NAME

curs_window, newwin, delwin, mvwin, subwin, derwin, mvderwin, dupwin, wsyncup, syncok, wcursyncup, wsyncdown – create curses windows

SYNOPSIS

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

#include <curses.h>

WINDOW *newwin(int *nlines*, **int** *ncols*, **int** *begin_y*, **int** *begin_x*);

int delwin(WINDOW **win*);

int mvwin(WINDOW **win*, **int** *y*, **int** *x*);

WINDOW *subwin(WINDOW **orig*, **int** *nlines*, **int** *ncols*, **int** *begin_y*, **int** *begin_x*);

WINDOW *derwin(WINDOW **orig*, **int** *nlines*, **int** *ncols*, **int** *begin_y*, **int** *begin_x*);

int mvderwin(WINDOW **win*, **int** *par_y*, **int** *par_x*);

WINDOW *dupwin(WINDOW **win*);

void wsyncup(WINDOW **win*);

int syncok(WINDOW **win*, **bool** *bf*);

void wcursyncup(WINDOW **win*);

void wsyncdown(WINDOW **win*);

MT-LEVEL

Unsafe

DESCRIPTION

The **newwin( )** routine creates and returns a pointer to a new window with the given number of lines, *nlines*, and columns, *ncols*. The upper left-hand corner of the window is at line *begin_y*, column *begin_x*. If either *nlines* or *ncols* is zero, they default to **LINES** — *begin_y* and **COLS** — *begin_x*. A new full-screen window is created by calling **newwin(0,0,0,0)**.

The **delwin( )** routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.

The **mvwin( )** routine moves the window so that the upper left-hand corner is at position (*x*, *y*). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.

The **subwin( )** routine creates and returns a pointer to a new window with the given number of lines, *nlines*, and columns, *ncols*. The window is at position (*begin_y*, *begin_x*) on the screen. (This position is relative to the screen, and not to the window *orig*.) The window is made in the middle of the window *orig*, so that changes made to one window will affect both windows. The subwindow shares memory with the window *orig*. When using this routine, it is necessary to call **touchwin( )** or **touchline( )** on *orig* before calling **wrefresh( )** on the subwindow.

The **derwin( )** routine is the same as **subwin( )**, except that *begin_y* and *begin_x* are relative to the origin of the window *orig* rather than the screen. There is no difference between the subwindows and the derived windows.

The **mvderwin( )** routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The **dupwin( )** routine creates an exact duplicate of the window *win*.

Each **curses** window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine **wrefresh( )** uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine **wsyncup( )** causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If **syncok( )** is called with second argument **TRUE** then **wsyncup( )** is called automatically whenever there is a change in the window.

The routine **wcursyncup( )** updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine **wsyncdown( )** updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by **wrefresh( )**.

RETURN VALUES

Routines that return an integer return the integer **ERR** upon failure and an integer value other than **ERR** upon successful completion.

delwin( ) returns the integer **ERR** upon failure and **OK** upon successful completion.

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

SEE ALSO

curs_refresh(3X), **curs_touch**(3X), **curses**(3X)

NOTES

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

If many small changes are made to the window, the **wsyncup( )** option could degrade performance.

Note that **syncok( )** may be a macro.