NAME

PC, **UP**, **BC**, **ospeed**, **tgetent**, **tgetflag**, **tgetnum**, **tgetstr**, **tgoto**, **tputs** - *curses* emulation of *termcap*

SYNOPSIS

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

char PC;
char * UP;
char * BC;
short ospeed;

int tgetent(char *bp, const char *name);
int tgetflag(const char *id);
int tgetnum(const char *id);
char *tgetstr(const char *id, char **area);
char *tgoto(const char *cap, int col, int row);
int tputs(const char *str, int affcnt, int (*putc)(int));
```

DESCRIPTION

ncurses provides the foregoing variables and functions as a compatibility layer for programs that use the *termcap* library. The API is the same, but behavior is emulated using the *terminfo* database. Thus, it can be used only to query the capabilities of terminal database entries for which a *terminfo* entry has been compiled.

Initialization

tgetent loads the terminal database entry for *name*; see **term**(7). This must be done before calling any of the other functions. It returns

1 on success,

0 if there is no such entry (or if the matching entry describes a generic terminal, having too little information for *curses* applications to run), and

-1 if the *terminfo* database could not be found.

This implementation differs from those of historical *termcap* libraries.

- *ncurses* ignores the buffer pointer *bp*, as do other *termcap* implementations conforming to portions of X/Open Curses now withdrawn. The BSD *termcap* library would store a copy of

the terminal type description in the area referenced by this pointer. *terminfo* stores terminal type descriptions in compiled form, which is not the same thing.

- The meanings of the return values differ. The BSD *termcap* library does not check whether the terminal type description includes the **generic** (**gn**) capability, nor whether the terminal type description supports an addressable cursor, a property essential for any *curses* implementation to operate.

Retrieving Capability Values

tgetflag reports the Boolean entry for *id*, or zero if it is not available.

tgetnum obtains the numeric entry for *id*, or -1 if it is not available.

tgetstr returns the string entry for *id*, or **NULL** if it is not available. Use **tputs** to output the string returned. The *area* parameter is used as follows.

- It is assumed to be the address of a pointer to a buffer managed by the calling application.
- However, *ncurses* checks to ensure that *area* is not **NULL**, and also that the resulting buffer pointer is not **NULL**. If either check fails, *area* is ignored.
- If the checks succeed, *ncurses* also copies the return value to the buffer pointed to by *area*, and the library updates *area* to point past the null character terminating this value.
- The return value itself is an address in the terminal type description loaded into memory.

Applying String Capabilities

String capabilities can be parameterized; see subsection "Parameterized Strings" in **terminfo**(5). **tgoto** applies its second and third arguments to the parametric placeholders in the capability stored in the first argument.

- The capability may contain padding specifications; see subsection "Delays and Padding" of **terminfo**(5). The output of **tgoto** should thus be passed to **tputs** rather than some other output function such as *printf*(3).
- While **tgoto** is assumed to be used for the two-parameter cursor positioning capability, *termcap* applications also use it for single-parameter capabilities.

  Doing so reveals a quirk in **tgoto**: most hardware terminals use cursor addressing with *row* first, but the original developers of the *termcap* interface chose to put the *col* (column) parameter first.

The **tgoto** function swaps the order of its parameters. It does this even for calls requiring only a single parameter. In that case, the first parameter is merely a placeholder.

- Normally the *ncurses* library is compiled without full *termcap* support. In that case, **tgoto** uses an internal version of **tparm**(3X) (a more capable function).

  Because it uses **tparm** internally, **tgoto** is able to use some *terminfo* features, but not all. In particular, it allows only numeric parameters; **tparm** supports string parameters.

  However, **tparm** is not a *termcap* feature, and portable *termcap* applications should not rely upon its availability.

tputs is described in **curs_terminfo**(3X). It can retrieve capabilities by either *termcap* or *terminfo* code.

Global Variables

The variables **PC**, **UP** and **BC** are set by **tgetent** to the *terminfo* entry's data for **pad_char**, **cursor_up** and **backspace_if_not_bs**, respectively. **UP** is not used by *ncurses*. **PC** is used by **delay_output**(3X). **BC** is used by **tgoto** emulation. The variable **ospeed** is set by *ncurses* using a system-specific encoding to indicate the terminal's data rate.

Releasing Memory

The *termcap* functions provide no means of freeing memory, because legacy *termcap* implementations used only the buffer areas provided by the caller via **tgetent** and **tgetstr**. Those buffers are unused in *terminfo*.

By contrast, *terminfo* allocates memory. It uses **setupterm**(3X) to obtain the data used by **tgetent** and the functions that retrieve capability values. One could use

```
del_curterm(cur_term);
```

to free this memory, but there is an additional complication with *ncurses*. It uses a fixed-size pool of storage locations, one per value of the terminal name parameter given to **tgetent**. The *screen*(1) program relies upon this arrangement to improve its performance.

An application that uses only the *termcap* functions, not the higher level *curses* API, could release the memory using **del_curterm**(3X), because the pool is freed using other functions; see **curs_memleaks**(3X).

RETURN VALUE

The return values of **tgetent**, **tgetflag**, **tgetname**, and **tgetstr** are documented above.

tgoto returns **NULL** on error. Error conditions include:

- uninitialized state (**tgetent** was not called successfully),
- *cap* being a null pointer,
- *cap* referring to a canceled capability,
- *cap* being a capability with string-valued parameters (a *terminfo*-only feature), and
- *cap* being a capability with more than two parameters.

See **curs_terminfo**(3X) regarding **tputs**.

NOTES

ncurses compares only the first two characters of the *id* parameter of **tgetflag**, **tgetnum**, and **tgetstr** to the capability names in the database.

PORTABILITY

These functions are no longer standardized (and the variables never were); *ncurses* provides them to support legacy applications. They should not be used in new programs.

Standards

- X/Open Curses, Issue 4, Version 2 (1996), describes these functions, marking them as "TO BE WITHDRAWN".
- X/Open Curses, Issue 7 (2009) marks the *termcap* interface (along with **vwprintw** and **vwscanw**) as withdrawn.

Neither X/Open Curses nor the SVr4 man pages documented the return values of **tgetent** correctly, though all three shown here were in fact returned ever since SVr1. In particular, an omission in the X/Open Curses specification has been misinterpreted to mean that **tgetent** returns **OK** or **ERR**. Because the purpose of these functions is to provide compatibility with the *termcap* library, that is a defect in X/Open Curses, Issue 4, Version 2 rather than in *ncurses*.

Compatibility with BSD *termcap*

Externally visible variables are provided for support of certain *termcap* applications. However, their correct usage is poorly documented; for example, it is unclear when reading and writing them is meaningful. In particular, some applications are reported to declare and/or modify **ospeed**.

The constraint that only the first two characters of the *id* parameter are used escapes many application developers. The BSD *termcap* library did not require a trailing null character on the capability

identifier passed to **tgetstr**, **tgetnum**, and **tgetflag**. Some applications thus assume that the *termcap* interface does not require the trailing null character for the capability identifier.

- *ncurses* disallows matches by the *termcap* interface against extended capability names that are longer than two characters; see **user_caps**(5).

The BSD *termcap* function **tgetent** returns the text of a *termcap* entry in the buffer passed as an argument. This library, like other *terminfo* implementations, does not store terminal type descriptions as text. It sets the buffer contents to a null-terminated string.

Header File

This library includes a *termcap.h* header for compatibility with other implementations, but the header is rarely used because the other implementations are not strictly compatible.

HISTORY

Bill Joy originated a forerunner of *termcap* called "ttycap", dated September 1977, and released in 1BSD (March 1978). It used many of the same function names as the later *termcap*, such as **tgetent**, **tgetflag**, **tgetnum**, and **tgetstr**.

A clear descendant, the *termlib* library, followed in 2BSD (May 1979), adding **tgoto** and **tputs**. The former applied at that time only to cursor positioning capabilities, thus the overly specific name. Little changed in 3BSD (late 1979) except the addition of test programs and a *termlib* man page, which documented the API shown in section "SYNOPSIS" above.

4BSD (November 1980) renamed *termlib* to *termcap* and added another test program. The library remained much the same though 4.3BSD (June 1986). 4.4BSD-Lite (June 1994) refactored it, leaving the API unchanged.

Function prototypes were a feature of ANSI C (1989). The library long antedated the standard and thus provided no header file declaring them. Nevertheless, the BSD sources included two different *termcap.h* header files over time.

- One was used internally by **jove**(1) from 4.3BSD onward. It declared global symbols for the *termcap* variables that it used.

- The other appeared in 4.4BSD-Lite Release 2 (June 1995) as part of *libedit* (also known as the *editline* library). CSRG source history shows that this was added in mid-1992. The *libedit* header file was used internally as a convenience for compiling the *editline* library. It declared function prototypes, but no global variables. This header file was added to NetBSD's *termcap* library in mid-1994.

Meanwhile, GNU *termcap* began development in 1990. Its first release (1.0) in 1991 included a *termcap.h* header. Its second (1.1) in September 1992 modified the header to use *const* for the function prototypes in the header where one would expect the parameters to be read-only. BSD *termcap* did not. The prototype for **tputs** also differed, but in that instance, it was *libedit* that differed from BSD *termcap*.

GNU *termcap* 1.3 was bundled with *bash*(1) in mid-1993 to support the *readline*(3) library.

ncurses 1.8.1 (November 1993) provided a *termcap.h* file. It reflected influence from GNU *termcap* and **emacs**(1) (rather than **jove**(1)), providing the following interface:

- global symbols used by *emacs*,
- *const*-qualified function prototypes, and
- a prototype for **tparam**, a GNU *termcap* feature.

Later (in mid-1996) the **tparam** function was removed from *ncurses*. Any two of the four implementations thus differ, and programs that intend to work with all *termcap* library interfaces must account for that fact.

BUGS

If you call **tgetstr** to fetch **column_address** (**ch**) or any other parameterized string capability, be aware that it is returned in *terminfo* notation, not the older and not-quite-compatible *termcap* notation. This does not cause problems if all you do with it is call **tgoto** or **tparm**, which both parametrically expand *terminfo*-style string capabilities as *terminfo* does. (If *ncurses* is configured to support *termcap*, **tgoto** checks whether the string is *terminfo*-style by looking for "**%p**" parameters or "<...>" delays, and invokes a *termcap*-style parser if the string appears not to use *terminfo* syntax.)

Because *terminfo*'s syntax for padding in string capabilities differs from *termcap*'s, users can be surprised.

- **tputs("50")** in a *terminfo* system transmits "50" rather than busy-waiting for 50 milliseconds.
- However, if *ncurses* is configured to support *termcap*, it may also have been configured to support BSD-style padding.

  In that case, **tputs** inspects strings passed to it, looking for digits at the beginning of the string.

  tputs("50") in a *termcap* system may busy-wait for 50 milliseconds rather than transmitting "50".

termcap has nothing analogous to *terminfo*'s **set_attributes** (**sgr**) capability. One consequence is that *termcap* applications assume that "**me**" (equivalent to *terminfo*'s **exit_attribute_mode** (**sgr0**) capability) does not reset the alternate character set. *ncurses* checks for, and modifies the data shared with, the *termcap* interface to accommodate the latter's limitation in this respect.

SEE ALSO

curses(3X), **curs_terminfo**(3X), **putc**(3), **term_variables**(3X), **terminfo**(5)

https://invisible-island.net/ncurses/tctest.html