

**NAME**

**getch**, **wgetch**, **mvgetch**, **mvwgetch**, **ungetch**, **has\_key** - get (or push back) characters from *curses* terminal keyboard

**SYNOPSIS**

```
#include <curses.h>
```

```
int getch(void);
int wgetch(WINDOW *win);
int mvgetch(int y, int x);
int mvwgetch(WINDOW *win, int y, int x);
```

```
int ungetch(int c);
```

```
/* extension */
```

```
int has_key(int c);
```

**DESCRIPTION****Reading Characters**

**wgetch** gathers a key stroke from the terminal keyboard associated with a *curses* window *win*. **ncurses(3X)** describes the variants of this function.

When input is pending, **wgetch** returns an integer identifying the key stroke; for alphanumeric and punctuation keys, this value corresponds to the character encoding used by the terminal. Use of the control key as a modifier often results in a distinct code. The behavior of other keys depends on whether *win* is in keypad mode; see subsection "Keypad Mode" below.

If no input is pending, then if the no-delay flag is set in the window (see **nodelay(3X)**), the function returns **ERR**; otherwise, *curses* waits until the terminal has input. If **cbreak(3X)** has been called, this happens after one character is read. If **nocbreak(3X)** has been called, it occurs when the next newline is read. If **halfdelay(3X)** has been called, *curses* waits until a character is typed or the specified delay elapses.

If **echo(3X)** has been called, and the window is not a pad, *curses* writes the returned character *c* to the window (at the cursor position) per the following rules.

- ⊕ If *c* matches the terminal's erase character, the cursor moves leftward one position and the new position is erased as if **wmove(3X)** and then **wdelch(3X)** were called. When the window's keypad mode is enabled (see below), **KEY\_LEFT** and **KEY\_BACKSPACE** are handled the same way.

- ⊕ *curses* writes any other *c* to the window, as with **wechochar**(3X).
- ⊕ If the window has been moved or modified since the last call to **wrefresh**(3X), *curses* calls **wrefresh**.

If *c* is a carriage return and **nl**(3X) has been called, **wgetch** returns the character code for line feed instead.

### Keypad Mode

To *curses*, key strokes not from the alphabetic section of the keyboard (those corresponding to the ECMA-6 character set--see *ascii*(7)--optionally modified by either the control or shift keys) are treated as *function keys*. (In *curses*, the term "function key" includes but is not limited to keycaps engraved with "F1", "PF1", and so on.) If the window is in keypad mode, these produce a numeric code corresponding to the **KEY\_** symbols listed in subsection "Predefined Key Codes" below; otherwise, they transmit a sequence of codes typically starting with the escape character, and which must be collected with multiple **wgetch** calls.

- ⊕ The *curses.h* header file declares many *predefined function keys* whose names begin with **KEY\_**; these object-like macros have values outside the range of eight-bit character codes.
- ⊕ In *ncurses*, *user-defined function keys* are configured with **define\_key**(3X); they have no names, but are also expected to have values outside the range of eight-bit codes.

A variable intended to hold a function key code must thus be of type *short* or larger.

Most terminals one encounters follow the ECMA-48 standard insofar as their function keys produce character sequences prefixed with the escape character ESC. This fact implies that *curses* cannot know whether the terminal has sent an ESC key stroke or the beginning of a function key's character sequence without waiting to see if, and how soon, further input arrives. When *curses* reads such an ambiguous character, it sets a timer. If the remainder of the sequence does not arrive within the designated time, **wgetch** returns the prefix character; otherwise, it returns the function key code corresponding to the unique sequence defined by the terminal. Consequently, a user of a *curses* application may experience a delay after pressing ESC while *curses* disambiguates the input; see section "EXTENSIONS" below. If the window is in "no time-out" mode, the timer does not expire; it is an infinite (or very large) value. See **notimeout**(3X). Because function key sequences usually begin with an escape character, the terminal may appear to hang in no time-out mode after the user has pressed ESC. Generally, further typing "awakens" *curses*.

### Ungetting Characters

**ungetch** places *c* into the input queue to be returned by the next call to **wgetch**. A single input queue

serves all windows.

### Predefined Key Codes

The header file *curses.h* defines the following function key codes.

- ⊕ Except for the special case of **KEY\_RESIZE**, a window's keypad mode must be enabled for **wgetch** to read these codes from it.
- ⊕ Not all of these are necessarily supported on any particular terminal.
- ⊕ The naming convention may seem obscure, with some apparent misspellings (such as "RSUME" for "resume"); the names correspond to the *terminfo* capability names for the keys, and were standardized before the IBM PC/AT keyboard layout achieved a dominant position in industry.

Symbol	Key name
<b>KEY_BREAK</b>	Break key
<b>KEY_DOWN</b>	Arrow keys
<b>KEY_UP</b>	
<b>KEY_LEFT</b>	
<b>KEY_RIGHT</b>	
<b>KEY_HOME</b>	Home key (upward+left arrow)
<b>KEY_BACKSPACE</b>	Backspace
<b>KEY_F0</b>	Function keys; space for 64 keys is reserved
<b>KEY_F(n)</b>	Function key <i>n</i> where $0 \leq n \leq 63$
<b>KEY_DL</b>	Delete line
<b>KEY_IL</b>	Insert line
<b>KEY_DC</b>	Delete character
<b>KEY_IC</b>	Insert character/Enter insert mode

<b>KEY_EIC</b>	Exit insert character mode
<b>KEY_CLEAR</b>	Clear screen
<b>KEY_EOS</b>	Clear to end of screen
<b>KEY_EOL</b>	Clear to end of line
<b>KEY_SF</b>	Scroll one line forward
<b>KEY_SR</b>	Scroll one line backward (reverse)
<b>KEY_NPAGE</b>	Next page/Page up
<b>KEY_PPAGE</b>	Previous page/Page down
<b>KEY_STAB</b>	Set tab
<b>KEY_CTAB</b>	Clear tab
<b>KEY_CATAB</b>	Clear all tabs
<b>KEY_ENTER</b>	Enter/Send
<b>KEY_SRESET</b>	Soft (partial) reset
<b>KEY_RESET</b>	(Hard) reset
<b>KEY_PRINT</b>	Print/Copy
<b>KEY_LL</b>	Home down/Bottom (lower left)
<b>KEY_A1</b>	Upper left of keypad
<b>KEY_A3</b>	Upper right of keypad
<b>KEY_B2</b>	Center of keypad
<b>KEY_C1</b>	Lower left of keypad
<b>KEY_C3</b>	Lower right of keypad

<b>KEY_BTAB</b>	Back tab key
<b>KEY_BEG</b>	Beg(inning) key
<b>KEY_CANCEL</b>	Cancel key
<b>KEY_CLOSE</b>	Close key
<b>KEY_COMMAND</b>	Cmd (command) key
<b>KEY_COPY</b>	Copy key
<b>KEY_CREATE</b>	Create key
<b>KEY_END</b>	End key
<b>KEY_EXIT</b>	Exit key
<b>KEY_FIND</b>	Find key
<b>KEY_HELP</b>	Help key
<b>KEY_MARK</b>	Mark key
<b>KEY_MESSAGE</b>	Message key
<b>KEY_MOUSE</b>	Mouse event occurred
<b>KEY_MOVE</b>	Move key
<b>KEY_NEXT</b>	Next object key
<b>KEY_OPEN</b>	Open key
<b>KEY_OPTIONS</b>	Options key
<b>KEY_PREVIOUS</b>	Previous object key
<b>KEY_REDO</b>	Redo key

**KEY\_REFERENCE** Ref(erence)  
key  
**KEY\_REFRESH** Refresh  
key  
**KEY\_REPLACE** Replace  
key  
**KEY\_RESIZE** Screen  
resized  
**KEY\_RESTART** Restart  
key  
**KEY\_RESUME** Resume  
key  
**KEY\_SAVE** Save  
key  
**KEY\_SELECT** Select  
key  
**KEY\_SUSPEND** Suspend  
key  
**KEY\_UNDO** Undo  
key

---

**KEY\_SBEG** Shifted beginning  
key  
**KEY\_SCANCEL** Shifted cancel  
key  
**KEY\_SCOMMAND** Shifted command  
key  
**KEY\_SCOPY** Shifted copy  
key  
**KEY\_SCREATE** Shifted create  
key  
**KEY\_SDC** Shifted delete character  
key  
**KEY\_SDL** Shifted delete line  
key  
**KEY\_SEND** Shifted end  
key  
**KEY\_SEOL** Shifted clear line  
key  
**KEY\_SEXIT** Shifted exit

	key
<b>KEY_SFIND</b>	Shifted find
	key
<b>KEY_SHELP</b>	Shifted help
	key
<b>KEY_SHOME</b>	Shifted home
	key
<b>KEY_SIC</b>	Shifted insert
	key
<b>KEY_SLEFT</b>	Shifted left arrow
	key
<b>KEY_SMESSAGE</b>	Shifted message
	key
<b>KEY_SMOVE</b>	Shifted move
	key
<b>KEY_SNEXT</b>	Shifted next object
	key
<b>KEY_SOPTIONS</b>	Shifted options
	key
<b>KEY_SPREVIOUS</b>	Shifted previous object
	key
<b>KEY_SPRINT</b>	Shifted print
	key
<b>KEY_SREDO</b>	Shifted redo
	key
<b>KEY_SREPLACE</b>	Shifted replace
	key
<b>KEY_SRIGHT</b>	Shifted right arrow
	key
<b>KEY_SRSUME</b>	Shifted resume
	key
<b>KEY_SSAVE</b>	Shifted save
	key
<b>KEY_SSUSPEND</b>	Shifted suspend
	key
<b>KEY_SUNDO</b>	Shifted undo
	key

Many keyboards feature a nine-key directional pad.

```

+----+-----+-----+
| A1| up| A3 |
+----+-----+-----+
|left| B2|right|
+----+-----+-----+
| C1|down| C3 |
+----+-----+-----+

```

Two of the symbols in the list above do *not* correspond to a physical key.

- ⊕ **wgetch** returns **KEY\_RESIZE**, even if the window's keypad mode is disabled, when *ncurses* handles a **SIGWINCH** signal; see **initscr(3X)** and **resizeterm(3X)**.
- ⊕ **wgetch** returns **KEY\_MOUSE** to indicate that a mouse event is pending collection; see **curs\_mouse(3X)**. Receipt of this code requires a window's keypad mode to be enabled, because to interpret mouse input (as with *xterm(1)*'s mouse protocol), *ncurses* must read an escape sequence, as with a function key.

### Testing Key Codes

In *ncurses*, **has\_key** returns a Boolean value indicating whether the terminal type recognizes its parameter as a key code value. See also **define\_key(3X)** and **key\_defined(3X)**.

### RETURN VALUE

Except for **has\_key**, these functions return **OK** on success and **ERR** on failure.

Functions taking a *WINDOW* pointer argument fail if the pointer is **NULL**.

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

**wgetch** also fails if

- ⊕ its timeout expires without any data arriving, or
- ⊕ execution was interrupted by a signal, in which case **errno** is set to **EINTR**.

**ungetch** fails if there is no more room in the input queue.

**has\_key** returns **TRUE** or **FALSE**.

### NOTES



*curses* discourages assignment of the ESC key to a discrete function by the programmer because the library requires a delay while it awaits the potential remainder of a terminal escape sequence.

Some key strokes are indistinguishable from control characters; for example, **KEY\_ENTER** may be the same as **^M**, and **KEY\_BACKSPACE** may be the same as **^H** or **^?**. Consult the terminal's *terminfo* entry to determine whether this is the case; see **infocmp**(1). Some *curses* implementations, including *ncurses*, honor the *terminfo* key definitions; others treat such control characters specially.

*curses* distinguishes the Enter keys in the alphabetic and numeric keypad sections of a keyboard because (most) terminals do. **KEY\_ENTER** refers to the key on the numeric keypad and, like other function keys, and is reliably recognized only if the window's keypad mode is enabled.

- ⊕ The *terminfo* **key\_enter** (**kent**) capability describes the character (sequence) sent by the Enter key of a terminal's numeric (or similar) keypad.
- ⊕ "Enter or send" is X/Open Curses's description of this key.

*curses* treats the Enter or Return key in the *alphabetic* section of the keyboard differently.

- ⊕ It usually produces a control code for carriage return (**^M**) or line feed (**^J**).
- ⊕ Depending on the terminal mode (raw, cbreak, or "cooked"), and whether **nl**(3X) or **nonl**(3X) has been called, **wgetch** may return either a carriage return or line feed upon an Enter or Return key stroke.

Use of **wgetch** with **echo**(3X) and neither **cbreak**(3X) nor **raw**(3X) is not well-defined.

Historically, the list of key code macros above was influenced by the function-key-rich keyboard of the AT&T 7300 (also known variously as the "3B1", "Safari 4", and "UNIX PC"), a 1985 machine.

Today's computer keyboards are based that of the IBM PC/AT and tend to have fewer. A *curses* application can expect such a keyboard to transmit key codes **KEY\_UP**, **KEY\_DOWN**, **KEY\_LEFT**, **KEY\_RIGHT**, **KEY\_HOME**, **KEY\_END**, **KEY\_PPAGE** (Page Up), **KEY\_NPAGE** (Page Down), **KEY\_IC** (Insert), **KEY\_DC** (Delete), and **KEY\_F**(*n*) for  $1 \leq n \leq 12$ .

**getch**, **mvgetch**, and **mvwgetch** may be implemented as macros.

## EXTENSIONS

In *ncurses*, when a window's "no time-out" mode is *not* set, the **ESCDELAY** variable configures the duration of the timer used to disambiguate a function key character sequence from a series of key strokes beginning with ESC typed by the user; see **curs\_variables**(3X).

**has\_key** was designed for **ncurses(3X)**, and is not found in SVr4 *curses*, 4.4BSD *curses*, or any other previous *curses* implementation.

## PORTABILITY

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

X/Open Curses, Issue 4 describes **getch**, **wgetch**, **mvwgetch**, **mvwgetch**, and **ungetch**. It specifies no error conditions for them.

**wgetch** reads only single-byte characters.

The echo behavior of these functions on input of **KEY\_** or backspace characters was not specified in the SVr4 documentation. This description is adapted from X/Open Curses.

The behavior of **wgetch** in the presence of signal handlers is unspecified in the SVr4 documentation and X/Open Curses. In historical *curses* implementations, it varied depending on whether the operating system's dispatch of a signal to a handler interrupting a *read(2)* call in progress, and also (in some implementations) whether an input timeout or non-blocking mode has been set. Programmers concerned about portability should be prepared for either of two cases: (a) signal receipt does not interrupt **wgetch**; or (b) signal receipt interrupts **wgetch** and causes it to return **ERR** with **errno** set to **EINTR**.

**KEY\_MOUSE** is mentioned in X/Open Curses, along with a few related *terminfo* capabilities, but no higher-level functions use the feature. The implementation in *ncurses* is an extension.

**KEY\_RESIZE** and **has\_key** are extensions first implemented for *ncurses*. By 2022, *PDCurses* and NetBSD *curses* had added them along with **KEY\_MOUSE**.

## SEE ALSO

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

**curses(3X)**, **curs\_addch(3X)**, **curs\_inopts(3X)**, **curs\_mouse(3X)**, **curs\_move(3X)**, **curs\_outopts(3X)**, **curs\_refresh(3X)**, **curs\_variables(3X)**, **resizeterm(3X)**, **ascii(7)**

ECMA-6 "7-bit coded Character Set" <<https://ecma-international.org/publications-and-standards/standards/ecma-6/>>

ECMA-48 "Control Functions for Coded Character Sets" <<https://ecma-international.org/>>

curs\_getch(3X)

Library calls

curs\_getch(3X)

publications-and-standards/standards/ecma-48/>