NAME

add_wch, **wadd_wch**, **mvadd_wch**, **mvwadd_wch**, **echo_wchar**, **wecho_wchar** - add a *curses* complex character to a window and advance 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);
```

DESCRIPTION

add_wch

The **add_wch**, **wadd_wch**, **mvadd_wch**, and **mvwadd_wch** functions put the complex character *wch* into the given window at its current position, which is then advanced. These functions perform wrapping and special-character processing as follows:

- If *wch* refers to a spacing character, then any previous character at that location is removed. A new character specified by *wch* is placed at that location with rendition specified by *wch*. The cursor then advances after this spacing character, to prepare for writing the next character on the screen.

  The newly added spacing character is the base of the active complex character. Subsequent non-spacing characters can be combined with this base until another spacing character is written to the screen, or the cursor is moved, e.g., using **wmove**.

- If *wch* refers to a non-spacing character, it is appended to the active complex character, retaining the previous characters at that location. The rendition specified by *wch* is ignored.

  The cursor is not advanced after adding a non-spacing character. Subsequent calls to add non-spacing characters will update the same position.

- If the character part of *wch* is a tab, newline, backspace or other control character, the window is updated and the cursor moves as if **addch** were called.

echo_wchar

The **echo_wchar** function is functionally equivalent to a call to **add_wch** followed by a call to **refresh**(3X). Similarly, the **wecho_wchar** is functionally equivalent to a call to **wadd_wch** followed by a call to **wrefresh**. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using the *__echo__* functions instead of their equivalents.

Line Graphics

Like **addch**(3X), **addch_wch** accepts symbols which make it simple to draw lines and other frequently used special characters. These symbols correspond to the same VT100 line-drawing set as **addch**(3X).

ACS Name	Unicode Default	ASCII Default	acsc Char	Glyph Name
WACS_BLOCK	0x25ae	#	0	solid square block
WACS_BOARD	0x2592	#	h	board of squares
WACS_BTEE	0x2534	+	v	bottom tee
WACS_BULLET	0x00b7	o	~	bullet
WACS_CKBOARD	0x2592	:	a	checker board (stipple)
WACS_DARROW	0x2193	v	.	arrow pointing down
WACS_DEGREE	0x00b0	’	f	degree symbol
WACS_DIAMOND	0x25c6	+	‘	diamond
WACS_GEQUAL	0x2265	>	>	greater-than-or-equal-to
WACS_HLINE	0x2500	-	q	horizontal line
WACS_LANTERN	0x2603	#	i	lantern symbol
WACS_LARROW	0x2190	<	,	arrow pointing left
WACS_LEQUAL	0x2264	<	y	less-than-or-equal-to
WACS_LLCORNER	0x2514	+	m	lower left-hand corner
WACS_LRCORNER	0x2518	+	j	lower right-hand corner

WACS_LTEE	0x2524	+	t	left tee
WACS_NEQUAL	0x2260	!	|	not-equal
WACS_PI	0x03c0	*	{	greek pi
WACS_PLMINUS	0x00b1	#	g	plus/minus
WACS_PLUS	0x253c	+	n	plus
WACS_RARROW	0x2192	>	+	arrow pointing right
WACS_RTEE	0x251c	+	u	right tee
WACS_S1	0x23ba	-	o	scan line 1
WACS_S3	0x23bb	-	p	scan line 3
WACS_S7	0x23bc	-	r	scan line 7
WACS_S9	0x23bd	_	s	scan line 9
WACS_STERLING	0x00a3	f	}	pound-sterling symbol
WACS_TTEE	0x252c	+	w	top tee
WACS_UARROW	0x2191	^	-	arrow pointing up
WACS_ULCORNER	0x250c	+	l	upper left-hand corner
WACS_URCORNER	0x2510	+	k	upper right-hand corner
WACS_VLINE	0x2502	|	x	vertical line

The wide-character configuration of *ncurses* also defines symbols for thick lines (**acsc** "J" to "V"):

ACS Name	**Unicode Default**	**ASCII Default**	**acsc Char**	**Glyph Name**
WACS_T_BTEE	0x253b	+	V	thick tee pointing up

WACS_T_HLINE	0x2501	-	Q	thick horizontal line
WACS_T_LLCORNER	0x2517	+	M	thick lower left corner
WACS_T_LRCORNER	0x251b	+	J	thick lower right corner
WACS_T_LTEE	0x252b	+	T	thick tee pointing right
WACS_T_PLUS	0x254b	+	N	thick large plus
WACS_T_RTEE	0x2523	+	U	thick tee pointing left
WACS_T_TTEE	0x2533	+	W	thick tee pointing down
WACS_T_ULCORNER	0x250f	+	L	thick upper left corner
WACS_T_URCORNER	0x2513	+	K	thick upper right corner
WACS_T_VLINE	0x2503	|	X	thick vertical line

and for double-lines (**acsc** "A" to "I"):

ACS Name	**Unicode Default**	**ASCII Default**	**acsc Char**	**Glyph Name**
WACS_D_BTEE	0x2569	+	H	double tee pointing up
WACS_D_HLINE	0x2550	-	R	double horizontal line
WACS_D_LLCORNER	0x255a	+	D	double lower left corner
WACS_D_LRCORNER	0x255d	+	A	double lower right corner
WACS_D_LTEE	0x2560	+	F	double tee pointing right
WACS_D_PLUS	0x256c	+	E	double large plus
WACS_D_RTEE	0x2563	+	G	double tee pointing

			left
WACS_D_TTEE	0x2566	+	I double tee pointing down
WACS_D_ULCORNER	0x2554	+	C double upper left corner
WACS_D_URCORNER	0x2557	+	B double upper right corner
WACS_D_VLINE	0x2551	|	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

All routines return the integer **ERR** upon failure and **OK** on success.

X/Open Curses does not specify any error conditions. This implementation returns an error

- if the window pointer is null or
- if it is not possible to add a complete character in the window.

The latter may be due to different causes:

- If **scrollok**(3X) is not enabled, writing a character at the lower right margin succeeds. However, an error is returned because it is not possible to wrap to a new line.
- If an error is detected when converting a multibyte character to a sequence of bytes, or if it is not possible to add all of the resulting bytes in the window, an error is returned.

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

NOTES

Note that **add_wch**, **mvadd_wch**, **mvwadd_wch**, and **echo_wchar** may be macros.

PORTABILITY

These functions are described in X/Open Curses, Issue 4. The defaults specified for line-drawing characters apply in the POSIX locale.

WACS Symbols

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. This implementation extends those symbols, providing new definitions which are not in the SVr4 implementations.

Not all Unicode-capable terminals provide support for VT100-style alternate character sets (i.e., the **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 Unix curses (AIX, HP-UX, Solaris) use only the **acsc** character-mapping to provide this feature. As a result, those implementations can only use 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.

In this implementation, the Unicode values are used instead of the terminal description's **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 BMP and as a result are not generally available in 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: <section> U+00A7 (section mark), <Theta> U+0398 (theta), <Phi> U+03A6 (phi), <delta> 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**). The X/Open Curses description does not mention this possibility, describing only the cases where *wch* is a spacing character or a non-spacing character.

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

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

In the latter case, *ncurses* adds the non-spacing character to the active (base) spacing character.

TABSIZE

The **TABSIZE** variable is implemented in SVr4 and other versions of *curses*, but is not specified by X/Open Curses (see **curs_variables**(3X)).

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_clear**(3X),

curs_getcchar(3X), **curs_outopts**(3X), **curs_refresh**(3X), **curs_variables**(3X), **putwc**(3)