NAME

curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines

SYNOPSIS

cc [*flag* ...] *file* ... **–lcurses** [*library* ..]

#include <curses.h>

int cbreak(void);

int nocbreak(void);

int echo(void);

int noecho(void);

int halfdelay(int *tenths***);**

int intrflush(WINDOW **win***, bool** *bf***);**

int keypad(WINDOW **win***, bool** *bf***);**

int meta(WINDOW **win***, bool** *bf***);**

int nodelay(WINDOW **win***, bool** *bf***);**

int notimeout(WINDOW **win***, bool** *bf***);**

int raw(void);

int noraw(void);

void noqiflush(void);

void qiflush(void);

void timeout(int *delay***);**

void wtimeout(WINDOW **win***, int** *delay***);**

int typeahead(int *fildes***);**

MT-LEVEL

Unsafe

DESCRIPTION

The **cbreak( )** and **nocbreak( )** routines put the terminal into and out of **cbreak( )** mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in **cbreak( )** mode, as the mode is inherited; therefore, a program should call **cbreak( )** or **nocbreak( )** explicitly. Most interactive programs using **curses** set the **cbreak( )** mode.

Note that **cbreak( )** overrides **raw( )**. (See **curs_getch**(3X) for a discussion of how these routines interact with **echo( )** and **noecho( )**.)

The **echo( )** and **noecho( )** routines control whether characters typed by the user are echoed by **getch( )** as they are typed. Echoing by the tty driver is always disabled, but initially **getch( )** is in echo mode, so characters typed are echoed. 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 disable echoing by calling **noecho( )**. (See **curs_getch**(3X) for a discussion of how these routines interact with **cbreak( )** and **nocbreak( )**.)

The **halfdelay( )** routine is used for 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* tenths of seconds, **ERR** is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use **nocbreak( )** to leave half-delay mode.

If the **intrflush( )** option is enabled, (*bf* is **TRUE**), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing **curses** to have the wrong idea of what is on the screen. Disabling (*bf* is **FALSE**), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The **keypad( )** option enables the keypad of the user's terminal. If enabled (*bf* is **TRUE**), the user can press a function key (such as an arrow key) and **wgetch( )** returns a single value representing the function key, as in **KEY_LEFT**. If disabled (*bf* is **FALSE**), **curses** does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when **wgetch( )** is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see **termio**(7I)). To force 8 bits to be returned, invoke **meta**(*win*, **TRUE**). To force 7 bits to be returned, invoke **meta**(*win*, **FALSE**). The window argument, *win*, is always ignored. If the terminfo capabilities **smm** (meta_on) and **rmm** (meta_off) are defined for the terminal, **smm** is sent to the terminal when **meta**(*win*, **TRUE)** is called and **rmm** is sent when **meta**(*win*, **FALSE**) is called.

The **nodelay( )** option causes **getch( )** to be a non-blocking call. If no input is ready, **getch( )** returns **ERR**. If disabled (*bf* is **FALSE**), **getch( )** waits until a key is pressed.

While interpreting an input escape sequence, **wgetch( )** sets a timer while waiting for the next character. If **notimeout(***win*, **TRUE**) is called, then **wgetch( )** does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the **raw( )** and **noraw( )** routines, the terminal is placed into or out of raw mode. Raw mode is similar to **cbreak( )** mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the BREAK key depends on other bits in the tty driver that are not set by **curses**.

When the **noqiflush( )** routine is used, normal flush of input and output queues associated with the **INTR**, **QUIT** and **SUSP** characters will not be done (see **termio**(7I)). When **qiflush( )** is called, the queues will be flushed when these control characters are read.

The **timeout( )** and **wtimeout( )** routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns **ERR** if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns **ERR** if there is still no input. Hence, these routines provide the same functionality as **nodelay( )**, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

curses does ''line-breakout optimization'' by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until **refresh( )** or **doupdate( )** is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to **newterm( )**, or **stdin** in the case that **initscr( )** was used, will be used to do this typeahead checking. The **typeahead( )** routine specifies that the file descriptor *fildes* is to be used to check for typeahead instead. If *fildes* is –1, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return **ERR** upon failure and an integer value other than **ERR** upon successful completion, unless otherwise noted in the preceding routine descriptions.

SEE ALSO

curs_getch(3X), **curs_initscr**(3X), **curses**(3X), **termio**(7I)

NOTES

The header **<curses.h>** automatically includes the headers **<stdio.h>** and **<unctrl.h>**.

Note that **echo( )**, **noecho( )**, **halfdelay( )**, **intrflush( )**, **meta( )**, **nodelay( )**, **notimeout( )**, **noqiflush( )**, **qiflush( )**, **timeout( )**, and **wtimeout( )** may be macros.