NAME

getstr, **getnstr**, **wgetstr**, **wgetnstr**, **mvgetstr**, **mvgetnstr**, **mvwgetstr**, **mvwgetnstr** − read a character string from *curses* terminal keyboard

SYNOPSIS

#include <curses.h>

int getstr(char * *str***);**
int wgetstr(WINDOW * *win***, char *** *str***);**
int mvgetstr(int *y***, int** *x***, char *** *str***);**
int mvwgetstr(WINDOW * *win***, int** *y***, int** *x***, char *** *str***);**

int getnstr(char * *str***, int** *n***);**
int wgetnstr(WINDOW * *win***, char *** *str***, int** *n***);**
int mvgetnstr(int *y***, int** *x***, char *** *str***, int** *n***);**
int mvwgetnstr(WINDOW * *win***, int** *y***, int** *x***, char *** *str***, int** *n***);**

DESCRIPTION

wgetstr populates a user-supplied string buffer *str* by repeatedly calling **wgetch**(3X) with the *win* argument until a line feed or carriage return character is input. The function

- does not copy the terminating character to *str*;
- always terminates *str* with a null character;
- interprets the screen's erase and kill characters (see **erasechar**(3X) and **killchar**(3X));
- recognizes function keys only if the screen's keypad option is enabled (see **keypad**(3X));
- treats the function keys **KEY_LEFT** and **KEY_BACKSPACE** the same as the erase character; and
- discards function key inputs other than those treated as the erase or kill characters, calling **beep**(3X).

If any characters have been written to the input buffer, the erase character replaces the character at the current position in the buffer with a null character, then decrements the position by one; the kill character does the same repeatedly, backtracking to the beginning of the buffer.

If the screen's echo option is enabled (see **echo**(3X)), **wgetstr** updates *win* with **waddch**(3X). Further,

- the erase character and its function key synonyms move the cursor to the left (if not already where it was located when **wgetstr** was called) and
- the kill character returns the cursor to where it was located when **wgetstr** was called.

wgetnstr is similar, but reads at most *n* characters, aiding the application to avoid overrunning the buffer to which *str* points. *curses* ignores an attempt to input more than *n* characters (other than the terminating line feed or carriage return), calling **beep**(3X). If *n* is ne gative, **wgetn_wstr** reads up to *LINE_MAX* characters (see *sysconf*(3)).

ncurses(3X) describes the variants of these functions.

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,
- *str* is a null pointer, or
- an internal **wgetch**(3X) call fails.

Further, in *ncurses*, these functions return **KEY_RESIZE** if a *SIGWINCH* event interrupts the function.

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

NOTES

All of these functions except **wgetnstr** may be implemented as macros.

Reading input that overruns the buffer pointed to by *str* causes undefined results. Use the **n**-infixed functions, and allocate sufficient storage for *str* — at least *n*+1 times **sizeof(char)**.

While these functions conceptually implement a series of calls to **wgetch**, they also temporarily change properties of the *curses* screen to permit simple editing of the input buffer. Each function saves the screen's state, calls **nl**(3X), and, if the screen was in canonical ("cooked") mode, **cbreak**(3X). Before returning, it restores the saved screen state. Other implementations differ in detail, affecting which control characters they can accept in the buffer; see section "PORTABILITY" below.

EXTENSIONS

getnstr, **wgetnstr**, **mvgetnstr**, and **mvwgetnstr**'s handing of negative *n* values is an *ncurses* extension.

The return value **KEY_RESIZE** is an *ncurses* extension.

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, but indicates that *wgetnstr* and its variants read "the entire multi-byte sequence associated with a character" and "fail" if *n* and *str* together do not describe a buffer "large enough to contain any complete characters". In *ncurses*, however, *wgetch* reads only single-byte characters, so this scenario does not arise.

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

SVr3 and early SVr4 *curses* implementations did not reject function keys; the SVr4 documentation asserted that, like the screen's erase and kill characters, they were

> interpreted, as well as any special keys (such as function keys, "home" key, "clear" key, *etc.*)

without further detail. It lied. The "character" value appended to the string by those implementations was predictable but not useful — being, in fact, the low-order eight bits of the key code's *KEY_* constant value. (The same language, unchanged except for styling, survived into X/Open Curses Issue 4, Version 2 but disappeared from Issue 7.)

A draft of X/Open Curses Issue 5 (which never saw final release) stated that these functions "read at most *n* bytes" but did not state whether the terminating null character counted toward that limit. X/Open Curses Issue 7 changed that to say they "read at most *n*−1 bytes" to allow for the terminating null character. As of 2018, some implementations count it, some do not.

- *ncurses* 6.1 and *PDCurses* do not count the null character toward the limit, while Solaris and NetBSD *curses* do.
- Solaris *xcurses* offers both behaviors: its wide-character *wgetn_wstr* reserves room for a wide null character, but its non-wide *wgetnstr* does not consistently count a null character toward the limit.

X/Open Curses does not specify what happens if the length *n* is negative.

- *ncurses* 6.2 uses *LINE_MAX* or a larger (system-dependent) value provided by *sysconf*(3). If neither *LINE_MAX* nor *sysconf* is available, *ncurses* uses the POSIX minimum value for *LINE_MAX* (2048). In either case, it reserves a byte for the terminating null character.
- In SVr4 *curses*, a negative *n* tells *wgetnstr* to assume that the caller's buffer is large enough to hold the result; that is, the function then acts like *wgetstr*. X/Open Curses does not mention this behavior (or anything related to nonpositive *n* values), however most *curses* libraries implement it. Most implementations nevertheless enforce an upper limit on the count of bytes they write to the destination buffer *str*.
- BSD *curses* lacked *wgetnstr*, and its *wgetstr* wrote to *str* unboundedly, as did that in SVr2.
- *PDCurses*, and SVr3 and later, and Solaris *curses* limit both functions to writing 256 bytes. Other System V-based platforms likely use the same limit.

- Solaris *xcurses* limits the write to *LINE_MAX* bytes (see *sysconf*(3)).
- NetBSD 7 *curses* imposes no particular limit on the length of the write, but does validate *n* to ensure that it is greater than zero. A comment in NetBSD's source code asserts that SUSv2 specifies this.

Implementations vary in their handling of input control characters.

- While they may enable the screen's echo option, some do not take it out of raw mode, and may take cbreak mode into account when deciding whether to handle echoing within *wgetnstr* or to rely on it as a side effect of calling *wgetch*.
- Originally, *ncurses*, like its progenitor *pcurses*, had its *wgetnstr* call *noraw* and *cbreak* before accepting input. That may have been done to make function keys work; it is not necessary with modern *ncurses*.

  Since 1995, *ncurses* has provided handlers for *SIGINTR* and *SIGQUIT* events, which are typically generated at the keyboard with **^C** and ^\ respectively. In cbreak mode, those handlers catch a signal and stop the program, whereas other implementations write those characters into the buffer.
- Starting with *ncurses* 6.3 (2021), *wgetnstr* preserves raw mode if the screen was already in that state, allowing one to enter the characters the terminal interprets as interrupt and quit events into the buffer, for better compatibility with SVr4 *curses*.

HISTORY

4BSD (1980) introduced *wgetstr* along with its variants.

SVr3.1 (1987) added *wgetnstr*, but none of its variants.

X/Open Curses Issue 4 (1995) specified *getnstr*, *mvgetnstr*, and *mvwgetnstr*.

SEE ALSO

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

curses(3X), **curs_addch**(3X), **curs_getch**(3X), **curs_inopts**(3X), **curs_termattrs**(3X),