NAME

newwin, **delwin**, **mvwin**, **subwin**, **derwin**, **mvderwin**, **dupwin**, **wsyncup**, **syncok**, **wcursyncup**, **wsyncdown** - create **curses** windows

SYNOPSIS

```
#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);
```

DESCRIPTION

newwin

Calling **newwin** creates and returns a pointer to a new window with the given number of lines and columns. 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)**.

delwin

Calling **delwin** deletes the named window, freeing all memory associated with it (it does not actually

erase the window's screen image). Subwindows must be deleted before the main window can be deleted.

mvwin

Calling **mvwin** 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.

subwin

Calling **subwin** 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. The subwindow shares memory with the window *orig*, so that changes made to one window will affect both windows. When using this routine, it is necessary to call **touchwin** or **touchline** on *orig* before calling **wrefresh** on the subwindow.

derwin

Calling **derwin** is the same as calling **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.

Calling **mvderwin** 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.

dupwin

Calling **dupwin** creates an exact duplicate of the window *win*.

wsyncup

Calling **wsyncup** touches all locations in ancestors of *win* that are changed in *win*. If **syncok** is called with second argument **TRUE** then **wsyncup** is called automatically whenever there is a change in the window.

wsyncdown

The **wsyncdown** routine touches each location in *win* that has been touched in any of its ancestor windows. This routine is called by **wrefresh**, so it should almost never be necessary to call it manually.

wcursyncup

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

RETURN VALUE

Routines that return an integer return the integer **ERR** upon failure and **OK** (SVr4 only specifies "an integer value other than **ERR**") upon successful completion.

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

X/Open defines no error conditions. In this implementation

delwin

returns an error if the window pointer is null, or if the window is the parent of another window.

derwin

returns an error if the parent window pointer is null, or if any of its ordinates or dimensions is negative, or if the resulting window does not fit inside the parent window.

dupwin

returns an error if the window pointer is null.

This implementation also maintains a list of windows, and checks that the pointer passed to **delwin** is one that it created, returning an error if it was not..

mvderwin

returns an error if the window pointer is null, or if some part of the window would be placed off-screen.

mvwin

returns an error if the window pointer is null, or if the window is really a pad, or if some part of the window would be placed off-screen.

newwin

will fail if either of its beginning ordinates is negative, or if either the number of lines or columns is negative.

syncok

returns an error if the window pointer is null.

subwin

returns an error if the parent window pointer is null, or if any of its ordinates or dimensions is negative, or if the resulting window does not fit inside the parent window.

The functions which return a window pointer may also fail if there is insufficient memory for its data structures. Any of these functions will fail if the screen has not been initialized, i.e., with **initscr** or **newterm**.

NOTES

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

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

BUGS

The subwindow functions (**subwin**, **derwin**, **mvderwin**, **wsyncup**, **wsyncdown**, **wcursyncup**, **syncok**) are flaky, incompletely implemented, and not well tested.

The System V curses documentation is very unclear about what **wsyncup** and **wsyncdown** actually do. It seems to imply that they are only supposed to touch exactly those lines that are affected by ancestor changes. The language here, and the behavior of the **curses** implementation, is patterned on the XPG4 curses standard. The weaker XPG4 spec may result in slower updates.

PORTABILITY

The XSI Curses standard, Issue 4 describes these functions.

SEE ALSO

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