NAME

tic – the *terminfo* entry-description compiler

SYNOPSIS

tic [**–01CDGIKLNTUVWacfgqrstx**] [**–e** *names*] [**–o** *dir*] [**–Q**[*n*]] [**–R** *subset*] [**–v**[*n*]] [**–w**[*n*]] *file*

DESCRIPTION

The **tic** command translates a **terminfo** file from source format into compiled format. The compiled format is necessary for use with the library routines in **ncurses**(3NCURSES).

As described in **term**(5), the database may be either a directory tree (one file per terminal entry) or a hashed database (one record per entry). The **tic** command writes only one type of entry, depending on how it was built:

- For directory trees, the top-level directory, e.g., /usr/share/terminfo, specifies the location of the database.
- For hashed databases, a filename is needed. If the given file is not found by that name, but can be found by adding the suffix ".db", then that is used.

  The default name for the hashed database is the same as the default directory name (only adding a ".db" suffix).

In either case (directory or hashed database), **tic** will create the container if it does not exist. For a directory, this would be the "terminfo" leaf, versus a "terminfo.db" file.

The results are normally placed in the system terminfo database **/etc/terminfo**. The compiled terminal description can be placed in a different terminfo database. There are two ways to achieve this:

- First, you may override the system default either by using the **–o** option, or by setting the variable **TERMINFO** in your shell environment to a valid database location.
- Secondly, if **tic** cannot write in */etc/terminfo* or the location specified using your TERMINFO variable, it looks for the directory *$HOME/.terminfo* (or hashed database *$HOME/.terminfo.db)*; if that location exists, the entry is placed there.

Libraries that read terminfo entries are expected to check in succession

- a location specified with the TERMINFO environment variable,
- *$HOME/.terminfo*,
- directories listed in the TERMINFO_DIRS environment variable,
- a compiled-in list of directories (no default value), and
- the system terminfo database (*/etc/terminfo*).

OPTIONS

–0 restricts the output to a single line

–1 restricts the output to a single column

–a tells **tic** to retain commented-out capabilities rather than discarding them. Capabilities are commented by prefixing them with a period. This sets the **–x** option, because it treats the commented-out entries as user-defined names. If the source is termcap, accept the 2-character names required by version 6. Otherwise these are ignored.

–C Force source translation to termcap format. Note: this differs from the **–C** option of **infocmp**(1) in that it does not merely translate capability names, but also translates terminfo strings to termcap format. Capabilities that are not translatable are left in the entry under their terminfo names but commented out with two preceding dots. The actual format used incorporates some improvements for escaped characters from terminfo format. For a stricter BSD-compatible translation, add the **–K** option.

If this is combined with **–c**, **tic** makes additional checks to report cases where the terminfo values do not have an exact equivalent in termcap form. For example:

- **sgr** usually will not convert, because termcap lacks the ability to work with more than two parameters, and because termcap lacks many of the arithmetic/logical operators used in terminfo.
- capabilities with more than one delay or with delays before the end of the string will not convert completely.

−c tells **tic** to only check *file* for errors, including syntax problems and bad use-links. If you specify **−C** (**−I**) with this option, the code will print warnings about entries which, after use resolution, are more than 1023 (4096) bytes long. Due to a fixed buffer length in older termcap libraries, as well as buggy checking for the buffer length (and a documented limit in terminfo), these entries may cause core dumps with other implementations.

tic checks string capabilities to ensure that those with parameters will be valid expressions. It does this check only for the predefined string capabilities; those which are defined with the **−x** option are ignored.

−D tells **tic** to print the database locations that it knows about, and exit. The first location shown is the one to which it would write compiled terminal descriptions. If **tic** is not able to find a writable database location according to the rules summarized above, it will print a diagnostic and exit with an error rather than printing a list of database locations.

−e *names*
Limit writes and translations to the following comma-separated list of terminals. If any name or alias of a terminal matches one of the names in the list, the entry will be written or translated as normal. Otherwise no output will be generated for it. The option value is interpreted as a file containing the list if it contains a '/'. (Note: depending on how tic was compiled, this option may require **−I** or **−C**.)

−f Display complex terminfo strings which contain if/then/else/endif expressions indented for readability.

−G Display constant literals in decimal form rather than their character equivalents.

−g Display constant character literals in quoted form rather than their decimal equivalents.

−I Force source translation to terminfo format.

−K Suppress some longstanding ncurses extensions to termcap format, e.g., "\s" for space.

−L Force source translation to terminfo format using the long C variable names listed in **<term.h>**

−N Disable smart defaults. Normally, when translating from termcap to terminfo, the compiler makes a number of assumptions about the defaults of string capabilities **reset1_string**, **carriage_return**, **cursor_left**, **cursor_down**, **scroll_forward**, **tab**, **newline**, **key_backspace**, **key_left**, and **key_down**, then attempts to use obsolete termcap capabilities to deduce correct values. It also normally suppresses output of obsolete termcap capabilities such as **bs**. This option forces a more literal translation that also preserves the obsolete capabilities.

−o*dir* Write compiled entries to given database location. Overrides the TERMINFO environment variable.

−Q*n* Rather than show source in terminfo (text) format, print the compiled (binary) format in hexadecimal or base64 form, depending on the option's value:

1 hexadecimal

2 base64

3 hexadecimal and base64

−q Suppress comments and blank lines when showing translated source.

−R*subset*
Restrict output to a given subset. This option is for use with archaic versions of terminfo like those on SVr1, Ultrix, or HP/UX that do not support the full set of SVR4/XSI Curses terminfo; and outright broken ports like AIX 3.x that have their own extensions incompatible with

SVr4/XSI. Available subsets are "SVr1", "Ultrix", "HP", "BSD" and "AIX"; see **terminfo**(5) for details.

−r Force entry resolution (so there are no remaining tc capabilities) even when doing translation to termcap format. This may be needed if you are preparing a termcap file for a termcap library (such as GNU termcap through version 1.3 or BSD termcap through 4.3BSD) that does not handle multiple tc capabilities per entry.

−s Summarize the compile by showing the database location into which entries are written, and the number of entries which are compiled.

−T eliminates size-restrictions on the generated text. This is mainly useful for testing and analysis, since the compiled descriptions are limited (e.g., 1023 for termcap, 4096 for terminfo).

−t tells **tic** to discard commented-out capabilities. Normally when translating from terminfo to termcap, untranslatable capabilities are commented-out.

−U tells **tic** to not post-process the data after parsing the source file. Normally, it infers data which is commonly missing in older terminfo data, or in termcaps.

−V reports the version of ncurses which was used in this program, and exits.

−v*n* specifies that (verbose) output be written to standard error trace information showing **tic**'s progress.

The optional parameter *n* is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If ncurses is built without tracing support, the optional parameter is ignored. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.

The debug flag levels are as follows:

1	Names of files created and linked
2	Information related to the “use” facility
3	Statistics from the hashing algorithm
5	String-table memory allocations
7	Entries into the string-table
8	List of tokens encountered by scanner
9	All values computed in construction of the hash table

If the debug level *n* is not given, it is taken to be one.

−W By itself, the **−w** option will not force long strings to be wrapped. Use the **−W** option to do this.

If you specify both **−f** and **−W** options, the latter is ignored when **−f** has already split the line.

−w*n* specifies the width of the output. The parameter is optional. If it is omitted, it defaults to 60.

−x Treat unknown capabilities as user-defined (see **user_caps(5)**). That is, if you supply a capability name which **tic** does not recognize, it will infer its type (boolean, number or string) from the syntax and make an extended table entry for that. User-defined capability strings whose name begins with “k” are treated as function keys.

PARAMETERS

file contains one or more **terminfo** terminal descriptions in source format [see **terminfo**(5)]. Each description in the file describes the capabilities of a particular terminal.

If *file* is “-”, then the data is read from the standard input. The *file* parameter may also be the path of a character-device.

PROCESSING

All but one of the capabilities recognized by **tic** are documented in **terminfo**(5). The exception is the **use** capability.

When a **use**=*entry*−*name* field is discovered in a terminal entry currently being compiled, **tic** reads in the binary from **/etc/terminfo** to complete the entry. (Entries created from *file* will be used first. **tic** duplicates the capabilities in *entry−name* for the current entry, with the exception of those capabilities that explicitly are defined in the current entry.

When an entry, e.g., **entry_name_1**, contains a **use=***entry_name_2* field, any canceled capabilities in *entry_name_2* must also appear in **entry_name_1** before **use=** for these capabilities to be canceled in **entry_name_1**.

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 512 bytes. Terminal names exceeding the maximum alias length (32 characters on systems with long filenames, 14 characters otherwise) will be truncated to the maximum alias length and a warning message will be printed.

HISTORY

System V Release 2 provided a **tic** utility. It accepted a single option: **−v** (optionally followed by a number). According to Ross Ridge's comment in *mytinfo*, this version of **tic** was unable to represent cancelled capabilities.

System V Release 3 provided a different **tic** utility, written by Pavel Curtis, (originally named "compile" in *pcurses*). This added an option **−c** to check the file for errors, with the caveat that errors in "use=" links would not be reported. System V Release 3 documented a few warning messages which did not appear in *pcurses*. While the program itself was changed little as development continued with System V Release 4, the table of capabilities grew from 180 (*pcurses*) to 464 (Solaris).

In early development of ncurses (1993), Zeyd Ben-Halim used the table from *mytinfo* to extend the *pcurses* table to 469 capabilities (456 matched SVr4, 8 were only in SVr4, 13 were not in SVr4). Of those 13, 11 were ultimately discarded (perhaps to match the draft of X/Open Curses). The exceptions were **memory_lock_above** and **memory_unlock** (see **user_caps**(5)).

Eric Raymond incorporated parts of *mytinfo* into ncurses to implement the termcap-to-terminfo source conversion, and extended that to begin development of the corresponding terminfo-to-termcap source conversion, Thomas Dickey completed that development over the course of several years.

In 1999, Thomas Dickey added the **−x** option to support user-defined capabilities.

In 2010, Roy Marples provided a **tic** program and terminfo library for NetBSD. This implementation adapts several features from ncurses, including **tic**'s **−x** option.

The **−c** option tells **tic** to check for problems in the terminfo source file. Continued development provides additional checks:

- *pcurses* had 8 warnings
- ncurses in 1996 had 16 warnings
- Solaris (SVr4) curses has 28 warnings
- NetBSD tic in 2019 has 19 warnings.
- ncurses in 2019 has 96 warnings

The checking done in ncurses' **tic** helps with the conversion to termcap, as well as pointing out errors and inconsistencies. It is also used to ensure consistency with the user-defined capabilities. There are 527 distinct capabilities in ncurses' terminal database; 128 of those are user-defined.

PORTABILITY

X/Open Curses, Issue 7 (2009) provides a brief description of **tic**. It lists one option: **−c**. The omission of **−v** is unexpected. The change history states that the description is derived from True64 UNIX. According to its manual pages, that system also supported the **−v** option.

Shortly after Issue 7 was released, Tru64 was discontinued. As of 2019, the surviving implementations of **tic** are SVr4 (AIX, HP-UX and Solaris), ncurses and NetBSD curses.

The X/Open rationale states that some implementations of **tic** read terminal descriptions from the standard input if the *file* parameter is omitted. None of these implementations do that. Further, it comments that some may choose to read from "./terminfo.src" but that is obsolescent behavior from SVr2, and is not (for example) a documented feature of SVr3.

COMPATIBILITY

There is some evidence that historic **tic** implementations treated description fields with no whitespace in them as additional aliases or short names. This **tic** does not do that, but it does warn when description fields may be treated that way and check them for dangerous characters.

EXTENSIONS

Unlike the SVr4 **tic** command, this implementation can actually compile termcap sources. In fact, entries in terminfo and termcap syntax can be mixed in a single source file. See **terminfo**(5) for the list of termcap names taken to be equivalent to terminfo names.

The SVr4 manual pages are not clear on the resolution rules for **use** capabilities. This implementation of **tic** will find **use** targets anywhere in the source file, or anywhere in the file tree rooted at **TERMINFO** (if **TERMINFO** is defined), or in the user's *$HOME/.terminfo* database (if it exists), or (finally) anywhere in the system's file tree of compiled entries.

The error messages from this **tic** have the same format as GNU C error messages, and can be parsed by GNU Emacs's compile facility.

The **−0**, **−1**, **−C**, **−G**, **−I**, **−N**, **−R**, **−T**, **−V**, **−a**, **−e**, **−f**, **−g**, **−o**, **−r**, **−s**, **−t** and **−x** options are not supported under SVr4. The SVr4 **−c** mode does not report bad "use=" links.

System V does not compile entries to or read entries from your *$HOME/.terminfo* database unless TERMINFO is explicitly set to it.

FILES

/etc/terminfo/?/*

Compiled terminal description database.

SEE ALSO

infocmp(1), **captoinfo**(1), **infotocap**(1), **toe**(1), **ncurses**(3NCURSES), **term**(5). **terminfo**(5). **user_caps**(5).

This describes **ncurses** version 6.1 (patch 20190803).

AUTHOR

Eric S. Raymond <esr@snark.thyrsus.com> and
Thomas E. Dickey <dickey@invisible-island.net>