NAME
>   term – format of compiled term file.

SYNOPSIS
>   **term**

DESCRIPTION
>   **STORAGE LOCATION**
>
>   Compiled terminfo descriptions are placed under the directory **/usr/share/terminfo**. Two configurations
>   are supported (when building the ncurses libraries):
>
>   **directory tree**
>   A two-level scheme is used to avoid a linear search of a huge UNIX system directory: **/usr/share/ter-**
>   **minfo/c/name** where *name* is the name of the terminal, and *c* is the first character of *name*. Thus,
>   *act4* can be found in the file **/usr/share/terminfo/a/act4**. Synonyms for the same terminal are imple-
>   mented by multiple links to the same compiled file.
>
>   **hashed database**
>   Using Berkeley database, two types of records are stored: the terminfo data in the same format as
>   stored in a directory tree with the terminfo's primary name as a key, and records containing only
>   aliases pointing to the primary name.
>
>   If built to write hashed databases, ncurses can still read terminfo databases organized as a directory
>   tree, but cannot write entries into the directory tree. It can write (or rewrite) entries in the hashed
>   database.
>
>   ncurses distinguishes the two cases in the TERMINFO and TERMINFO_DIRS environment variable
>   by assuming a directory tree for entries that correspond to an existing directory, and hashed database
>   otherwise.
>
>   **STORAGE FORMAT**
>
>   The format has been chosen so that it will be the same on all hardware. An 8 or more bit byte is assumed,
>   but no assumptions about byte ordering or sign extension are made.
>
>   The compiled file is created with the **tic** program, and read by the routine *setupterm*. The file is divided
>   into six parts: the header, terminal names, boolean flags, numbers, strings, and string table.
>
>   The header section begins the file. This section contains six short integers in the format described below.
>   These integers are
>
>   (1) the magic number (octal 0432);
>
>   (2) the size, in bytes, of the names section;
>
>   (3) the number of bytes in the boolean section;
>
>   (4) the number of short integers in the numbers section;
>
>   (5) the number of offsets (short integers) in the strings section;
>
>   (6) the size, in bytes, of the string table.
>
>   Short integers are stored in two 8-bit bytes. The first byte contains the least significant 8 bits of the value,
>   and the second byte contains the most significant 8 bits. (Thus, the value represented is 256*second+first.)
>   The value -1 is represented by the two bytes 0377, 0377; other negative values are illegal. This value gener-
>   ally means that the corresponding capability is missing from this terminal. Note that this format corre-
>   sponds to the hardware of the VAX and PDP-11 (that is, little-endian machines). Machines where this does
>   not correspond to the hardware must read the integers as two bytes and compute the little-endian value.
>
>   The terminal names section comes next. It contains the first line of the terminfo description, listing the var-
>   ious names for the terminal, separated by the '|' character. The section is terminated with an ASCII NUL
>   character.
>
>   The boolean flags have one byte for each flag. This byte is either 0 or 1 as the flag is present or absent.
>   The capabilities are in the same order as the file <term.h>.

Between the boolean section and the number section, a null byte will be inserted, if necessary, to ensure that the number section begins on an even byte (this is a relic of the PDP-11's word-addressed architecture, originally designed in to avoid IOT traps induced by addressing a word on an odd byte boundary). All short integers are aligned on a short word boundary.

The numbers section is similar to the flags section. Each capability takes up two bytes, and is stored as a little-endian short integer. If the value represented is -1, the capability is taken to be missing.

The strings section is also similar. Each capability is stored as a short integer, in the format above. A value of -1 means the capability is missing. Otherwise, the value is taken as an offset from the beginning of the string table. Special characters in ˆX or \c notation are stored in their interpreted form, not the printing representation. Padding information $<nn> and parameter information %x are stored intact in uninterpreted form.

The final section is the string table. It contains all the values of string capabilities referenced in the string section. Each string is null terminated.

EXTENDED STORAGE FORMAT

The previous section describes the conventional terminfo binary format. With some minor variations of the offsets (see PORTABILITY), the same binary format is used in all modern UNIX systems. Each system uses a predefined set of boolean, number or string capabilities.

The ncurses libraries and applications support extended terminfo binary format, allowing users to define capabilities which are loaded at runtime. This extension is made possible by using the fact that the other implementations stop reading the terminfo data when they have reached the end of the size given in the header. ncurses checks the size, and if it exceeds that due to the predefined data, continues to parse according to its own scheme.

First, it reads the extended header (5 short integers):

(1)   count of extended boolean capabilities

(2)   count of extended numeric capabilities

(3)   count of extended string capabilities

(4)   size of the extended string table in bytes.

(5)   last offset of the extended string table in bytes.

Using the counts and sizes, ncurses allocates arrays and reads data for the extended capabilties in the same order as the header information.

The extended string table contains values for string capabilities. After the end of these values, it contains the names for each of the extended capabilities in order, e.g., booleans, then numbers and finally strings.

PORTABILITY

Note that it is possible for *setupterm* to expect a different set of capabilities than are actually present in the file. Either the database may have been updated since *setupterm* has been recompiled (resulting in extra unrecognized entries in the file) or the program may have been recompiled more recently than the database was updated (resulting in missing entries). The routine *setupterm* must be prepared for both possibilities – this is why the numbers and sizes are included. Also, new capabilities must always be added at the end of the lists of boolean, number, and string capabilities.

Despite the consistent use of little-endian for numbers and the otherwise self-describing format, it is not wise to count on portability of binary terminfo entries between commercial UNIX versions. The problem is that there are at least three versions of terminfo (under HP-UX, AIX, and OSF/1) which diverged from System V terminfo after SVr1, and have added extension capabilities to the string table that (in the binary format) collide with System V and XSI Curses extensions. See **terminfo**(5) for detailed discussion of terminfo source compatibility issues.

EXAMPLE

As an example, here is a hex dump of the description for the Lear-Siegler ADM-3, a popular though rather stupid early terminal:

```
adm3a|lsi adm3a,
    am,
    cols#80, lines#24,
    bel=^G, clear= 32$<1>, cr=^M, cub1=^H, cud1=^J,
    cuf1=^L, cup=\E=%p1%{32}%+%c%p2%{32}%+%c, cuu1=^K,
    home=^^, ind=^J,


0000   1a 01 10 00 02 00 03 00   82 00 31 00 61 64 6d 33   ........ ..1.adm3
0010   61 7c 6c 73 69 20 61 64   6d 33 61 00 00 01 50 00   a|lsi ad m3a...P.
0020   ff ff 18 00 ff ff 00 00   02 00 ff ff ff ff 04 00   ........ ........
0030   ff ff ff ff ff ff ff ff   0a 00 25 00 27 00 ff ff   ........ ..%.'...
0040   29 00 ff ff ff ff 2b 00   ff ff 2d 00 ff ff ff ff   ).....+. ..-.....
0050   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0060   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0070   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0080   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0090   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
00a0   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
00b0   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
00c0   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
00d0   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
00e0   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
00f0   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0100   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0110   ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff   ........ ........
0120   ff ff ff ff ff ff 2f 00   07 00 0d 00 1a 24 3c 31   ....../. .....$<1
0130   3e 00 1b 3d 25 70 31 25   7b 33 32 7d 25 2b 25 63   >..=%p1% {32}%+%c
0140   25 70 32 25 7b 33 32 7d   25 2b 25 63 00 0a 00 1e   %p2%{32} %+%c....
0150   00 08 00 0c 00 0b 00 0a   00                        ........ .
```

LIMITS

Some limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

FILES

/usr/share/terminfo/*/*   compiled terminal capability data base

SEE ALSO

curses(3X), **terminfo**(5).

AUTHORS

Thomas E. Dickey
extended terminfo format for ncurses 5.0
hashed database support for ncurses 5.6

Eric S. Raymond