeds 30,000 (thirty seconds), *ncurses* caps it at that value.

ripoffline

ripoffline provides access to the same facility that **slk_init**(3X) uses to reduce the size of the screen. The application must call **ripoffline** before **initscr**(3X) or **newterm**(3X) so that the latter functions prepare a **stdscr** of the correct size.

- If *line* is positive, **ripoffline** removes a line from the top of what will become **stdscr**.
- If *line* is negative, **ripoffline** removes a line from the bottom of what will become **stdscr**.

When **initscr** initializes *curses*, it calls the *init* function supplied to **ripoffline** by the application with two arguments:

- a pointer to the one-line *WINDOW* that it allocates, and
- an integer with the number of columns in the window.

Inside this *init* function, the values of the integer variables **LINES** and **COLS** (see **curs_variables**(3X)) are not guaranteed to be reliable; it must not call **wrefresh**(3X) or **doupdate**(3X). A**wnoutr efresh**(3X) call is permissible.

ripoffline can be called up to five times before **initscr** or **newterm**.

RETURN VALUE

Except for **curs_set**, these functions return **OK** on success and **ERR** on failure.

curs_set returns the previous cursor visibility, and returns **ERR** if the terminal type does not support the requested *visibility*.

napms always succeeds.

mvcur fails if the position (*newrow*, *newcol*) is outside the screen boundaries.

In *ncurses*,

- **def_prog_mode**, **def_shell_mode**, **reset_prog_mode**, and **reset_shell_mode** return **ERR** if the terminal was not initialized, or if the operating system's function for obtaining terminal settings fails.
- **ripoffline** returns **ERR** if the accumulated quantity of ripped-off lines would exceed the maximum (5).

NOTES

getsyx is a macro; use of the **&** operator before its arguments is unnecessary.

The **endwin** function of both *ncurses* and SVr4 *curses* calls **curs_set** if the latter has previously been called to set the cursor visibility to a value other than normal; that is, either invisible or very visible. There is no way for *ncurses* to determine the initial cursor visibility to restore it.

ncurses imposes a limit of 30 seconds on a delay requested of **napms**.

While the *init* function called by **ripoffline** is specified to return an *int*, *ncurses* pays no attention to its return value.

If **ripoffline** cannot allocate memory for the required *WINDOW* structure backing the ripped-off line, it stores a null pointer to the *WINDOW* pointer argument supplied by the *init* function the application

specifies. The application must check this argument for validity after calling **initscr** and prior to performing *curses* operations on that window.

EXTENSIONS

In *ncurses*, **mvcur** accepts **−1** for either or both old coordinates. This value tells *ncurses* that the old location is unknown, and that it must use only absolute motion, as with the **cursor_address** (**cup**) capability, rather than the least costly combination of absolute and relative motion.

PORTABILITY

Applications employing *ncurses* extensions should condition their use on the visibility of the **NCURSES_VERSION** preprocessor macro.

The *virtual screen* functions *setsyx* and *getsyx* are not described in X/Open Curses Issue 4. SVr4 documents each of them as returning an *int*. This is misleading, as they are macros with no documented semantics for returning values.

All other functions are as described in X/Open Curses. It specifies no error conditions for them, except as described for *curs_set* in section "RETURN VALUE" above.

The System V Interface Definition, Version 4 (1995), specified all of these functions except *curs_set* as returning *OK*.

Older SVr4 man pages warn that the return value of *curs_set* "is currently incorrect". This implementation gets it right, but counting on its correctness anywhere else may be unwise.

X/Open Curses specifies *ripoffline* as returning *OK* with no possibility of failure ("[c]alls to *ripoffline* above this limit [five lines] have no effect but report success").

X/Open Curses notes:

> After use of *mvcur*(), the model Curses maintains of the state of the terminal might not match the actual state of the terminal. An application should touch and refresh the window before resuming conventional use of Curses.

Both *ncurses* and SVr4 *curses* implement *mvcur* using the *SCREEN* object allocated in either **initscr**(3X) or **newterm**(3X). X/Open Curses states that the old location must be given for *mvcur* to accommodate terminals that lack absolute cursor positioning.

If interrupted by a signal, *ncurses* restarts *napms*. That, and the limitation to 30 seconds, differ from other implementations.

SEE ALSO

curses(3X), **curs_initscr**(3X), **curs_outopts**(3X), **curs_refresh**(3X), **curs_scr_dump**(3X), **curs_slk**(3X), **curs_variables**(3X)