NAME

attr_get, **wattr_get**, **attr_set**, **wattr_set**, **attr_off**, **wattr_off**, **attr_on**, **wattr_on**, **attroff**, **wattroff**, **attron**, **wattron**, **attrset**, **wattrset**, **chgat**, **wchgat**, **mvchgat**, **mvwchgat**, **color_set**, **wcolor_set**, **standend**, **wstandend**, **standout**, **wstandout** − manipulate attributes of character cells in *curses* windows

SYNOPSIS

```
#include <curses.h>

int attr_get(attr_t *attrs, short *pair, void *opts);
int wattr_get(WINDOW *win, attr_t *attrs, short *pair, void *opts);
int attr_set(attr_t attrs, short pair, void *opts);
int wattr_set(WINDOW *win, attr_t attrs, short pair, void *opts);

int attr_off(attr_t attrs, void *opts);
int wattr_off(WINDOW *win, attr_t attrs, void *opts);
int attr_on(attr_t attrs, void *opts);
int wattr_on(WINDOW *win, attr_t attrs, void *opts);

int attroff(int attrs);
int wattroff(WINDOW *win, int attrs);
int attron(int attrs);
int wattron(WINDOW *win, int attrs);
int attrset(int attrs);
int wattrset(WINDOW *win, int attrs);

int chgat(int n, attr_t attr, short pair, const void *opts);
int wchgat(WINDOW *win,
    int n, attr_t attr, short pair, const void *opts);
int mvchgat(int y, int x,
    int n, attr_t attr, short pair, const void *opts);
int mvwchgat(WINDOW *win, int y, int x,
    int n, attr_t attr, short pair, const void *opts);

int color_set(short pair, void* opts);
int wcolor_set(WINDOW *win, short pair, void* opts);

int standend(void);
int wstandend(WINDOW *win);
int standout(void);
int wstandout(WINDOW *win);
```

DESCRIPTION

These routines manipulate the current attributes of the named window, which then apply to all characters that are written into the window with **waddch**, **waddstr** and **wprintw**. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible, they are displayed as appropriate modifications to the graphic rendition of characters put on the screen.

These routines do not affect the attributes used when erasing portions of the window. See**curs_bkgd**(3X) for functions which modify the attributes used for erasing and clearing.

Window Attributes

There are two sets of functions:

- functions for manipulating the window attributes and color: **wattr_set** and **wattr_get**.
- functions for manipulating only the window attributes (not color): **wattr_on** and **wattr_off**.

The **wattr_set** function sets the current attributes of the given window to *attrs*, with color specified by *pair*.

Use **wattr_get** to retrieve attributes for the given window.

Use **attr_on** and **wattr_on** to turn on window attributes, i.e., values logically “or”-ed together in *attr*,

without affecting other attributes. Use**attr_off** and **wattr_off** to turn of f window attributes, again values logically "or"-ed together in *attr*, without affecting other attributes.

Legacy Window Attributes

The X/Open window attribute routines which *set* or *get*, turn *on* or *off* are extensions of older routines which assume that color pairs are logically "or"-ed into the attribute parameter. These newer routines use similar names, because X/Open simply added an underscore (_) for the newer names.

The *int* datatype used in the legacy routines is treated as if it is the same size as *chtype* (used by **addch**(3X)). It holds the common video attributes (such as bold, reverse), as well as a few bits for color. Those bits correspond to the **A_COLOR** symbol. The **COLOR_PAIR** macro provides a value which can be logically "or"-ed into the attribute parameter. For example, as long as that value fits into the **A_COLOR** mask, then these calls produce similar results:

```
attrset(A_BOLD | COLOR_PAIR(pair));
attr_set(A_BOLD, pair, NULL);
```

However, if the value does not fit, then the **COLOR_PAIR** macro uses only the bits that fit. For example, because in *ncurses* **A_COLOR** has eight (8) bits, then **COLOR_PAIR(***259***)** is 4 (i.e., 259 is 4 more than the limit 255).

The **PAIR_NUMBER** macro extracts a pair number from an *int* (or *chtype*). For example, the *input* and *output* values in these statements would be the same:

```
int value = A_BOLD | COLOR_PAIR(input);
int output = PAIR_NUMBER(value);
```

The **attrset** routine is a legacy feature predating SVr4 *curses* but kept in X/Open Curses for the same reason that SVr4 *curses* kept it: compatibility.

The remaining **attr*** functions operate exactly like the corresponding **attr_*** functions, except that they take arguments of type *int* rather than *attr_t*.

There is no corresponding **attrget** function as such in X/Open Curses, although *ncurses* provides **getattrs** (see **curs_legacy**(3X)).

Change Character Rendition

The routine **chgat** changes the attributes of a given number of characters starting at the current cursor location of **stdscr**. It does not update the cursor and does not perform wrapping. A character count of −1 or greater than the remaining window width means to change attributes all the way to the end of the current line. The**wchgat** function generalizes this to an y window; the **mvwchgat** function does a cursor move before acting.

In these functions, the color *pair* argument is a color pair index (as in the first argument of **init_pair**, see **curs_color**(3X)).

Change Window Color

The routine **color_set** sets the current color of the given window to the foreground/background combination described by the color *pair* parameter.

Standout

The routine **standout** is the same as **attron(A_STANDOUT)**. The routine**standend** is the same as **attrset(A_NORMAL)** or **attrset(0)**, that is, it turns off all attributes.

X/Open Curses does not mark these "restricted", because

- they have well established legacy use, and
- there is no ambiguity about the way the attributes might be combined with a color pair.

Video Attributes

The following video attributes, defined in *curses.h*, can be passed to **attron**, **attroff**, **attrset**, and logically "or"-ed with characters passed to **addch**(3X).

Name	Description
A_NORMAL	Normal display (no highlight)
A_STANDOUT	Best highlighting mode available
A_UNDERLINE	Underlining
A_REVERSE	Reverse video
A_BLINK	Blinking
A_DIM	Half bright
A_BOLD	Extra bright or bold
A_PROTECT	Protected mode
A_INVIS	Invisible or blank mode
A_ALTCHARSET	Alternate character set
A_ITALIC	Italics (non-X/Open extension)
A_ATTRIBUTES	Mask to extract character code
A_CHARTEXT	Mask to extract attributes
A_COLOR	Mask to extract color pair identifier

attr_on, **attr_off**, and **attr_set** support the foregoing as well as the following additional attributes.

Name	Description
WA_HORIZONTAL	Horizontal highlight
WA_LEFT	Left highlight
WA_LOW	Low highlight
WA_RIGHT	Right highlight
WA_TOP	Top highlight
WA_VERTICAL	Vertical highlight

RETURN VALUE

These functions return **OK** on success and **ERR** on failure.

In *ncurses*, they return **ERR** if *win* is *NULL*.

wcolor_set returns **ERR** if *pair* is outside the range **0**. .**COLOR_PAIRS−1**.

wattr_get does *not* fail if its *attrs* or *pair* parameter is *NULL*.

Functions prefixed with “mv” first perform cursor movement and fail if the position (*y*, *x*) is outside the window boundaries.

NOTES

attr_on, **attr_off**, **attr_set**, **wattr_set**, **chgat**, **mvchgat**, **mvwchgat**, **wchgat**, and **color_set** are part of *ncurses*’s wide-character API, and are not available in its non-wide-character configuration.

attron, **wattron**, **attroff**, **wattroff**, **attrset**, **wattrset**, **standout**, and **standend** may be implemented as macros.

Color pair values may be logically “or”-ed with attributes if the pair number is less than 256. The alternate functions such as **color_set** can pass a color pair value directly. However, *ncurses* ABI 4 and 5 simply logically “or” this value within the alternate functions. You must use *ncurses* ABI 6 to support more than 256 color pairs.

EXTENSIONS

This implementation provides the **A_ITALIC** attribute for terminals which have the **enter_italics_mode** (**sitm**) and **exit_italics_mode** (**ritm**) capabilities. Italics are not mentioned in X/Open Curses. Unlike the other video attributes, **A_ITALIC** is unrelated to the **set_attributes** capabilities. This implementation makes the assumption that **exit_attribute_mode** may also reset italics.

Each of the functions added by XSI Curses has a parameter *opts*, which X/Open Curses still (after more than twenty years) documents as reserved for future use, saying that it should be *NULL*. This implementation uses that parameter in ABI 6 for the functions which have a color pair parameter to support *extended color pairs*:

- For functions which modify the color, e.g., **wattr_set** and **wattr_on**, if *opts* is set it is treated as a pointer to *int*, and used to set the color pair instead of the *short pair* parameter.
- For functions which retrieve the color, e.g., **wattr_get**, if *opts* is set it is treated as a pointer to *int*, and used to retrieve the color pair as an *int* value, in addition to retrieving it via the standard pointer to *short* parameter.
- For functions which turn attributes off, e.g., **wattr_off**, the *opts* parameter is ignored except to check that it is *NULL*.

PORTABILITY

These functions are described in X/Open Curses Issue 4. It specifies no error conditions for them.

The standard defined the dedicated type for highlights, *attr_t*, which was not defined in SVr4 *curses*. The functions taking *attr_t* arguments were not supported under SVr4.

SVr4 describes the functions not taking *attr_t* or *pair* arguments as always returning **1**.

Very old versions of this library did not force an update of the screen when changing the attributes. Use **touchwin** to force the screen to match the updated attributes.

X/Open Curses states that whether the traditional functions **attron**/**attroff**/**attrset** can manipulate attributes other than **A_BLINK**, **A_BOLD**, **A_DIM**, **A_REVERSE**, **A_STANDOUT**, or **A_UNDERLINE** is "unspecified". Under this implementation as well as SVr4*cur ses*, these functions correctly manipulate all other highlights (specifically, **A_ALTCHARSET**, **A_PROTECT**, and **A_INVIS**).

X/Open Curses added these entry points:

attr_get, **attr_on**, **attr_off**, **attr_set**, **wattr_on**, **wattr_off**, **wattr_get**, **wattr_set**

The new functions are intended to work with a new series of highlight macros prefixed with **WA_**. The older macros have direct counterparts in the newer set of names:

Name	Description
WA_NORMAL	Normal display (no highlight)
WA_STANDOUT	Best highlighting mode available
WA_UNDERLINE	Underlining
WA_REVERSE	Reverse video
WA_BLINK	Blinking
WA_DIM	Half bright
WA_BOLD	Extra bright or bold
WA_ALTCHARSET	Alternate character set

X/Open Curses does not assign values to these symbols, nor does it state whether or not they are related to the similarly-named A_NORMAL, etc.:

- X/Open Curses specifies that each pair of corresponding **A_** and **WA_**-using functions operates on the same current-highlight information.
- However, in some implementations, those symbols have unrelated values.

  For example, the Solaris *xpg4* (X/Open) *curses* declares *attr_t* to be an unsigned short integer (16-bits), while *chtype* is a unsigned integer (32-bits). The **WA_** symbols in this case are different from the **A_** symbols because they are used for a smaller datatype which does not represent **A_CHARTEXT** or **A_COLOR**.

  In this implementation (as in many others), the values happen to be the same because it simplifies copying information between *chtype* and *cchar_t* variables.
- Because *ncurses*'s *attr_t* can hold a color pair (in the **A_COLOR** field), a call to **wattr_on**, **wattr_off**, or **wattr_set** may alter the window's color. If the color pair information in the attribute parameter is zero, no change is made to the window's color.

  This is consistent with SVr4 *curses*; X/Open Curses does not specify this.

The X/Open Curses extended conformance level adds new highlights **A_HORIZONTAL**, **A_LEFT**, **A_LOW**, **A_RIGHT**, **A_TOP**, **A_VERTICAL** (and corresponding **WA_** macros for each). As of August 2013, no known terminal provides these highlights (i.e., via the **sgr1** capability).

HISTORY

4BSD (1980) used a *char* to represent each cell of the terminal screen. It assumed 7-bit character codes, employing the eighth bit of a byte to represent a *standout* attribute (often implemented as bold and/or reverse video). It introduced *standout*, *standend*, *wstandout*, and *wstandend* functions to manipulate this bit. Despite their inflexibility, they carried over into System V *curses* and ultimately X/Open Curses due to their pervasive use in legacy applications. While some 1980s terminals supported a variety of video attributes, BSD *curses* could do nothing with them.

SVr2 (1984) provided an improved *curses* library, introducing *chtype* to create the abstract notion of a *curses* character; this was by default an *unsigned short*, with a provision for compile-time redefinition to other integral types (a freedom not necessarily available to users of shared libraries, and in any event a source license was necessary to exercise it). It added the functions *attron*, *attroff*, *attrset*, *wattron*, *wattroff*, and *wattrset*, and defined the *A_* macros listed above (except for *A_ITALIC* and *A_COLOR*) for use by applications to manipulate other attributes. The values of these macros were not necessarily the same in different systems, even among those certified as System V.

SVr3.2 (1988) added the *A_COLOR* macro along with a color system; see **curs_color**(3X).

X/Open Curses Issue 4 (1995) is largely based on SVr4 *curses*, but recognized that the *wchar_t* type of ISO C95 was intended to house only a single character code, not a sequence of codes combining with a base character, let alone could it reliably offer room for a color pair identifier and a set of attribute bits with a potential for further growth — thus the standard invented the *curses* complex character type *cchar_t* and a separate type *attr_t* for storage of attribute bits. The new types brought along several new functions to manipulate them, some corresponding to existing *chtype*-based functions (*attr_on*, *attr_off*, *attr_set*, *wattr_on*, *wattr_off*, and *wattr_set*), and some new (*chgat* and its variants, *color_set*, and *wcolor_set*).

Different Unix systems used differently sized bit fields in *chtype* for the character code and the color pair identifier, and took into account platforms' different integer sizes (32- versus 64-bit).

The following table showing the number of bits for *A_COLOR* and *A_CHARTEXT* was gleaned from the *curses* header files for various operating systems and architectures. The inferred architecture and notes reflect the format and size of the defined constants as well as clues such as the alternate character set implementation. A 32-bit library can be used on a 64-bit system, but not necessarily the converse.

			Bits		
Year	**System**	**Arch**	**Color**	**Char**	**Notes**
1992	Solaris 5.2	32	6	17	SVr4 *curses*
1992	HP-UX 9	32	no	8	SVr2 *curses*
1992	AIX 3.2	32	no	23	SVr2 *curses*
1994	OSF/1 r3	32	no	23	SVr2 *curses*
1995	HP-UX 10.00	32	6	16	SVr3 *curses_colr*
1995	HP-UX 10.00	32	6	8	SVr4, X/Open *curses*
1995	Solaris 5.4	32/64	7	16	X/Open *curses*
1996	AIX 4.2	32	7	16	X/Open *curses*
1996	OSF/1 r4	32	6	16	X/Open *curses*
1997	HP-UX 11.00	32	6	8	X/Open *curses*
2000	UWIN	32/64	7/31	16	uses *chtype*

Notes:

Regarding HP-UX,

- HP-UX 10.20 (1996) added support for 64-bit PA-RISC processors in 1996.
- HP-UX 10.30 (1997) marked "curses_colr" obsolete. That version of *curses* was dropped with HP-UX 11.30 in 2006.

Regarding OSF/1 (and Tru64),

- These used 64-bit hardware. Like *ncurses*, the OSF/1 *curses* interface is not customized for 32-bit and 64-bit versions.
- Unlike other systems which evolved from AT&T code, OSF/1 provided a new implementation for X/Open Curses.

Regarding Solaris,

- The initial release of Solaris was in 1992.
- Its XPG4 (X/Open Curses-conforming) *xcurses* library was developed by Mortice Kern Systems from 1990 to 1995. Sun's copyright began in 1996.
- Sun updated the X/Open Curses interface after 64-bit support was introduced in 1997, but did not modify the SVr4 *curses* interface.

Regarding UWIN,

- Development of the *curses* library began in 1991, stopped in 2000.
- Color support was added in 1998.
- The library uses only *chtype* (not *cchar_t*).

Once X/Open Curses was adopted in the mid-1990s, the constraint of a 32-bit interface with many colors and wide-characters for *chtype* became a moot point. The *cchar_t* structure (whose size and members are not specified in X/Open Curses) could be extended as needed.

Other interfaces are rarely used now.

- BSD *curses* was improved slightly in 1993/1994 using Keith Bostic's modification to make the library 8-bit clean for *nvi*(1). He moved the *standout* attribute to a structure member. The resulting 4.4BSD *curses* was replaced by *ncurses* over the next ten years.
- UWIN has been defunct since 2012.

ncurses 6.0 (2015) added the *A_ITALIC* macro.

SEE ALSO

curses(3X), **curs_addch**(3X), **curs_addstr**(3X), **curs_bkgd**(3X), **curs_printw**(3X), **curs_variables**(3X)