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);

/* (integer) constants */
/* . . . */ ACS_BLOCK;
/* . . . */ ACS_BOARD;
/* . . . */ ACS_BTEE;
/* . . . */ ACS_BULLET;
/* . . . */ ACS_CKBOARD;
/* . . . */ ACS_DARROW;
/* . . . */ ACS_DEGREE;
/* . . . */ ACS_DIAMOND;
/* . . . */ ACS_HLINE;
/* . . . */ ACS_LANTERN;
/* . . . */ ACS_LARROW;
/* . . . */ ACS_LLCORNER;
/* . . . */ ACS_LRCORNER;
/* . . . */ ACS_LTEE;
/* . . . */ ACS_PLMINUS;
/* . . . */ ACS_PLUS;
/* . . . */ ACS_RARROW;
/* . . . */ ACS_RTEE;
/* . . . */ ACS_S1;
/* . . . */ ACS_S9;
/* . . . */ ACS_TTEE;
/* . . . */ ACS_UARROW;
/* . . . */ ACS_ULCORNER;
/* . . . */ ACS_URCORNER;
/* . . . */ ACS_VLINE;
/* extensions */
/* . . . */ ACS_GEQUAL;
/* . . . */ ACS_LEQUAL;
/* . . . */ ACS_NEQUAL;
/* . . . */ ACS_PI;
/* . . . */ ACS_S3;
/* . . . */ ACS_S7;
/* . . . */ ACS_STERLING;
```

DESCRIPTION

waddch

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

Construct a *curses* character from a *char* by assignment or typecast. Subsection "Video Attributes" of **attron**(3X) describes how to manipulate its attributes and color pair. (A color pair selection is not honored unless initialized; see **start_color**(3X).)

The object or expression *ch* may contain attributes and/or a color pair identifier. (A*c htype* can be copied from place to place using **winch**(3X) and **waddch**.) *curses* defines constants to aid the manipulation of character attributes; see **curs_attr**(3X). A*c h* whose character component is a space, and whose only attribute is **A_NORMAL**, is a *blank character*, and therefore combines with the window's background character; see **curs_bkgd**(3X).

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 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 *ch* is any other nonprintable character, *curses* draws it 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.

Adding printable characters with **waddch** 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 **waddch** 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.

wechochar

echochar and **wechochar** are equivalent to calling (**w**)**addch** followed by (**w**)**refresh** on **stdscr** or the specified window. *curses* interprets these functions as a hint to its optimizer that only a single character cell in the window is being altered between refreshes; 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 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	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*, 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 a write to its bottom right location is attempted, or
- it is not possible to add a complete character at the cursor position.

The last may be due to different causes:

- conversion of a wide character to a multibyte character sequence can fail, or
- at least one of the bytes resulting from wide character conversion to a multibyte character sequence cannot be added to the window. See section “PORTABILITY” below regarding the use of **waddch** with wide characters.

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.

EXTENSIONS

The 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 and 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, which identifies them as VT100 special characters.

The DEC Special Character and Line Drawing Set (VT100) is indexed by an ASCII character in the range 96 (‘) to 126 (~). That index character is part of the definition for the curses **ACS_** symbols. The VT100 special characters can be categorized in three groups:

- useful graphic symbols with a standard **ACS_** symbol, (e.g., the line-drawing symbols),
- possibly useful characters (these non-standard symbols),
- representations of control characters (e.g., newline and vertical tabulation).

A few **ACS_** symbols do not fit into DEC’s VT100 scheme. The AT&T Teletype 5410v1 arrow symbols and **ACS_BLOCK** use indices outside the range 96 to 126. Two of the Teletype symbols use indices in that range, with different meaning versus the VT100:

- **ACS_BOARD** corresponds to the VT100 symbol for newline
- **ACS_LANTERN** corresponds to the VT100 symbol for vertical tabulation

AT&T defined **ACS_** names for the most useful graphic symbols, as well as for its own. Its header file commented:

```
/*
 * Standard alternate character set.  The current ACS world is
 * evolving, so we support only a widely available subset: the
 * line drawing characters from the VT100, plus a few from the
 * Teletype 5410v1.  Eventually there may be support of more
 * sophisticated ACS line drawing, such as that in the Teletype
 * 5410, the HP line drawing set, and the like.  There may be
 * support for some non line oriented characters as well.
 *
 * Line drawing ACS names are of the form ACS_trbl, where t is
 * the top, r is the right, b is the bottom, and l is the left.
 * t, r, b, and l might be B (blank), S (single), D (double), or
 * T (thick).  The subset defined here only uses B and S.
 */
```

Although these less-useful graphic symbols were not given names, they were used in *terminfo* entries. The *ncurses* developers invented ACS-prefixed names for them.

PORTABILITY

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

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

SVr4 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, defines the ACS symbols as constants; others define them as elements of an array.

SVr4 used an array, *acs_map*, as does *ncurses*. NetBSD*cur ses* 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.

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. That character may have been more than eight bits wide in an SVr3 or SVr4 implementation, but X/Open Curses leaves the width of a non-wide character code unspecified. The standard further does not specify the internal structure of a *chtype*, though the use of bitwise operators to combine the character code with attributes and a color pair identifier into a *chtype* for passage to *waddch* is common. A portable application uses only the macros discussed in **curs_attr**(3X) to manipulate a *chtype*.

In *ncurses*, *chtype* holds an eight-bit character, but the library allows a multibyte character sequence to be passed via a succession of calls to *waddch*. Other implementations do not; a*waddc h* call transmits exactly one character, which may be rendered in one or more screen locations depending on whether it is printable (see **unctrl**(3X)). Depending on the locale,*ncur ses* inspects the byte passed in each *waddch* call and checks whether the latest call continues a multibyte character. When a character is*complete*, *ncur ses* 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 the foregoing 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*.
- If it cannot, use only *wadd_wch*.

HISTORY

4BSD (1980) introduced *waddch* and its variants.

SVr3 (1987) added the *echochar* and *wechochar* functions and most of the *ACS_* constants, except for *ACS_GEQUAL*, *ACS_LEQUAL*, *ACS_NEQUAL*, *ACS_PI*, *ACS_S3*, *ACS_S7*, and *ACS_STERLING*.

ncurses 1.9.6 (1995) furnished the remaining *ACS_* constants.

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_bkgd**(3X), **curs_clear**(3X), **curs_inch**(3X), **curs_outopts**(3X), **curs_refresh**(3X), **curs_variables**(3X), **putchar**(3)