NAME

curs_sp_funcs - *curses* screen-pointer extension

SYNOPSIS

```
#include <curses.h>

int alloc_pair_sp(SCREEN* sp, int fg, int bg);
int assume_default_colors_sp(SCREEN* sp, int fg, int bg);
int baudrate_sp(SCREEN* sp);
int beep_sp(SCREEN* sp);
bool can_change_color_sp(SCREEN* sp);
int cbreak_sp(SCREEN* sp);
int color_content_sp(SCREEN* sp, short color, short* r, short* g, short* b);
int curs_set_sp(SCREEN* sp, int visibility);
int def_prog_mode_sp(SCREEN* sp);
int def_shell_mode_sp(SCREEN* sp);

int define_key_sp(SCREEN* sp, const char * definition, int keycode);
int delay_output_sp(SCREEN* sp, int ms);
int doupdate_sp(SCREEN* sp);
int echo_sp(SCREEN* sp);
int endwin_sp(SCREEN* sp);
char erasechar_sp(SCREEN* sp);
int erasewchar_sp(SCREEN* sp, wchar_t *wc);
int extended_color_content_sp(SCREEN * sp, int color, int * r, int * g, int * b);
int extended_pair_content_sp(SCREEN* sp, int pair, int * fg, int * bg);
int extended_slk_color_sp(SCREEN* sp, int pair);

void filter_sp(SCREEN* sp);
int find_pair_sp(SCREEN* sp, int fg, int bg);
int flash_sp(SCREEN* sp);
int flushinp_sp(SCREEN* sp);
int free_pair_sp(SCREEN* sp, int pair);
int get_escdelay_sp(SCREEN* sp);
int getmouse_sp(SCREEN* sp, MEVENT* event);
WINDOW* getwin_sp(SCREEN* sp, FILE* filep);
int halfdelay_sp(SCREEN* sp, int tenths);
bool has_colors_sp(SCREEN* sp);

bool has_ic_sp(SCREEN* sp);
```

bool has_il_sp(SCREEN* *sp*);
int has_key_sp(SCREEN* *sp*, int *c*);
bool has_mouse_sp(SCREEN* *sp*);
int init_color_sp(SCREEN* *sp*, short *color*, short *r*, short *g*, short *b*);
int init_extended_color_sp(SCREEN* *sp*, int *color*, int *r*, int *g*, int *b*);
int init_extended_pair_sp(SCREEN* *sp*, int *pair*, int *fg*, int *bg*);
int init_pair_sp(SCREEN* *sp*, short *pair*, short *fg*, short *bg*);
int intrflush_sp(SCREEN* *sp*, WINDOW* *win*, bool *bf*);
int is_cbreak_sp(SCREEN* *sp*);

int is_echo_sp(SCREEN* *sp*);
int is_nl_sp(SCREEN* *sp*);
int is_raw_sp(SCREEN* *sp*);
bool is_term_resized_sp(SCREEN* *sp*, int *lines*, int *columns*);
bool isendwin_sp(SCREEN* *sp*);
int key_defined_sp(SCREEN* *sp*, const char **definition*);
char* keybound_sp(SCREEN* *sp*, int *keycode*, int *count*);
NCURSES_CONST char* keyname_sp(SCREEN* *sp*, int *c*);
int keyok_sp(SCREEN* *sp*, int *keycode*, bool *enable*);
char killchar_sp(SCREEN* *sp*);

int killwchar_sp(SCREEN* *sp*, wchar_t **wc*);
char* longname_sp(SCREEN* *sp*);
int mcprint_sp(SCREEN* *sp*, char **data*, int *len*);
int mouseinterval_sp(SCREEN* *sp*, int *erval*);
mmask_t mousemask_sp(SCREEN* *sp*, mmask_t *newmask*, mmask_t **oldmask*);
int mvcur_sp(SCREEN* *sp*, int *oldrow*, int *oldcol*, int *newrow*, int *newcol*);
int napms_sp(SCREEN* *sp*, int *ms*);
WINDOW* newpad_sp(SCREEN* *sp*, int *nrows*, int *ncols*);
SCREEN* new_prescr(void);
SCREEN* newterm_sp(SCREEN* *sp*, const char **type*, FILE **outfd*, FILE **infd*);

WINDOW* newwin_sp(SCREEN* *sp*, int *nlines*, int *ncols*, int *begin_y*, int *begin_x*);
int nl_sp(SCREEN* *sp*);
int nocbreak_sp(SCREEN* *sp*);
int noecho_sp(SCREEN* *sp*);
void nofilter_sp(SCREEN* *sp*);
int nonl_sp(SCREEN* *sp*);
void noqiflush_sp(SCREEN* *sp*);
int noraw_sp(SCREEN* *sp*);

int pair_content_sp(SCREEN* *sp*, short *pair*, short* *fg*, short* *bg*);
void qiflush_sp(SCREEN* *sp*);

int raw_sp(SCREEN* *sp*);
void reset_color_pairs_sp(SCREEN* *sp*);
int reset_prog_mode_sp(SCREEN* *sp*);
int reset_shell_mode_sp(SCREEN* *sp*);
int resetty_sp(SCREEN* *sp*);
int resize_term_sp(SCREEN* *sp*, int *lines*, int *columns*);
int resizeterm_sp(SCREEN* *sp*, int *lines*, int *columns*);
int ripoffline_sp(SCREEN* *sp*, int *line*, int (init*)(WINDOW* *win*, int *fmt*));**
int savetty_sp(SCREEN* *sp*);
int scr_init_sp(SCREEN* *sp*, const char **filename*);

int scr_restore_sp(SCREEN* *sp*, const char **filename*);
int scr_set_sp(SCREEN* *sp*, const char **filename*);
int set_escdelay_sp(SCREEN* *sp*, int *ms*);
int set_tabsize_sp(SCREEN* *sp*, int *cols*);
int slk_attrset_sp(SCREEN* *sp*, const chtype *a*);
int slk_attr_set_sp(SCREEN* *sp*, const attr_t *attrs*, short *pair*, voidopts*);**
int slk_attroff_sp(SCREEN* *sp*, const chtype *a*);
int slk_attron_sp(SCREEN* *sp*, const chtype *a*);
attr_t slk_attr_sp(SCREEN* *sp*);
int slk_clear_sp(SCREEN* *sp*);

int slk_color_sp(SCREEN* *sp*, short *pair*);
int slk_init_sp(SCREEN* *sp*, int *fmt*);
char* slk_label_sp(SCREEN* *sp*, int *labnum*);
int slk_noutrefresh_sp(SCREEN* *sp*);
int slk_refresh_sp(SCREEN* *sp*);
int slk_restore_sp(SCREEN* *sp*);
int slk_set_sp(SCREEN* *sp*, int *labnum*, const char * *label*, int *fmt*);
int slk_touch_sp(SCREEN* *sp*);
int start_color_sp(SCREEN* *sp*);
attr_t term_attrs_sp(SCREEN* *sp*);

chtype termattrs_sp(SCREEN* *sp*);
char* termname_sp(SCREEN* *sp*);
int typeahead_sp(SCREEN* *sp*, int *fd*);
int unget_wch_sp(SCREEN* *sp*, const wchar_t *wc*);

int ungetch_sp(SCREEN* *sp*, int *c*);
int ungetmouse_sp(SCREEN* *sp*, MEVENT* *event*);
int use_default_colors_sp(SCREEN* *sp*);
void use_env_sp(SCREEN* *sp*, bool *bf*);
int use_legacy_coding_sp(SCREEN* *sp*, int *level*);
void use_tioctl_sp(SCREEN **sp*, bool *bf*);

int vid_attr_sp(SCREEN* *sp*, attr_t *attrs*, short *pair*, void * *opts*);
int vid_puts_sp(SCREEN* *sp*, attr_t *attrs*, short *pair*, void * *opts*, NCURSES_SP_OUTC *putc*);
int vidattr_sp(SCREEN* *sp*, chtype *attrs*);
int vidputs_sp(SCREEN* *sp*, chtype *attrs*, NCURSES_SP_OUTC *putc*);
wchar_t* wunctrl_sp(SCREEN* *sp*, cchar_t **wch*);

#include <form.h>

FORM* new_form_sp(SCREEN* *sp*, FIELD ***fields*);

#include <menu.h>

MENU* new_menu_sp(SCREEN* *sp*, ITEM ***items*);

#include <panel.h>

PANEL* ceiling_panel(SCREEN* *sp*);
PANEL* ground_panel(SCREEN* *sp*);
void update_panels_sp(SCREEN* *sp*);

#include <term.h>

int del_curterm_sp(SCREEN* *sp*, TERMINAL **oterm*);
int putp_sp(SCREEN* *sp*, const char **str*);
int restartterm_sp(SCREEN* *sp*, NCURSES_CONST charterm*, int *filedes*, int **errret*);**
TERMINAL* set_curterm_sp(SCREEN* *sp*, TERMINALnterm*);**
int tgetent_sp(SCREEN* *sp*, char **bp*, const char **name*);
int tgetflag_sp(SCREEN* *sp*, const char **capname*);
int tgetnum_sp(SCREEN* *sp*, const char **capname*);
char* tgetstr_sp(SCREEN* *sp*, const char **capname*, char ***area*);
char* tgoto_sp(SCREEN* *sp*, const char **capname*, int *col*, int *row*);
int tigetflag_sp(SCREEN* *sp*, const char **capname*);

int tigetnum_sp(SCREEN* *sp***, const char *****capname***);**
char* tigetstr_sp(SCREEN* *sp***, const char *****capname***);**
/ tparm_sp may use 9 long parameters rather than being variadic */*
char* tparm_sp(SCREEN* *sp***, const char *****str***, ...);**
int tputs_sp(SCREEN* *sp***, const char *****str***, int** *affcnt***, NCURSES_SP_OUTC** *putc***);**

#include <unctrl.h>

NCURSES_CONST char* unctrl_sp(SCREEN* *sp***, chtype** *ch***);**

DESCRIPTION

This implementation can be configured to provide a set of functions which improve the ability to manage multiple screens. This feature can be added to any of the configurations supported by *ncurses*; it adds new symbols without changing the meaning of any of the existing ones.

Improved Functions

Most of the functions are new versions of existing functions. A parameter is added at the front of the parameter list. It is a *SCREEN* pointer.

The existing functions all use the current screen, which is a static variable. The extended functions use the specified screen, thereby reducing the number of variables which must be modified to update multiple screens.

New Functions

Here are the new functions:

ceiling_panel
: this returns a pointer to the topmost panel in the given screen.

ground_panel
: this returns a pointer to the lowest panel in the given screen.

new_prescr
: when creating a new screen, the library uses static variables which have been preset, e.g., by **use_env**(3X), **filter**(3X), etc. With the screen-pointer extension, there are situations where it must create a current screen before the unextended library does. The **new_prescr** function is used internally to handle these cases. It is also provided to allow applications to customize library initialization.

NOTES

This extension introduces some new names:

NCURSES_SP_FUNCS
: This is set to the library patch-level number. In the unextended library, this is zero (0), to make it useful for checking if the extension is provided.

NCURSES_SP_NAME
: The new functions are named using the macro *NCURSES_SP_NAME*, which hides the actual implementation. Currently this adds a "_sp" suffix to the name of the unextended function. This manual page indexes the extensions showing the full name. However the proper usage of these functions uses the macro, to provide for the possibility of changing the naming convention for specific library configurations.

NCURSES_SP_OUTC
: This is a new function-pointer type to use in the screen-pointer functions where an *NCURSES_OUTC* is used in the unextended library.

NCURSES_OUTC
: This is a function-pointer type used for the cases where a function passes characters to the output stream, e.g., **vidputs**(3X).

PORTABILITY

These routines are specific to *ncurses*. They were not supported on Version 7, BSD or System V implementations. It is recommended that any code depending on *ncurses* extensions be conditioned using *NCURSES_SP_FUNCS*.

SEE ALSO

curses(3X), **curs_opaque**(3X), **curs_threads**(3X)