NAME

add_wch, **wadd_wch**, **mvadd_wch**, **mvwadd_wch**, **echo_wchar**, **wecho_wchar** − add a *curses* complex character to a window, possibly advancing the cursor

SYNOPSIS

#include <curses.h>

int add_wch(const cchar_t * *wch***);**
int wadd_wch(WINDOW * *win***, const cchar_t *** *wch***);**
int mvadd_wch(int *y***, int** *x***, const cchar_t *** *wch***);**
int mvwadd_wch(WINDOW * *win***, int** *y***, int** *x***, const cchar_t *** *wch***);**

int echo_wchar(const cchar_t * *wch***);**
int wecho_wchar(WINDOW * *win***, const cchar_t ***_wch_**);**

/ (integer) constants */*
/* . . . */ **WACS_BLOCK;**
/* . . . */ **WACS_BOARD;**
/* . . . */ **WACS_BTEE;**
/* . . . */ **WACS_BULLET;**
/* . . . */ **WACS_CKBOARD;**
/* . . . */ **WACS_DARROW;**
/* . . . */ **WACS_DEGREE;**
/* . . . */ **WACS_DIAMOND;**
/* . . . */ **WACS_HLINE;**
/* . . . */ **WACS_LANTERN;**
/* . . . */ **WACS_LARROW;**
/* . . . */ **WACS_LLCORNER;**
/* . . . */ **WACS_LRCORNER;**
/* . . . */ **WACS_LTEE;**
/* . . . */ **WACS_PLMINUS;**
/* . . . */ **WACS_PLUS;**
/* . . . */ **WACS_RARROW;**
/* . . . */ **WACS_RTEE;**
/* . . . */ **WACS_S1;**
/* . . . */ **WACS_S9;**
/* . . . */ **WACS_TTEE;**
/* . . . */ **WACS_UARROW;**
/* . . . */ **WACS_ULCORNER;**
/* . . . */ **WACS_URCORNER;**
/* . . . */ **WACS_VLINE;**
/ extensions */*
/* . . . */ **WACS_GEQUAL;**
/* . . . */ **WACS_LEQUAL;**
/* . . . */ **WACS_NEQUAL;**
/* . . . */ **WACS_PI;**
/* . . . */ **WACS_S3;**
/* . . . */ **WACS_S7;**
/* . . . */ **WACS_STERLING;**
/ extensions for thick lines */*
/* . . . */ **WACS_T_BTEE;**
/* . . . */ **WACS_T_HLINE;**
/* . . . */ **WACS_T_LLCORNER;**
/* . . . */ **WACS_T_LRCORNER;**
/* . . . */ **WACS_T_LTEE;**
/* . . . */ **WACS_T_PLUS;**
/* . . . */ **WACS_T_RTEE;**

/* . . . */ **WACS_T_TTEE;**
/* . . . */ **WACS_T_ULCORNER;**
/* . . . */ **WACS_T_URCORNER;**
/* . . . */ **WACS_T_VLINE;**
/ extensions for double lines */*
/* . . . */ **WACS_D_BTEE;**
/* . . . */ **WACS_D_HLINE;**
/* . . . */ **WACS_D_LLCORNER;**
/* . . . */ **WACS_D_LRCORNER;**
/* . . . */ **WACS_D_LTEE;**
/* . . . */ **WACS_D_PLUS;**
/* . . . */ **WACS_D_RTEE;**
/* . . . */ **WACS_D_TTEE;**
/* . . . */ **WACS_D_ULCORNER;**
/* . . . */ **WACS_D_URCORNER;**
/* . . . */ **WACS_D_VLINE;**

DESCRIPTION

wadd_wch

wadd_wch writes the *curses* complex character *wch* to the window *win*, then may advance the cursor position, analogously to the standard C library's *putwchar*(3). **ncurses**(3X) describes the variants of this function.

Construct a *curses* complex character from a *wchar_t* with **setcchar**(3X). A*cc har_t* can be copied from place to place using **win_wch**(3X) and **wadd_wch**. *curses* defines constants to aid the manipulation of character attributes; see **curs_attr**(3X). A complex character whose only character component is a wide space, and whose only attribute is **WA_NORMAL**, is a *blank character*, and therefore combines with the window's background character; see **curs_bkgrnd**(3X).

Much behavior depends on whether the wide characters in *wch* are spacing or non-spacing; see subsection "Complex Characters" below.

- If *wch* contains a spacing character, then any character at the cursor is first removed. The complex character *wch*, with its attributes and color pair identifier, becomes the *base* of the *active complex character*.
- If *wch* contains only non-spacing characters, they are combined with the active complex character. *curses* ignores its attributes and color pair identifier, and does not advance the cursor.

Further non-spacing characters added with **wadd_wch** are not written at the new cursor position but combine with the active complex character until another spacing character is written to the window or the cursor is moved.

If *wch* is a backspace, carriage return, line feed, or tab, the cursor moves appropriately within the window.

- Backspace moves the cursor one character left; at the left margin of a window, it does nothing.
- Carriage return moves the cursor to the left margin on the same line of the window.
- Line feed does a **clrtoeol**(3X), then advances as if from the right margin.
- Tab advances the cursor to the next tab stop (possibly on the next line); these are placed at every eighth column by default.

  Alter the tab interval with the **TABSIZE** extension; see **curs_variables**(3X).

If *wch* is any other nonprintable character, *curses* draws it in printable form using the same convention as **wunctrl(**3X). Calling**win_wch**(3X) on the location of a nonprintable character does not retrie ve the character itself, but its **wunctrl(**3X) representation.

Adding spacing characters with **wadd_wch** causes it to wrap at the right margin of the window:

- If the cursor is not at the bottom of the scrolling region and advancement occurs at the right margin, the cursor automatically wraps to the beginning of the next line.
- If the cursor is at the bottom of the scrolling region when advancement occurs at the right margin, and **scrollok**(3X) is enabled for *win*, the scrolling region scrolls up one line and the cursor wraps as above. Otherwise, advancement and scrolling do not occur, and **wadd_wch** returns **ERR**.

A window's margins may coincide with the screen boundaries. This may be a problem when *ncurses* updates the screen to match the curses window. When their right and bottom margins coincide, *ncurses* uses different strategies to handle the variations of scrolling and wrapping at the lower-right corner by depending on the terminal capabilities:

- If the terminal does not automatically wrap as characters are added at the right margin (i.e., auto right margins), *ncurses* writes the character directly.
- If the terminal has auto right margins, but also has capabilities for turning auto margins off and on, *ncurses* turns the auto margin feature off temporarily when writing to the lower-right corner.
- If the terminal has an insertion mode which can be turned off and on, *ncurses* writes the character just before the lower-right corner, and then inserts a character to push the update into the corner.

wecho_wchar

echo_wchar and **wecho_wchar** are equivalent to calling (**w**)**add_wch** followed by (**w**)**refresh** on **stdscr** or the specified window. *curses* interprets these functions as a hint that only a single (complex) character is being output; for non-control characters, a considerable performance gain may be enjoyed by employing them.

Forms-Drawing Characters

curses defines macros starting with **WACS_** that can be used with **wadd_wch** to write line-drawing and other symbols to the screen. *ncur ses* terms these *forms-drawing characters. curses* uses the ACS default listed below if the terminal type lacks the **acs_chars** (**acsc**) capability; that capability does not define a replacement for the character; or if the terminal type and locale configuration require Unicode to access these characters, but the library is unable to use Unicode. The "acsc char" column corresponds to how the characters are specified in the **acs_chars** (**acsc**) string capability, and the characters in it may appear on the screen if the terminal type's database entry incorrectly advertises ACS support. The name "ACS" originates in the Alternate Character Set feature of the DEC VT100 terminal.

Symbol	Unicode Default	ACS Default	acsc char	Glyph Name
WACS_BLOCK	U+25ae	#	0	solid square block
WACS_BOARD	U+2592	#	h	board of squares
WACS_BTEE	U+2534	+	v	bottom tee
WACS_BULLET	U+00b7	o	~	bullet
WACS_CKBOARD	U+2592	:	a	checker board (stipple)
WACS_DARROW	U+2193	v	.	arrow pointing down
WACS_DEGREE	U+00b0	'	f	degree symbol
WACS_DIAMOND	U+25c6	+	`	diamond
WACS_GEQUAL	U+2265	>	>	greater-than-or-equal-to
WACS_HLINE	U+2500	−	q	horizontal line
WACS_LANTERN	U+2603	#	i	lantern symbol
WACS_LARROW	U+2190	<	,	arrow pointing left
WACS_LEQUAL	U+2264	<	y	less-than-or-equal-to
WACS_LLCORNER	U+2514	+	m	lower left-hand corner
WACS_LRCORNER	U+2518	+	j	lower right-hand corner
WACS_LTEE	U+2524	+	t	left tee
WACS_NEQUAL	U+2260	!	|	not-equal
WACS_PI	U+03c0	*	{	greek pi
WACS_PLMINUS	U+00b1	#	g	plus/minus

WACS_PLUS	U+253c	+	n	plus
WACS_RARROW	U+2192	>	+	arrow pointing right
WACS_RTEE	U+251c	+	u	right tee
WACS_S1	U+23ba	−	o	scan line 1
WACS_S3	U+23bb	−	p	scan line 3
WACS_S7	U+23bc	−	r	scan line 7
WACS_S9	U+23bd	_	s	scan line 9
WACS_STERLING	U+00a3	f	}	pound-sterling symbol
WACS_TTEE	U+252c	+	w	top tee
WACS_UARROW	U+2191	^	−	arrow pointing up
WACS_ULCORNER	U+250c	+	l	upper left-hand corner
WACS_URCORNER	U+2510	+	k	upper right-hand corner
WACS_VLINE	U+2502	|	x	vertical line

The *ncurses* wide API also defines symbols for thick lines (**acsc** “J” through “N”, “T” through “X”, and “Q”):

ACS Name	Unicode Default	ASCII Default	acsc Char	Glyph Name
WACS_T_BTEE	U+253b	+	V	thick tee pointing up
WACS_T_HLINE	U+2501	-	Q	thick horizontal line
WACS_T_LLCORNER	U+2517	+	M	thick lower left corner
WACS_T_LRCORNER	U+251b	+	J	thick lower right corner
WACS_T_LTEE	U+252b	+	T	thick tee pointing right
WACS_T_PLUS	U+254b	+	N	thick large plus
WACS_T_RTEE	U+2523	+	U	thick tee pointing left
WACS_T_TTEE	U+2533	+	W	thick tee pointing down
WACS_T_ULCORNER	U+250f	+	L	thick upper left corner
WACS_T_URCORNER	U+2513	+	K	thick upper right corner
WACS_T_VLINE	U+2503	|	X	thick vertical line

and for double lines (**acsc** “A” through “I”, plus “R” and “Y”):

ACS Name	Unicode Default	ASCII Default	acsc Char	Glyph Name
WACS_D_BTEE	U+2569	+	H	double tee pointing up
WACS_D_HLINE	U+2550	-	R	double horizontal line
WACS_D_LLCORNER	U+255a	+	D	double lower left corner
WACS_D_LRCORNER	U+255d	+	A	double lower right corner
WACS_D_LTEE	U+2560	+	F	double tee pointing right
WACS_D_PLUS	U+256c	+	E	double large plus
WACS_D_RTEE	U+2563	+	G	double tee pointing left
WACS_D_TTEE	U+2566	+	I	double tee pointing down
WACS_D_ULCORNER	U+2554	+	C	double upper left corner
WACS_D_URCORNER	U+2557	+	B	double upper right corner
WACS_D_VLINE	U+2551	|	Y	double vertical line

Unicode’s descriptions for these characters differs slightly from *ncurses*, by introducing the term “light” (along with less important details). Here are its descriptions for the normal, thick, and double horizontal lines:

- U+2500 BOX DRAWINGS LIGHT HORIZONTAL
- U+2501 BOX DRAWINGS HEAVY HORIZONTAL
- U+2550 BOX DRAWINGS DOUBLE HORIZONTAL

RETURN VALUE

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

In *ncurses*, these functions fail if

- the *curses* screen has not been initialized,
- (for functions taking a *WINDOW* pointer argument) *win* is a null pointer,
- wrapping to a new line is impossible because **scrollok**(3X) has not been called on *win* (or **stdscr**, as applicable) when writing to its bottom right location is attempted, or
- it is not possible to add a complete character at the cursor position.

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

NOTES

add_wch, **mvadd_wch**, **mvwadd_wch**, and **echo_wchar** may be implemented as macros.

EXTENSIONS

The symbols *WACS_S3*, *WACS_S7*, *WACS_LEQUAL*, *WACS_GEQUAL*, *WACS_PI*, *WACS_NEQUAL*, and *WACS_STERLING* are not standard. However, many publicly available *terminfo* entries include **acs_chars** (**acsc**) capabilities in which their key characters (**pryz{|}**) are embedded, and a second-hand list of their character descriptions has come to light. The *ncurses* developers invented WACS-prefixed names for them.

PORTABILITY

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

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

The defaults specified for forms-drawing characters apply in the POSIX locale. X/Open Curses makes it clear that the WACS_ symbols should be defined as a pointer to *cchar_t* data, e.g., in the discussion of *border_set*. A few implementations are problematic:

- NetBSD *curses* defines the symbols as a *wchar_t* within a *cchar_t*.
- HP-UX *curses* equates some of the *ACS_* symbols to the analogous *WACS_* symbols as if the *ACS_* symbols were wide characters. The misdefined symbols are the arrows and other symbols which are not used for line-drawing.

X/Open Curses does not specify symbols for thick- or double-lines. SVr4 *curses* implementations defined their line-drawing symbols in terms of intermediate symbols. *ncurses* extends those symbols, providing new definitions not found in SVr4 implementations.

Not all Unicode-capable terminals provide support for VT100-style alternate character sets (i.e., the *acsc_chars* (**acsc**) capability), with their corresponding line-drawing characters. X/Open Curses did not address the aspect of integrating Unicode with line-drawing characters. Existing implementations of System V *curses* (AIX, HP-UX, Solaris) use only the *acsc_chars* (**acsc**) character-mapping to provide this feature. As a result, those implementations can use only single-byte line-drawing characters. *ncurses* 5.3 (2002) provided a table of Unicode values to solve these problems. NetBSD *curses* incorporated that table in 2010.

ncurses uses the Unicode values instead of the terminal type description's *acsc_chars* (**acsc**) mapping as discussed in **ncurses**(3X) for the environment variable *NCURSES_NO_UTF8_ACS*. In contrast, for the same cases, the line-drawing characters described in **addch**(3X) will use only the ASCII default values.

Having Unicode available does not solve all of the problems with line-drawing for *curses*:

- The closest Unicode equivalents to the VT100 graphics *S1*, *S3*, *S7*, and *S9* frequently are not displayed at the regular intervals which the terminal used.
- The *lantern* is a special case. It originated with the AT&T 4410 terminal in the early 1980s. There is no accessible documentation depicting the lantern symbol on the AT&T terminal.

  Lacking documentation, most readers assume that a *storm lantern* was intended. But there are several possibilities, all with problems.

Unicode 6.0 (2010) does provide two lantern symbols: U+1F383 and U+1F3EE. Those were not available in 2002, and are irrelevant since they lie outside the Basic Multilingual Plane and as a result are unavailable on many terminals. They are not storm lanterns, in any case.

Most *storm lanterns* have a tapering glass chimney (to guard against tipping); some have a wire grid protecting the chimney.

For the tapering appearance, U+2603 was adequate. In use on a terminal, no one can tell what the image represents. Unicode calls it a snowman.

Others have suggested these alternatives: § U+00A7 (section mark), Θ U+0398 (theta), Φ U+03A6 (phi), δ U+03B4 (delta), U+2327 (x in a rectangle), U+256C (forms double vertical and horizontal), and U+2612 (ballot box with x).

Complex Characters

The complex character type *cchar_t* can store more than one wide character (*wchar_t*). X/Open Curses does not mention this possibility, specifying behavior only where *wch* is a single character, either spacing or non-spacing.

ncurses assumes that *wch* is constructed using **setcchar**(3X), and in turn that the result

- contains at most one spacing character at the beginning of its list of wide characters, and zero or more non-spacing characters, or
- holds one non-spacing character.

In the latter case, *ncurses* adds the non-spacing character to the active complex character.

HISTORY

X/Open Curses Issue 4 (1995) initially specified these functions. The System V Interface Definition (SVID) Version 4 of the same year specified functions named *waddwch* (and the usual variants), *echowchar*, and *wechowchar*. These were later additions to SVr4.*x*, not appearing in the first SVr4 (1989). They differed from X/Open's later *wadd_wch* and *wecho_wchar* in that they each took an argument of type *wchar_t* instead of *cchar_t*. SVID defined no *WACS_* symbols.

X/Open Curses Issue 4 also defined many of the *WACS_* constants, excepting *WACS_GEQUAL*, *WACS_LEQUAL*, *WACS_NEQUAL*, *WACS_PI*, *WACS_S3*, *WACS_S7*, and *WACS_STERLING*; and those for drawing thick and double lines.

ncurses 5.3 (2002) furnished the remaining *WACS_* constants.

SEE ALSO

curs_addch(3X) describes comparable functions of the *ncurses* library in its non-wide-character configuration.

curses(3X), **curs_addwstr**(3X), **curs_add_wchstr**(3X), **curs_attr**(3X), **curs_bkgrnd**(3X), **curs_clear**(3X), **curs_getcchar**(3X), **curs_outopts**(3X), **curs_refresh**(3X), **curs_variables**(3X), **putwc**(3)