NAME

get_wch, **wget_wch**, **mvget_wch**, **mvwget_wch**, **unget_wch** − get (or push back) a wide character from *curses* terminal keyboard buffer

SYNOPSIS

```
#include <curses.h>

int get_wch(wint_t * wch);
int wget_wch(WINDOW * win, wint_t * wch);
int mvget_wch(int y, int x, wint_t * wch);
int mvwget_wch(WINDOW * win, int y, int x, wint_t * wch);

int unget_wch(const wchar_t wc);
```

DESCRIPTION

Reading Characters

wget_wch gathers a key event from the terminal keyboard associated with a *curses* window *win*, returning **OK** if a wide character is read, **KEY_CODE_YES** if a function key is read, and **ERR** if no key event is available. **ncurses**(3X) describes the variants of this function.

When input is pending, **wget_wch** stores an integer identifying the key event in *wch*; for alphanumeric and punctuation keys, this value corresponds to the character encoding used by the terminal. Use of the control key as a modifier, by holding it down while pressing and releasing another key, often results in a distinct code. The behavior of other keys depends on whether *win* is in keypad mode; see subsections "Keypad Mode" and "Key Codes" in **getch**(3X).

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) or **raw**(3X) has been called, this happens after one character is read. If **nocbreak**(3X) or **noraw**(3X) has been called, it occurs when the next newline is read. (Because the terminal's canonical or "cooked" mode is line-buffered, multiple **wget_wch** calls may then be necessary to empty the input queue.) If **halfdelay**(3X) has been called, *curses* waits until input is available or the specified delay elapses.

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

- If the wide character matches the terminal's erase character (see **erasewchar**(3X)), 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 wide character to the window, as with **wecho_wchar**(3X).
- If the window *win* has been moved or modified since the last call to **wrefresh**(3X), *curses* calls **wrefresh** on it.

If the wide character is a carriage return and **nl**(3X) has been called, **wget_wch** stores the wide character code for line feed in *wch* instead.

Ungetting Characters

unget_wch places *wc* into the input queue to be retrieved by the next call to **wget_wch**. A single input queue serves all windows associated with the screen.

RETURN VALUE

wget_wch returns **OK** when it reads a wide character, **KEY_CODE_YES** when it reads a function key code, and **ERR** on failure. **wget_wch** fails if its timeout expires without any data arriving, which cannot happen if **nodelay**(3X) is in effect on the window.

In *ncurses*, **wget_wch** also fails if

- the *curses* screen has not been initialized,

- (for functions taking a *WINDOW* pointer argument) *win* is a null pointer, or
- execution was interrupted by a signal, in which case *errno* is set to *EINTR*.

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

unget_wch returns **OK** on success and **ERR** on failure. In*ncurses*, **unget_wch** fails if

- the *curses* screen has not been initialized, or
- there is no more room in the input queue.

NOTES

See the "NOTES" section of**wgetch**(3X).

All of these functions except **wget_wch** and **unget_wch** may be implemented as macros.

Unlike **wgetch**(3X), **wget_wch** stores the value of the input character in an additional *wch* parameter instead of the return value.

Unlike **ungetch**, **unget_wch** cannot distinguish function key codes from conventional character codes. An application can overcome this limitation by pushing function key codes with **ungetch** and subsequently checking the return value of **wget_wch** for a match with **KEY_CODE_YES**.

EXTENSIONS

See the "EXTENSIONS" section of **wgetch**(3X).

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.

See the "PORTABILITY" section of**wgetch**(3X) re garding the interaction of *wget_wch* with signal handlers.

HISTORY

X/Open Curses Issue 4 (1995) initially specified these functions. The System V Interface Definition Version 4 of the same year specified functions named *wgetwch* (with its variants) *ungetwch*. These were later additions to SVr4.*x*, not appearing in the first SVr4 (1989). They differ from X/Open's later *wget_wch* and *unget_wch* in that *wgetwch* takes no *wch* argument, but returns the (wide) key code as an *int* (with no provision for distinguishing a character code from a function key code); and *ungetwch* takes a non-*const int*ar - gument.

SEE ALSO

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

curses(3X), **curs_add_wch**(3X), **curs_inopts**(3X), **curs_move**(3X), **curs_refresh**(3X)