NAME

addch, **waddch**, **mvaddch**, **mvwaddch**, **echochar**, **wechochar** - add a *curses* character to a window and advance the cursor

SYNOPSIS

```
#include <curses.h>

int addch(const chtype ch);
int waddch(WINDOW *win, const chtype ch);
int mvaddch(int y, int x, const chtype ch);
int mvwaddch(WINDOW *win, int y, int x, const chtype ch);

int echochar(const chtype ch);
int wechochar(WINDOW *win, const chtype ch);
```

DESCRIPTION

Adding Characters

waddch puts the character *ch* at the cursor position of window *win*, then advances the cursor position, analogously to the standard C library's *putchar*(3). **ncurses**(3X) describes the variants of this function.

If advancement occurs at the right margin,

- the cursor automatically wraps to the beginning of the next line; and
- at the bottom of the current scrolling region, and if **scrollok**(3X) is enabled for *win*, the scrolling region scrolls up one line.

If *ch* 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 current line of the window.
- Line feed does a **clrtoeol**(3X), then moves the cursor to the left margin on the next line of the window, and if **scrollok**(3X) is enabled for *win*, scrolls the window if the cursor was already on the last line.
- 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 *ch* is any other nonprintable character, it is drawn in printable form, using the same convention as **unctrl**(3X).

Calling **winch**(3X) on the location of a nonprintable character does not return the character itself, but its **unctrl**(3X) representation.

ch may contain rendering and/or color attributes, and others can be combined with the parameter by logically "or"ing with it. (A character with its attributes can be copied from place to place using **winch**(3X) and **waddch**.) See **curs_attr**(3X) for values of predefined video attribute constants that can be usefully "or"ed with characters.

Echoing Characters

echochar and **wechochar** are equivalent to calling (**w**)**addch** followed by (**w**)**refresh**. *curses* interprets these functions as a hint that only a single 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 **ACS_** that can be used with **waddch** to write line-drawing and other special characters to the screen. *ncurses* terms these *forms-drawing characters.* The ACS default listed below is used if the **acs_chars** (**acsc**) *terminfo* capability does not define a terminal-specific replacement for it, or if the terminal and locale configuration requires 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** string capability, and the characters in it may appear on the screen if the terminal's database entry incorrectly advertises ACS support. The name "ACS" originates in the Alternate Character Set feature of the DEC VT100 terminal.

Symbol	ACS Default	acsc char	Glyph Name
ACS_BLOCK	#	0	solid square block
ACS_BOARD	#	h	board of squares
ACS_BTEE	+	v	bottom tee
ACS_BULLET	o	~	bullet
ACS_CKBOARD	:	a	checker board

			(stipple)
ACS_DARROW	v	.	arrow pointing down
ACS_DEGREE	’	f	degree symbol
ACS_DIAMOND	+	‘	diamond
ACS_GEQUAL	>	>	greater-than-or-equal-to
ACS_HLINE	-	q	horizontal line
ACS_LANTERN	#	i	lantern symbol
ACS_LARROW	<	,	arrow pointing left
ACS_LEQUAL	<	y	less-than-or-equal-to
ACS_LLCORNER	+	m	lower left-hand corner
ACS_LRCORNER	+	j	lower right-hand corner
ACS_LTEE	+	t	left tee
ACS_NEQUAL	!	|	not-equal
ACS_PI	*	{	greek pi
ACS_PLMINUS	#	g	plus/minus
ACS_PLUS	+	n	plus
ACS_RARROW	>	+	arrow pointing right
ACS_RTEE	+	u	right tee
ACS_S1	-	o	scan line 1
ACS_S3	-	p	scan line 3
ACS_S7	-	r	scan line 7
ACS_S9	_	s	scan line 9
ACS_STERLING	f	}	pound-sterling symbol
ACS_TTEE	+	w	top

			tee
ACS_UARROW	^	-	arrow pointing up
ACS_ULCORNER	+	l	upper left-hand corner
ACS_URCORNER	+	k	upper right-hand corner
ACS_VLINE	|	x	vertical line

RETURN VALUE

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

In *ncurses*, **waddch** returns **ERR** if it is not possible to add a complete character at the cursor position, as when conversion of a multibyte character to a byte sequence fails, or at least one of the resulting bytes cannot be added to the window. See section "PORTABILITY" below regarding the use of **waddch** with multibyte characters.

waddch can successfully write a character at the bottom right location of the window. However, *ncurses* returns **ERR** if **scrollok**(3X) is not enabled in that event, because it is not possible to wrap to a new line.

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

NOTES

addch, **mvaddch**, **mvwaddch**, and **echochar** may be implemented as macros.

PORTABILITY

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

SVr4 *curses* describes a successful return value only as "an integer value other than **ERR**".

The defaults specified for forms-drawing characters apply in the POSIX locale.

ACS Symbols

X/Open Curses states that the **ACS_** definitions are *char* constants.

Some implementations are problematic.

- Solaris *curses*, for example, define the ACS symbols as constants; others define them as elements of an array.

  This implementation uses an array, **acs_map**, as did SVr4 *curses*. NetBSD also uses an array, actually named **_acs_char**, with a **#define** for compatibility.

- HP-UX *curses* equates some of the **ACS_** symbols to the analogous **WACS_** symbols as if the **ACS_** symbols were wide characters (see **curs_add_wch**(3X)). The misdefined symbols are the arrows and others that are not used for line drawing.

- X/Open Curses (Issues 2 through 7) has a typographical error for the **ACS_LANTERN** symbol, equating its "VT100+ Character" to "I" (capital I), while the header files for SVr4 *curses* and other implementations use "i" (small i).

  None of the terminal descriptions on Unix platforms use uppercase I, except for Solaris (in its *terminfo* entry for *screen*(1), apparently based on the X/Open documentation around 1995). On the other hand, its **gs6300** (AT&T PC6300 with EMOTS Terminal Emulator) description uses lowercase i.

Some ACS symbols (**ACS_S3**, **ACS_S7**, **ACS_LEQUAL**, **ACS_GEQUAL**, **ACS_PI**, **ACS_NEQUAL**, and **ACS_STERLING**) were not documented in any publicly released System V. However, many publicly available *terminfo* entries include **acsc** strings 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 ACS-prefixed names for them.

The *displayed* values of **ACS_** constants depend on

- the *ncurses* ABI--for example, wide-character versus non-wide-character configurations (the former is capable of displaying Unicode while the latter is not), and

- whether the locale uses UTF-8 encoding.

In certain cases, the terminal is unable to display forms-drawing characters *except* by using UTF-8; see the discussion of the *NCURSES_NO_UTF8_ACS* environment variable in **ncurses**(3X)).

Character Set

X/Open Curses assumes that the parameter passed to **waddch** contains a single character. As discussed in **curs_attr**(3X), that character may have been more than eight bits wide in an SVr3 or SVr4 implementation, but in the X/Open Curses model, the details are not given. The important distinction between SVr4 *curses* and X/Open Curses is that the latter separates non-character information

(attributes and color) from the character code, which SVr4 packs into a *chtype* for passage to **waddch**.

In *ncurses*, *chtype* holds an eight-bit character. But the library allows a multibyte character to be passed in a succession of calls to **waddch**. Other implementations do not; a **waddch** call transmits exactly one character, which may be rendered in one or more screen locations depending on whether it is printable.

Depending on the locale settings, *ncurses* inspects the byte passed in each **waddch** call, and checks whether the latest call continues a multibyte sequence. When a character is *complete*, *ncurses* displays the character and advances the cursor.

If the calling application interrupts the succession of bytes in a multibyte character sequence by changing the current location--for example, with **wmove**(3X)--*ncurses* discards the incomplete character.

For portability to other implementations, do not rely upon this behavior. Check whether a character can be represented as a single byte in the current locale.

- If it can, call either **waddch** or **wadd_wch**(3X).

- If it cannot, use only **wadd_wch**(3X).

TABSIZE

SVr4 and other versions of *curses* implement the **TABSIZE** variable, but X/Open Curses does not specify it (see **curs_variables**(3X)).

SEE ALSO

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

curses(3X), **curs_addchstr**(3X), **curs_addstr**(3X), **curs_attr**(3X), **curs_clear**(3X), **curs_inch**(3X), **curs_outopts**(3X), **curs_refresh**(3X), **curs_variables**(3X), **putchar**(3)