NAME

start_color, **has_colors**, **can_change_color**, **init_pair**, **init_color**, **init_extended_pair**, **init_extended_color**, **color_content**, **pair_content**, **extended_color_content**, **extended_pair_content**, **reset_color_pairs**, **COLOR_PAIR**, **PAIR_NUMBER**, **COLORS**, **COLOR_PAIRS**, **COLOR_BLACK**, **COLOR_RED**, **COLOR_GREEN**, **COLOR_YELLOW**, **COLOR_BLUE**, **COLOR_MAGENTA**, **COLOR_CYAN**, **COLOR_WHITE**, **A_COLOR** − manipulate terminal colors with *curses*

SYNOPSIS

```
#include <curses.h>

/* variables */
int COLOR_PAIRS;
int COLORS;

int start_color(void);

bool has_colors(void);
bool can_change_color(void);

int init_pair(short pair, short f, short b);
int init_color(short color, short r, short g, short b);
/* extensions */
int init_extended_pair(int pair, int f, int b);
int init_extended_color(int color, int r, int g, int b);

int color_content(short color, short *r, short *g, short *b);
int pair_content(short pair, short *f, short *b);
/* extensions */
int extended_color_content(int color, int *r, int *g, int *b);
int extended_pair_content(int pair, int *f, int *b);

/* extension */
void reset_color_pairs(void);

/* macros */
int COLOR_PAIR(int n);
PAIR_NUMBER(int attr);
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
A_COLOR
```

DESCRIPTION

Overview

curses supports color attributes on terminals with that capability. Call **start_color** (typically right after **initscr**(3X)) to enable this feature. Colors are always used in pairs. A *color pair* couples a foreground color for characters with a background color for the blank field on which characters are rendered. **init_pair** initializes a color pair. The macro **COLOR_PAIR**(*n*) can then convert the pair to a video attribute.

If a terminal has the relevant capability, **init_color** permits (re)definition of a color. **has_colors** and **can_change_color** return **TRUE** or **FALSE**, depending on whether the terminal has color capability and whether the programmer can change the colors. **color_content** permits extraction of the red, green, and blue components of an initialized color. **pair_content** permits discovery of a color pair's current definition.

Rendering

curses combines the following data to render a character cell. Any of them can include color information.

- *curses* character attributes, as from **waddch**(3X) or **wadd_wch**(3X)
- window attributes, as from **wattrset**(3X) or **wattr_set**(3X)
- window background character attributes, as from **wbkgdset**(3X) or **wbkgrndset**(3X)

Per-character and window attributes are usually set through a function parameter containing attributes including a color pair value. Some functions, such as**wattr_set**, use a separate color pair number parameter .

The background character is a special case: it includes a character code, just as if it were passed to **waddch**.

The *curses* library does the actual work of combining these color pairs in an internal function called from **waddch**:

- If the parameter passed to **waddch** is *blank*, and it uses the special color pair 0, *curses* next checks the window attribute.
  - If the window attribute does not use color pair 0, *curses* uses the color pair from the window attribute.
  - Otherwise, *curses* uses the background character.
- If the parameter passed to **waddch** is *not blank*, or it does not use the special color pair 0, *curses* prefers the color pair from the parameter, if it is nonzero. Otherwise, it tries the window attribute next, and finally the background character.

Some *curses* functions such as **wprintw** call **waddch**. Those do not combine its parameter with a color pair. Consequently those calls use only the window attribute or the background character.

CONSTANTS

ISO 6429 and ECMA-48 define eight standard colors (also known as "ANSI" colors). *cur ses.h* defines object-like macros **COLOR_BLACK**, **COLOR_RED**, **COLOR_GREEN**, **COLOR_YELLOW**, **COLOR_BLUE**, **COLOR_MAGENTA**, **COLOR_CYAN**, and **COLOR_WHITE** accordingly. *curses* assumes that **COLOR_BLACK** is the default background color for all terminals. *ncur ses* offers an extension to override that assumption; see **assume_default_colors**(3X). Some terminals support additional colors that lack standard names.

A_COLOR is a bit mask that, when bitwise "and"-ed with a *chtype*, extracts its color pair identifier.

VARIABLES

COLORS

is initialized by **start_color** to the maximum number of colors the terminal can support.

COLOR_PAIRS

is initialized by **start_color** to the maximum number of color pairs the terminal can support. Often, its value is the product **COLORS** × **COLORS**, but this is not always true.

- A few terminals use the HLS color space (see **start_color** below), ignoring this rule; and
- terminals supporting a large number of colors are limited to the number of color pairs that a *signed short* value can represent.

FUNCTIONS

start_color

The **start_color** routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after **initscr**. **start_color** does this:

- It initializes two global variables, **COLORS** and **COLOR_PAIRS** (respectively defining the maximum number of colors and color pairs the terminal can support).
- It initializes the special color pair **0** to the default foreground and background colors. No other color pairs are initialized.

- It restores the colors on the terminal to the values they had when the terminal was just turned on.
- If the terminal supports the **initc** (**initialize_color**) capability, **start_color** initializes its internal table representing the red, green, and blue components of the color palette.

  The components depend on whether the terminal uses CGA (aka "ANSI") or HLS (i.e., the **hls** (**hue_lightness_saturation**) capability is set). The table is initialized first for eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), using weights that depend upon the CGA/HLS choice. For "ANSI" colors the weights are **680** or **0** depending on whether the corresponding red, green, or blue component is used or not. That permits using **1000** to represent bold/bright colors. After the initial eight colors (if the terminal supports more than eight colors) the components are initialized using the same pattern, but with weights of **1000**. SVr4 uses a similar scheme, but uses **1000** for the components of the initial eight colors.

  start_color does not attempt to set the terminal's color palette to match its built-in table. An application may use **init_color** to alter the internal table along with the terminal's color.

These limits apply to color values and color pairs. Values outside these limits are not valid, and may result in a runtime error:

- **COLORS** corresponds to the terminal database's **max_colors** capability, (see **terminfo**(5)).
- color values are expected to be in the range **0** to **COLORS−1**, inclusive (including **0** and **COLORS−1**).
- a special color value **−1** is used in certain extended functions to denote the *default color* (see **use_default_colors**(3X)).
- **COLOR_PAIRS** corresponds to the terminal database's **max_pairs** capability, (see **terminfo**(5)).
- valid color pair values are in the range **1** to **COLOR_PAIRS−1**, inclusive.
- color pair **0** is special; it denotes "no color".

  Color pair **0** is assumed to be white on black, but is actually whatever the terminal implements before color is initialized. It cannot be modified by the application.

has_colors

has_colors returns **TRUE** if the terminal supports colors and **FALSE** if it does not. **initscr**(3X) or **newterm**(3X) must be called first, but **start_color** need not be. An application might call **has_colors** to decide whether to use color or a video attribute like **A_BOLD** to render text.

Color support in *curses* requires that the terminal type description support the capabilities **max_colors** (**colors**), **max_pairs** (**pairs**), and any of

- **set_foreground** (**setf**) and **set_background** (**setb**);
- **set_a_foreground** (**setaf**) and **set_a_background** (**setab**); or
- **set_color_pair** (**scp**).

can_change_color

The **can_change_color** routine requires no arguments. It returns**TR UE** if the terminal supports colors and can change their definitions; other, it returns **FALSE**. This routine facilitates writing terminal-independent programs.

init_pair

The **init_pair** routine changes the definition of a color pair. It takes three arguments: the number of the color pair to be changed, the foreground color number, and the background color number. For portable applications:

- The first argument must be a valid color pair value. If default colors are used (see **use_default_colors**(3X)) the upper limit is adjusted to allow for extra pairs which use a default color in foreground and/or background.

- The second and third arguments must be valid color values.

If the color pair was previously initialized, the screen is refreshed and all occurrences of that color pair are changed to the new definition.

As an extension, *ncurses* allows you to set color pair **0** via the **assume_default_colors**(3X) routine, or to specify the use of default colors (color number **−1**) if you first invoke the **use_default_colors**(3X) routine.

init_extended_pair
Because **init_pair** uses signed **short**s for its parameters, that limits color pairs and color-values to 32767 on modern hardware. The extension **init_extended_pair** uses **int**s for the color pair and color-value, allowing a larger number of colors to be supported.

init_color
The **init_color** routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components).

- The first argument must be a valid color value; default colors are not allowed here. (See the section **Colors** for the default color index.)
- Each of the last three arguments must be a value in the range **0** through **1000**.

When **init_color** is used, all occurrences of that color on the screen immediately change to the new definition.

init_extended_color
Because **init_color** uses signed **short**s for its parameters, that limits color-values and their red, green, and blue components to 32767 on modern hardware. The extension **init_extended_color** uses **int**s for the color value and for setting the red, green, and blue components, allowing a larger number of colors to be supported.

color_content
The **color_content** routine gives programmers a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of **short**s for storing the information about the amounts of red, green, and blue components in the given color.

- The first argument must be a valid color value, i.e., **0** through **COLORS−1**, inclusive.
- The values that are stored at the addresses pointed to by the last three arguments are in the range **0** (no component) through **1000** (maximum amount of component), inclusive.

extended_color_content
Because **color_content** uses signed **short**s for its parameters, that limits color-values and their red, green, and blue components to 32767 on modern hardware. The extension **extended_color_content** uses **int**s for the color value and for returning the red, green, and blue components, allowing a larger number of colors to be supported.

pair_content
The **pair_content** routine allows programmers to find out what colors a given color pair consists of. It requires three arguments: the color pair number, and two addresses of **short**s for storing the foreground and the background color numbers.

- The first argument must be a valid color value, i.e., in the range **1** through **COLOR_PAIRS−1**, inclusive.
- The values that are stored at the addresses pointed to by the second and third arguments are in the range **0** through **COLORS**, inclusive.

extended_pair_content
Because **pair_content** uses signed **short**s for its parameters, that limits color pair and color-values to 32767 on modern hardware. The extension **extended_pair_content** uses **int**s for the color pair and for returning the foreground and background colors, allowing a larger number of colors to be supported.

reset_color_pairs
The extension **reset_color_pairs** tells *ncurses* to discard all of the color pair information which was set with **init_pair**. It also touches the current- and standard-screens, allowing an application to switch color

palettes rapidly.

COLOR_PAIR

COLOR_PAIR(*n*) converts a color pair number to an attribute. Attributes can hold color pairs in the range 0 to 255. If you need a color pair larger than that, you must use functions such as **attr_set** (which pass the color pair as a separate parameter) rather than the legacy functions such as **attrset**.

PAIR_NUMBER

PAIR_NUMBER(*attr*) extracts the color information from its *attr* parameter and returns it as a color pair number; it is the inverse operation of **COLOR_PAIR**.

RETURN VALUE

can_change_color and **has_colors** return **TRUE** or **FALSE**. The other functions return**OK** on success and **ERR** on failure.

In *ncurses*, functions returning an *int* recognize several error conditions.

- All return **ERR** if the screen has not been initialized; see **initscr**(3X) or **newterm**(3X).
- All except **start_color** return **ERR** if **start_color** has not been called, or itself returned **ERR**.
- **start_color** returns **ERR** if it cannot allocate memory for its color pair table.
- **init_color** returns **ERR** if the terminal type does not support assignable color values; that is, if the **initialize_color** (**initc**) capability is absent from its description.
- **init_color** returns **ERR** if any of its *r*, *g*, *b* arguments is outside the range 0-1000 inclusive.
- **init_pair**, **init_color**, **init_extended_pair**, **init_extended_color**, **color_content**, **pair_content**, **extended_color_content**, and **extended_pair_content** return **ERR** on attempts to use
  - color identifiers outside the range 0-**COLORS**−1 inclusive, the default colors extension notwithstanding, or
  - color pair identifiers outside the range 0-**COLOR_PAIRS**−1 inclusive.

NOTES

In *ncurses*, **init_pair** accepts negative foreground and background color arguments to support its **use_default_colors**(3X) extension, but only after the latter function has been called.

The assumption that **COLOR_BLACK** is the terminal's default background color can be overridden using *ncurses*'s **assume_default_colors**(3X) extension.

In *ncurses*, each pointer passed to **color_content** and **pair_content** can be null, in which case the library ignores it, permitting the application to disregard unnecessary information.

In *ncurses*, each screen has a color activation flag, color palette, color pair table, and associated **COLORS** and **COLOR_PAIRS** values; **start_color** affects only the current screen. The SVr4 and X/Open Curses interface was not really designed with this in mind; historical implementations may use a single shared color palette.

Setting an implicit background color via a color pair affects only character cells that a character write operation explicitly touches. To change the background color used when parts of a window are blanked by erasing or scrolling operations, see **curs_bkgd**(3X).

Several caveats apply to IBM PC-compatible machines of the 80486 era and earlier with CGA/EGA/VGA video.

- **COLOR_YELLOW** was frequently converted, in the analog domain, to a shade of brown if the intensity bit was not set. To get yellow on such devices, one would combine **COLOR_YELLOW** with the **A_BOLD** attribute.
- The **A_BLINK** attribute should in theory make the background bright. This often fails to work, and even VGA controllers for which it mostly works, such as those from Paradise and compatibles, do the wrong thing when you try to set a bright "yellow" background — you get a blinking yellow foreground instead.

- Color RGB values are not configurable on these devices (in text mode).

EXTENSIONS

The functions marked as extensions originated in *ncurses*, and are not found in SVr4 *curses*, 4.4BSD *curses*, or any other previous *curses* implementation.

PORTABILITY

Applications employing *ncurses* extensions should condition their use on the visibility of the **NCURSES_VERSION** preprocessor macro.

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

ncurses satisfies X/Open Curses's minimum maximums for *COLORS* and *COLOR_PAIRS*.

X/Open Curses does not specify a limit for the number of colors and color pairs which a terminal can support. However, in its use of *short* for the parameters, it carries over SVr4's implementation detail for the compiled *terminfo* database, which uses signed 16-bit numbers. *ncurses* provides extended versions of the functions using *short* parameters, allowing applications to use larger color and pair identifiers.

SVr4 *curses* returns *ERR* from *pair_content* if its *pair* argument was not initialized using *init_pairs*, and from *color_content* if the terminal does not support changing colors. *ncurses* does neither.

HISTORY

SVr3.2 (1987) introduced color support to *curses* with all of the symbols in the synopsis above except those marked as extensions. It reserved color pair 0 as the terminal's initial, "uncolored" state, and limited the number of possible color pairs to 64, because the color pair datum was encoded in six bits of a *chtype*.

SVr4 made only internal changes, such as moving the storage of color state from the *SCREEN* structure (pointed to by *SP*) to the *TERMINAL* structure (pointed to by *cur_term*).

Other *curses* implementations impose different limits on the number of colors and color pairs.

- *PCCurses* (1987-1990) provided for only 8 colors (and therefore required at most 8×8 = 64 color pairs).
- *PDCurses* (1992-present) inherited the 8-color limitation from *PCCurses*, but changed this to 256 in version 2.5 (2001), and widened its *chtype* from 16 to 32 bits.
- X/Open Curses (1992-present) specified a new structure type, *cchar_t*, to store the character code, attribute flags, and color pair identifier, allowing an increased range of color pairs. It specifies a *short* as storing identifiers for colors and color pairs, limiting portable values to 15 bits; negative values are invalid in System V.
- *ncurses* (1992-present), in its non-wide-character configuration, uses 8 bits of *chtype* for the color pair identifier.

  Version 5.3 (2002) offered a wide-character interface, but encoded the color pair identifier with attributes in the character type.

  Since version 6 (2015), *ncurses* uses a separate *int* for the color pair identifier in a *cchar_t*, introducing extension functions to manage the wider type. When a color pair value fits in 8 bits, *ncurses* permits color pair data to be manipulated via the functions taking *chtype* arguments, even when a *curses* window uses wide-character cells.
- NetBSD *curses* used 6 bits for the color pair identifier from 2000 (when it first added color support) until 2004. At that point, NetBSD widened the color pair identifier to use 9 bits. As of 2025, that size is unchanged. Like *ncurses* before version 6, the NetBSD color pair datum is stored in the attributes field of *cchar_t*, limiting the number of color pairs.

SEE ALSO

curses(3X), **curs_attr**(3X), **curs_initscr**(3X), **curs_variables**(3X), **default_colors**(3X)