NAME

initscr, **newterm**, **endwin**, **isendwin**, **set_term**, **delscreen** − initialize, manipulate, or tear down *curses* terminal interface

SYNOPSIS

```
#include <curses.h>

WINDOW * initscr(void);
int endwin(void);

bool isendwin(void);

SCREEN * newterm(const char * type, FILE * outf, FILE * inf);
SCREEN * set_term(SCREEN * new);
void delscreen(SCREEN * sp);
```

DESCRIPTION

initscr

initscr determines the terminal type and initializes the library's *SCREEN*, *WINDOW*, and other data structures. It is normally the first*cur ses* function call a program performs. However, an application with unusual needs might employ a few other *curses* functions beforehand:

- **slk_init**(3X) to set up soft-label keys;
- **filter**(3X) if the program is designed to operate in a process pipeline;
- **ripoffline**(3X) to reserve up to five lines at the top and/or bottom of the screen from management by **stdscr**, the standard *curses* window; and
- **use_env**(3X) and/or **use_tioctl**(3X) to configure use of the process environment and operating system's terminal driver, respectively, when determining the dimensions of the terminal display.

Further, a *curses* program might call **newterm** prior to or instead of **initscr** in two specialized cases described in its subsection below.

initscr causes the first **refresh**(3X) call to clear the screen. If errors occur, **initscr** writes an appropriate diagnostic message to the standard error stream and exits; otherwise, it returns a pointer to **stdscr**.

newterm

An application that manages multiple terminals should call **newterm** once for each such device *instead* of **initscr**. **newterm**'s arguments are

- the *type* of the associated terminal, or a null pointer to use the *TERM* environment variable;
- an output stream *outf* connected to the terminal; and
- an input stream *inf* connected to the terminal.

newterm returns a variable of pointer-to-*SCREEN* type, which should be saved for later use with **set_term** and **delscreen**.

newterm passes the file descriptor of the output stream to the *terminfo* function **setupterm**(3X), which returns a pointer to a *TERMINAL* structure that **newterm** stores in the *SCREEN* it returns to the application.

An application that needs to inspect a terminal type's capabilities, so that it can continue to run in a line-oriented mode if the terminal type does not support capabilities the application demands, would also use **newterm**. If at most one terminal connection is needed, the programmer could perform such a capability test, decide the mode in which to operate, then call **delscreen** on the pointer returned by **newterm**, and proceed with either **initscr** or a non-*curses* interface.

endwin

The program must also call **endwin** for each terminal being used before exiting from *curses*. If**newterm** is called more than once for the same terminal, the first terminal referred to must be the last one for which **endwin** is called.

A program should always call **endwin** before exiting the application or temporarily suspending *curses*'s

management of the terminal. **endwin**:

- (if **start_color**(3X) has been called) resets the terminal's foreground and background colors to correspond with those of color pair 0 (the default pair),
- moves the cursor to the lower left-hand corner of the screen,
- (if **start_color**(3X) has been called) restores the default color pair,
- clears the line,
- sets the cursor to normal visibility (see **curs_set**(3X)),
- if applicable, stops cursor-addressing mode using the **exit_ca_mode** (**rmcup**) terminal capability, and
- restores terminal modes (see **reset_shell_mode**(3X)).

Calling **refresh**(3X) or **doupdate**(3X) after a temporary suspension causes *curses* to resume managing the terminal.

isendwin

isendwin returns **TRUE** if **wrefresh**(3X) has not been called since the most recent **endwin** call, and **FALSE** otherwise.

set_term

set_term re-orients the *curses* library's operations to another terminal when the application has arranged to manage more than one with **newterm**. **set_term** expects a *SCREEN* pointer previously returned by **newterm** as an argument, and returns the previous one. **set_term** is the only standard *cur ses* API function that manipulates *SCREEN* pointers; all others affect only the current terminal (but see **curs_sp_funcs**(3X)).

delscreen

delscreen frees the storage backing the supplied *SCREEN* pointer argument. **endwin** does not, so that an application can resume managing a terminal with *curses* after a (possibly conditional or temporary) suspension; see **curs_kernel**(3X). Use**delscr een** after **endwin** when the application has no more need of a terminal device but will not soon exit.

RETURN VALUE

delscreen returns no value. **endwin** returns **OK** on success and **ERR** on failure. **isendwin** returns **TRUE** or **FALSE** as described above.

In *ncurses*,

- **endwin** returns **ERR** if
  - the terminal was not initialized,
  - it is called more than once without updating the screen, or
  - its call of **reset_shell_mode**(3X) returns **ERR**; and
- **newterm** returns **ERR** if it cannot allocate storage for the *SCREEN* structure or the *WINDOW* structures automatically associated with it: **curscr**, **newscr**, and **stdscr**.

Functions that return pointers return null pointers on error. In*ncur ses*, **set_term** does not fail, and **initscr** exits the application if it does not operate successfully.

NOTES

ncurses establishes signal handlers when a function that initializes a *SCREEN*, either **initscr** or **newterm**, is first called. Applications that wish to handle the following signals themselves should set up their corresponding handlers *after* initializing the screen.

SIGINT

ncurses's handler *attempts* to clean up the screen on exit. Although it*usually* w orks as expected, there are limitations.

- Walking the *SCREEN* list is unsafe, since all list management is done without any signal blocking.

- When an application has been built with the *_REENTRANT* macro defined (and corresponding system support), **set_term** uses functions that could deadlock or misbehave in other ways.
- **endwin** calls other functions, many of which use *stdio*(3) or other library functions that are clearly unsafe.

SIGTERM

ncurses uses the same handler as for *SIGINT*, with the same limitations. It is not mentioned in X/Open Curses, but is more suitable for this purpose than *SIGQUIT* (which is used in debugging).

SIGTSTP

ncurses's handler manages the terminal-generated stop signal, used in job control. When resuming the process, *ncurses* discards pending input with **flushinp**(3X) and repaints the screen, assuming that it has been completely altered. It also updates the saved terminal modes with **def_shell_mode**(3X).

SIGWINCH

ncurses handles changes to the terminal's window size, a phenomenon ignored in standardization efforts. It sets a (signal-safe) variable that is later tested by **wgetch**(3X) and **wget_wch**(3X).

- **wgetch** returns the key code **KEY_RESIZE**.
- **wget_wch** returns **KEY_CODE_YES** and sets its *wch* parameter to **KEY_RESIZE**.

At the same time, *ncurses* calls **resizeterm**(3X) to adjust the standard screen **stdscr** and update global variables such as **LINES** and **COLS**.

PORTABILITY

X/Open Curses Issue 4 describes these functions. It specifies no error conditions for them.

Differences

X/Open Curses specifies that portable applications must not call *initscr* more than once.

- The portable way to use *initscr* is once only, using *refresh* to restore the screen after *endwin*.
- *ncurses* permits use of *initscr* after *endwin*.

initscr in BSD, from its inception (1980) through the Net/2 release (1991) returned *ERR* cast to a *WINDOW* pointer when detecting an error. 4.4BSD (1995) instead returned a null pointer. Neither exited the application. It is safe but redundant to check the return value of *initscr* in X/Open Curses.

Calling *endwin* does not dispose of the memory allocated by *initscr* or *newterm*. Deleting a*SCREEN* provides a way to do this.

- X/Open Curses does not say what happens to *WINDOW*s when *delscreen* "frees storage associated with the *SCREEN*" nor does the SVr4 documentation help, adding that it should be called after*endwin* if a *SCREEN* is no longer needed.
- However, every *WINDOW* is implicitly associated with a *SCREEN*, so it is reasonable to expect *delscreen* to dispose of them.
- SVr4 deletes the standard *WINDOW* structures *stdscr* and *curscr* as well as a work area *newscr*. It ignores other windows.
- Since version 4.0 (1996), *ncurses* has maintained a list of all windows for each screen, using that information to delete those windows when *delscreen* is called.
- NetBSD copied this feature of *ncurses* in 2001. *PDCurses* follows the SVr4 model, deleting only the standard *WINDOW* structures and *newscr*.

High-level versus Low-level Functions

Implementations disagree regarding the level of abstraction applicable to a function or property. For example, *SCREEN* (returned by *newterm*) and *TERMINAL* (returned by **setupterm**(3X)) hold file descriptors for the output stream. If an application switches screens using *set_term*, or switches terminals using **set_curterm**(3X), applications using the output file descriptor can behave differently depending on the structure holding the corresponding descriptor.

- NetBSD's *baudrate* function uses the descriptor in *TERMINAL*. *ncurses* and SVr4 use the descriptor in *SCREEN*.
- NetBSD and *ncurses* use the descriptor in *TERMINAL* for terminal I/O modes, e.g., **def_shell_mode**(3X), **def_prog_mode**(3X). SVr4 uses the descriptor in*SCREEN*.

Unset *TERM* Environment Variable

If the *TERM* variable is not set in the environment or has an empty value, *initscr* uses the value "unknown", which normally corresponds to a terminal entry with the **generic** (**gn**) capability. Generic entries are detected by **setupterm**(3X) and cannot be used for full-screen operation. Other implementations may handle a missing or empty*TERM* v ariable differently.

Signal Handlers

Quoting X/Open Curses Issue 7, section 3.1.1:

> Curses implementations may provide for special handling of the SIGINT, SIGQUIT, and SIGTSTP signals if their disposition is SIG_DFL at the time *initscr*() is called. . .
>
> Any special handling for these signals may remain in effect for the life of the process or until the process changes the disposition of the signal.
>
> None of the Curses functions are required to be safe with respect to signals. . .

Section "NOTES" above discusses *ncurses*'s signal handlers.

HISTORY

4BSD (1980) introduced *initscr* and *endwin*.

SVr2 (1984) added *newterm* and *set_term*.

SVr3.1 (1987) supplied *delscreen* and *isendwin*.

SEE ALSO

curses(3X), **curs_kernel**(3X), **curs_refresh**(3X), **curs_slk**(3X), **curs_terminfo**(3X), **curs_util**(3X), **curs_variables**(3X)