NAME

newwin, **delwin**, **mvwin**, **subwin**, **derwin**, **mvderwin**, **dupwin**, **wsyncup**, **syncok**, **wcursyncup**, **wsyncdown** − create and manipulate *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)**.

Regardless of the function used for creating a new window (e.g., **newwin**, **subwin**, **derwin**, **newpad**), rather than a duplicate (with **dupwin**), all of the window modes are initialized to the default values. The following functions set a window's modes after it is created:

idcok, **idlok**, **immedok**, **keypad**, **leaveok**, **nodelay**, **notimeout**, **scrollok**, **setscrreg**, **syncok**, **wbkgdset**, **wbkgrndset**, and **wtimeout**.

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*, its *ancestor*, 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.

mvderwin
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

Functions that return integers return **ERR** upon failure and **OK** upon success.

Functions that return pointers return a null pointer on failure.

ncurses defines several error conditions.

- **delwin** returns **ERR** if *win* is a null pointer, or if it is the parent of another window.

  ncurses maintains a list of windows, and checks that the pointer passed to **delwin** is one that it created, returning **ERR** if it was not.
- **derwin** returns **ERR** if *orig* is a null pointer, or if any of the ordinate or dimension arguments is negative, or if the resulting window does not fit inside the parent window.
- **dupwin** returns **ERR** if *win* is a null pointer.
- **mvderwin** returns **ERR** if *win* is a null pointer, or if any part of the window would be placed off-screen.
- **mvwin** returns **ERR** if *win* is a null pointer, if *win* is a pad, or if any part of the window would be placed off-screen.
- **newwin** returns **ERR** if any of its arguments is negative.
- **subwin** returns **ERR** if *orig* is a null pointer, or if any of the ordinate or dimension arguments is negative, or if the resulting window does not fit inside the parent window.
- **syncok** returns **ERR** if *win* is a null pointer.

Functions that return a window pointer fail if memory allocation for their data structures fails.

All of these functions fail if the screen has not been initialized; see **initscr**(3X) or **newterm**(3X).

NOTES

syncok may be implemented as a macro.

Calling **syncup** on a window and making many small changes to it could degrade performance.

PORTABILITY

X/Open Curses Issue 4 describes these functions. It specifies no error conditions for *delwin*, *derwin*, *dupwin*, *newwin*, *mvderwin*, or *syncok*.

For functions returning integers (except *delwin*), SVr4 describes a successful return value only as “an

integer value other than *ERR*".

Regarding *delwin*, X/Open Curses states that

> [t]he application must delete subwindows before deleting the main window.

If *delwin* is asked to delete a parent window, it can succeed only if the *curses* library keeps a list of its subwindows. SVr4*cur ses* kept a count of the number of subwindows rather than a list. It simply returned **ERR** when asked to delete a subwindow. Solaris X/Open*cur ses* (*xcurses*) does not make even that check, and will delete a parent window that still has subwindows. *PDCurses* also behaves this way.

ncurses 4.0 (1996) and later maintains a list of windows for each screen to ensure that a window has no subwindows before allowing its deletion. NetBSD *curses* has followed suit since 2003.

SVr4 *curses* documentation is unclear about what *wsyncup* and *wsyncdown* actually do. It seems to imply that they are supposed to touch only those lines that are affected by changes to a window's ancestors. The description and behavior of these functions in *ncurses* is patterned on the X/Open Curses standard; this approach may result in slower updates.

SEE ALSO

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