NAME

cbreak, **echo**, **halfdelay**, **intrflush**, **is_cbreak**, **is_echo**, **is_nl**, **is_raw**, **keypad**, **meta**, **nl**, **nocbreak**, **nodelay**, **noecho**, **nonl**, **noqiflush**, **noraw**, **notimeout**, **qiflush**, **raw**, **timeout**, **wtimeout**, **typeahead** − get and set *curses* terminal input options

SYNOPSIS

```
#include <curses.h>

int cbreak(void);
int nocbreak(void);

int echo(void);
int noecho(void);

int intrflush(WINDOW * win /* ignored */, bool bf);
int keypad(WINDOW * win, bool bf);
int meta(WINDOW * win /* ignored */, bool bf);
int nodelay(WINDOW * win, bool bf);
int notimeout(WINDOW * win, bool bf);

int nl(void);
int nonl(void);

void qiflush(void);
void noqiflush(void);

int raw(void);
int noraw(void);

int halfdelay(int tenths);
void timeout(int delay);
void wtimeout(WINDOW * win, int delay);

int typeahead(int fd);

/* extensions */
int is_cbreak(void);
int is_echo(void);
int is_nl(void);
int is_raw(void);
```

DESCRIPTION

curses offers configurable parameters permitting an application to control the handling of input from the terminal. Some, such as those affecting the terminal's *mode* or line discipline, are global, applying to all windows; others apply only to a specific window. The library does not automatically apply such parameters to new or derived windows; an application must configure each window for the desired behavior.

Some descriptions below make reference to an *input character reading function*: this is **wgetch**(3X) in the non-wide character *curses* API and **wget_wch**(3X) in the wide character API. In addition to the variant forms of these described in **ncurses**(3X), the *curses* functions **wgetstr**(3X) and **wget_wstr**(3X) and their own variants call the appropriate input character reading function.

cbreak, nocbreak

Normally, the terminal driver buffers typed characters, not delivering them to an application until a line feed or carriage return is typed. This canonical ("cooked") line discipline also supports software flow control, simple line editing functions (character and word erase, and whole-line erasure or "kill"), and job control. **cbreak** configures the terminal in *cbreak mode*, which disables line buffering and erase and kill character processing — the interrupt, quit, suspend, and flow control characters are unaffected — and makes characters typed by the user immediately available to the program. **nocbr eak** restores canonical ("cooked") mode.

The state of the terminal is unknown to a *curses* application when it starts; therefore, a program should call **cbreak** or **nocbreak** explicitly. Most interactive programs using *curses* set cbreak mode. Calling **cbreak**

overrides **raw**. The man page for the input character reading function discusses how **cbreak** and **nocbreak** interact with **echo** and **noecho**.

echo, noecho

echo and **noecho** determine whether characters typed by the user are written to the *curses* window by the input character reading function as they are typed. *cur ses* always disables the terminal driver's own echoing. By default, a *curses* screen's echo option is set. Authors of most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they call **noecho**. The man page for the input character reading function discusses how **echo** and **noecho** interact with **cbreak** and **nocbreak**.

halfdelay

halfdelay configures *half-delay mode*, which is similar to cbreak mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenth-seconds, an input character reading function returns **ERR** if no input is pending. The value of *tenths* must be between 1 and 255. Use **nocbreak** to leave half-delay mode.

intrflush

intrflush calls **qiflush** (see below) if *bf* is **TRUE**, and **noqiflush** if *bf* is **FALSE**. It ignores its*win* ar gument.

keypad

keypad enables recognition of a terminal's function keys. If enabled (*bf* is **TRUE**) then when an input character reading function reads ESC, it waits for further input corresponding to an escape sequence defined by the terminal type description. If a valid sequence populates the input stream, the input character reading function returns a value representing the function key, such as **KEY_LEFT**. (Wide-character API users: **wget_wch**(3X) returns **KEY_CODE_YES** to indicate the availability of a function key code in its *wch* parameter.) If the sequence is invalid, the input character reading function returns only its last character. If disabled (*bf* is **FALSE**), *curses* does not treat function keys specially and the program has to interpret escape sequences itself. If the terminal type description defines the **keypad_local** (**rmkx**) and **keypad_xmit** (**smkx**) capabilities, enabling a window's keypad mode sets the terminal's keypad to transmit, and disabling keypad mode sets the terminal's keypad to work locally. By default, a window's keypad mode is off.

meta

Initially, whether the terminal returns 7- or 8-bit character codes on input depends on the configuration of the terminal driver; on POSIX systems, see *termios*(3). To force 8 bits to be returned, call **meta(**. . ., **TRUE)**; this is equivalent, on POSIX systems, to setting the CS8 flag on the terminal. To force 7 bits to be returned, call **meta(**. . ., **FALSE)**; this is equivalent, on POSIX systems, to setting the CS7 flag on the terminal. *curses* ignores the window argument *win*. If the*terminfo* string capabilities **meta_on** (**smm**) and **meta_off** (**rmm**) are defined for the terminal type, enabling meta mode sends **smm**'s value to the terminal and disabling it sends that of **rmm** to the terminal.

nl, nonl

Initially, whether the terminal reports a carriage return using the character code for a line feed in cbreak or raw modes depends on the configuration of the terminal driver; see *termios*(3). **nl** configures the terminal to perform this translation. **nonl** disables it. Under its canonical ("cook ed") line discipline, the terminal driver always translates carriage returns to line feeds.

nodelay

nodelay configures the input character reading function to be non-blocking for window *win*. If no input is ready, the reading function returns **ERR**. If disabled (*bf* is **FALSE**), the reading function does not return until it has input.

notimeout

When **keypad** has been called on a window and the input character reading function reads an ESC character from it, *curses* sets a timer while waiting for the next character. If the timer elapses,*cur ses* interprets the ESC as an explicit press of the terminal's Escape key (or equivalent). **notimeout(***win***, TRUE)** disables this timer. The purpose of the timeout is to distinguish sequences produced by a function key from those

typed by a user. If this timer is disabled,*cur ses* waits forever for subsequent keystrokes until it determines the escape sequence to be valid or invalid.

qiflush, noqiflush

qiflush and **noqiflush** configure the terminal driver's treatment of its input and output queues when it handles the interrupt, suspend, or quit characters under the canonical ("cooked") or cbreak line disciplines on POSIX systems; see*termios*(3). The def ault behavior is inherited from the terminal driver settings. Calling **qiflush** configures the terminal to *flush* the queues (discarding their contents) when any of these events occurs, giving the impression of faster response to user input, but making the library's model of the screen contents incorrect. Calling **noqiflush** prevents such flushing, but might frustrate impatient users on slow connections if a *curses* update of the screen is in progress when the event occurs; see **typeahead** below for a mitigation of this problem. You may want to call **noqiflush** in a signal handler if, after the handler exits, you want output to continue as though the signal had not occurred.

raw, noraw

raw configures the terminal to read input in *raw mode*, which is similar to cbreak mode (see **cbreak** above) except that it furthermore passes through the terminal's configured interrupt, quit, suspend, and flow control characters uninterpreted to the application, instead of generating a signal or acting on I/O flow. The behavior of the terminal's "Break" key (if any) depends on terminal driver configuration parameters that *curses* does not handle. **noraw** restores the terminal's canonical ("cooked") line discipline.

timeout, wtimeout

wtimeout configures whether a *curses* input character reading function called on window *win* uses blocking or non-blocking reads. If *delay* is negative, *curses* uses a blocking read, waiting indefinitely for input. If *delay* is zero, the read is non-blocking; an input character reading function returns **ERR** if no input is pending. If*delay* is positi ve, an input character reading function blocks for *delay* milliseconds, and returns **ERR** if the delay elapses and there is still no input pending. **timeout** calls **wtimeout** on **stdscr**.

typeahead

Normally, a *curses* library checks the terminal's input file descriptor for activity with *poll*(2) or *select*(2) while updating the screen; if it finds any, it postpones output until the next **wrefresh**(3X) or **doupdate**(3X) call, allowing faster response to user key strokes. The library tests the file descriptor corresponding to the *FILE* stream pointer passed to **newterm**(3X) (or *stdin* if **initscr**(3X) was called), for pending input. **typeahead** instructs *curses* to test file descriptor *fd* instead. An *fd* of **−1** disables the check.

RETURN VALUE

timeout and **wtimeout** return no value.

cbreak, **nocbreak**, **echo**, **noecho**, **halfdelay**, **intrflush**, **keypad**, **meta**, **nodelay**, **notimeout**, **nl**, **nonl**, **raw**, **noraw**, and **typeahead** return **OK** on success and **ERR** on failure.

In *ncurses*, the functions in the previous paragraph return **ERR** if

- the library's *TERMINAL* structure for the device has not been initialized with **initscr**(3X), **newterm**(3X), or **setupterm**(3X), or
- *win* is a null pointer (except with **intrflush** and **meta**, which ignore its value).

Further, **halfdelay** returns **ERR** if *delay* is outside the range 1..255.

See section "EXTENSIONS" below for the return values of **is_cbreak**, **is_echo**, **is_nl**, and **is_raw**.

NOTES

echo, **noecho**, **halfdelay**, **intrflush**, **meta**, **nl**, **nonl**, **nodelay**, **notimeout**, **noqiflush**, **qiflush**, **timeout**, and **wtimeout** may be implemented as macros.

noraw and **nocbreak** follow historical practice in that they attempt to restore the terminal's canonical ("cooked") line discipline from raw and cbreak, respectively. Mixing**raw**/**noraw** calls with **cbreak**/**nocbreak** calls leads to terminal driver control states that are hard to predict or understand; doing so is not recommended.

curses documentation uses the terms "delay" and "timeout" freely to describe two related but distinct aspects of input handling, at the risk of confusing the user. The functions**halfdelay**, **nodelay**, **timeout**, and

wtimeout configure whether the input character reading function (**wgetch**(3X) or **wget_wch**(3X)) waits for keyboard input to begin, and for how long. **k eypad** configures whether that function waits for further input if the first character it reads is ESC. Calling **notimeout**, which has nothing to do with **timeout** or **wtimeout**, makes this delay in expectation of further characters effectively infinite. X/Open Curses affords no means of otherwise configuring the length of this second delay, but an AIX and *ncurses* extension, **ESCDELAY**, is available both as an environment variable and a global symbol permitting the user and application, respectively, to do so; see **ncurses**(3X) and **curs_variables**(3X).

EXTENSIONS

ncurses provides four "is_" functions corresponding to **cbreak**, **echo**, **nl**, and **raw**, permitting their states to be queried by the application.

Query	Set	Reset
is_cbreak	cbreak	nocbreak
is_echo	echo	noecho
is_nl	nl	nonl
is_raw	raw	noraw

In each case, the function returns

1 if the option is set,

0 if the option is unset, or

−1 if the library's *TERMINAL* structure for the device has not been initialized.

PORTABILITY

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

Except as noted in section "EXTENSIONS" above, 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*".

ncurses follows X/Open Curses and the historical practice of System V *curses*, clearing the terminal driver's "echo" flag when initializing the screen. BSD *curses* did not, but its *raw* function turned it off as a side effect. For best portability, call *echo* or *noecho* explicitly just after initialization, even if your program retains the terminal's canonical ("cooked") line discipline.

X/Open Curses is ambiguous regarding whether *raw* should disable the carriage return and line feed translation feature controlled by *nl* and *nonl*. BSD*cur ses* turned off these translations; System V *curses* did not. *ncurses* does so, on the assumption that a programmer requesting raw input wants a clean (ideally, 8-bit clean) connection that the operating system will not alter.

When **keypad** is first enabled for a window, *ncurses* loads the standard function key string capabilities for the terminal type description of its screen; see the entries beginning with "key_" in**terminf o**(5). If that description includes extended string capabilities, produced by the **−x** option of **tic**(1), for example, then *ncurses* also defines keys for the capabilities whose codes begin with "k". *ncur ses* generates a numeric key code for each such extended capability; depending on previous loads of terminal type descriptions, these may differ from one execution of a program to the next. **keyname**(3X) recognizes the generated key codes and returns a name beginning with "k" denoting the *terminfo* capability name rather than "KEY_", used for *curses* key names. On the other hand, an application can use **define_key**(3X) to bind a selected key to a string of the programmer's choice. This feature enables an application to check for its presence with **tigetstr**(3X), and reassign the numeric key code to match its own needs.

Low-level applications can use **tigetstr**(3X) to obtain the definition of any string capability. *curses* applications use the input character reading function to obtain key codes from input and rely upon the order in which the string capabilities are loaded. Multiple key capability strings can have the same value, but the input character reading function can report only one key code. Most *curses* implementations (including *ncurses*) load key definitions in the order they appear in the **strfnames** array of string capability names; see **term_variables**(3X). The last capability read using a particular definition determines the key code to be

reported. In*ncur ses*, extended capabilities can be interpreted as key definitions. The library loads these after its built-in definitions, and if an extended capability's value is the same as one previously loaded, the library uses the later definition.

HISTORY

4BSD (1980) introduced *echo*, *noecho*, *nl*, *nonl*, *raw*, and *noraw*.

SVr2 (1984) featured a new terminal driver, extending the *curses* API to support it with *cbreak*, *nocbreak*, *intrflush*, *keypad*, *meta*, *nodelay*, and *typeahead*.

SVr3 (1987) added *halfdelay*, *notimeout*, and *wtimeout*. *qiflush* and *noqiflush* appeared in SVr3.1 (1987), at which point *intrflush* became a wrapper for either of these functions, depending on the value of its Boolean argument. SVr3.1 also added*timeout*.

ncurses 6.5 (2024) introduced *is_cbreak*, *is_echo*, *is_nl*, and *is_raw*.

Formerly, *ncurses* used *nl* and *nonl* to control the conversion of newlines to carriage return/line feed on output as well as input. X/Open Curses documents the use of these functions only for input. This difference arose from converting the *pcurses* source (1986), which used *ioctl*(2) calls and the *sgttyb* structure, to *termios* (the POSIX terminal API). In the former, both input and output conversions were controlled via a single option "CRMOD", while the latter separates these features. Because that conversion interferes with output optimization, *ncurses* 6.2 (2020) amended *nl* and *nonl* to eliminate their effect on output.

SEE ALSO

curses(3X), **curs_getch**(3X), **curs_initscr**(3X), **curs_util**(3X), **define_key**(3X), **termios**(3), **term_variables**(3X)