## 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** - **curses** color manipulation routines

## SYNOPSIS

```
#include <curses.h>

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);

/* extensions */
void reset_color_pairs(void);

int COLOR_PAIR(int n);
PAIR_NUMBER(attrs);
```

## DESCRIPTION

### Overview

**curses** supports color attributes on terminals with that capability. To use these routines **start_color** must be called, usually right after **initscr**. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the blank field on which the characters are displayed). A programmer initializes a color-pair with the routine **init_pair**. After it has been initialized, **COLOR_PAIR**(*n*) can be used to convert the pair to a video attribute.

If a terminal is capable of redefining colors, the programmer can use the routine **init_color** to change

the definition of a color. The routines **has_colors** and **can_change_color** return **TRUE** or **FALSE**, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine **color_content** allows a programmer to extract the amounts of red, green, and blue components in an initialized color. The routine **pair_content** allows a programmer to find out how a given color-pair is currently defined.

**Color Rendering**

The **curses** library combines these inputs to produce the actual foreground and background colors shown on the screen:

- per-character video attributes (e.g., via **waddch**),
- the window attribute (e.g., by **wattrset**), and
- the background character (e.g., **wbkgdset**).

Per-character and window attributes are usually set by a parameter containing video attributes including a color pair value. Some functions such as **wattr_set** use a separate parameter which is the color pair number.

The background character is a special case: it includes a character value, 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

In **<curses.h>** the following macros are defined. These are the standard colors (ISO-6429). **curses** also assumes that **COLOR_BLACK** is the default background color for all terminals.

**COLOR_BLACK**
**COLOR_RED**
**COLOR_GREEN**
**COLOR_YELLOW**
**COLOR_BLUE**
**COLOR_MAGENTA**
**COLOR_CYAN**
**COLOR_WHITE**

Some terminals support more than the eight (8) "ANSI" colors. There are no standard names for those additional colors.

## 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.

## 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 legal, 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)).
- legal 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**

The **has_colors** routine requires no arguments. It returns **TRUE** if the terminal can manipulate colors; otherwise, it returns **FALSE**. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

**can_change_color**

The **can_change_color** routine requires no arguments. It returns **TRUE** 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 legal 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 legal 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 legal 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 legal 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 legal 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.

**PAIR_NUMBER**

**PAIR_NUMBER(***attrs*) extracts the color value from its *attrs* parameter and returns it as a color pair number.

**COLOR_PAIR**

Its inverse **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**.

## RETURN VALUE

The routines **can_change_color** and **has_colors** return **TRUE** or **FALSE**.

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

X/Open defines no error conditions. SVr4 does document some error conditions which apply in general:

- This implementation will return **ERR** on attempts to use color values outside the range **0** to **COLORS**-1 (except for the default colors extension), or use color pairs outside the range **0** to **COLOR_PAIRS-1**.

  Color values used in **init_color** must be in the range **0** to **1000**.

  An error is returned from all functions if the terminal has not been initialized.

  An error is returned from secondary functions such as **init_pair** if **start_color** was not called.

- SVr4 does much the same, except that it returns **ERR** from **pair_content** if the pair was not initialized using **init_pairs** and it returns **ERR** from **color_content** if the terminal does not support changing colors.

  This implementation does not return **ERR** for either case.

Specific functions make additional checks:

**init_color**
returns an error if the terminal does not support this feature, e.g., if the **initialize_color** capability is absent from the terminal description.

**start_color**
returns an error if the color table cannot be allocated.

## NOTES

In the **ncurses** implementation, there is a separate color activation flag, color palette, color pairs table, and associated **COLORS** and **COLOR_PAIRS** counts for each screen; the **start_color** function only affects the current screen. The SVr4/XSI interface is not really designed with this in mind, and 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 on older x86 machines (e.g., i386, i486) with VGA-compatible graphics:

- COLOR_YELLOW is actually brown. To get yellow, use COLOR_YELLOW combined with the **A_BOLD** attribute.

- The A_BLINK attribute should in theory cause the background to go bright. This often fails to work, and even some cards for which it mostly works (such as the 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 settable.

## PORTABILITY

This implementation satisfies XSI Curses's minimum maximums for **COLORS** and **COLOR_PAIRS**.

The **init_pair** routine accepts negative values of foreground and background color to support the **use_default_colors**(3X) extension, but only if that routine has been first invoked.

The assumption that **COLOR_BLACK** is the default background color for all terminals can be modified using the **assume_default_colors**(3X) extension.

This implementation checks the pointers, e.g., for the values returned by **color_content** and **pair_content**, and will treat those as optional parameters when null.

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. This implementation provides extended versions of those functions which use **short** parameters, allowing applications to use larger color- and pair-numbers.

The **reset_color_pairs** function is an extension of ncurses.

## SEE ALSO

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