## NAME

infocmp – compare or print out terminfo descriptions

## SYNOPSIS

**/usr/bin/infocmp [–d] [–c] [–n] [–I] [–L] [–C] [–r] [–u] [–s d | i | l | c] [–v] [-V] [–1] [–w** *width***] [–A** *directory***] [–B** *directory***] [***termname* . . .**]**

## DESCRIPTION

**infocmp** compares a binary **terminfo** entry with other terminfo entries, rewrites a **terminfo** description to take advantage of the **use=** terminfo field, or prints out a **terminfo** description from the binary file ( **term** ) in a variety of formats. It displays boolean fields first, then numeric fields, followed by the string fields. If no options are specified and zero, or one *termname* is specified, the –**I** option is assumed. If more than one *termname* is specified, the –**d** option is assumed.

## OPTIONS

The –**d ,** –**c ,** and –**n** options can be used for comparisons. **infocmp** compares the **terminfo** description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termname*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: **F** for boolean variables, –**1** for integer variables, and **NULL** for string variables.

**–d** Produce a list of each capability that is different between two entries. This option is useful to show the difference between two entries, created by different people, for the same or similar terminals.

**–c** Produce a list of each capability that is common between two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the –**u** option is worth using.

**–n** Produce a list of each capability that is in neither entry. If no *termname* is given, the environment variable **TERM** will be used for both of the *termname*s. This can be used as a quick check to see if anything was left out of a description.

The –**I ,** –**L ,** and –**C** options will produce a source listing for each terminal named.

**–I** Use the **terminfo** names.

**–L** Use the long C variable name listed in **< term.h >**.

**–C** Use the **termcap** names. The source produced by the –**C** option may be used directly as a **termcap** entry, but not all of the parameterized strings may be changed to the **termcap** format. **infocmp** will attempt to convert most of the parameterized information, but anything not converted will be plainly marked in the output and commented out. These should be edited by hand.

**–r** When using –**C ,** put out all capabilities in **termcap** form.

If no *termname* is given, the environment variable **TERM** will be used for the terminal name.

All padding information for strings will be collected together and placed at the beginning of the string where **termcap** expects it. Mandatory padding (padding information with a trailing '/') will become optional.

All **termcap** variables no longer supported by **terminfo ,** but are derivable from other **terminfo** variables, will be displayed. Not all **terminfo** capabilities will be translated; only those variables which were part of **termcap** will normally be displayed. Specifying the **−r** option will take off this restriction, allowing all capabilities to be displayed in **termcap** form.

Note that because padding is collected to the beginning of the capability, not all capabilities are displayed. Mandatory padding is not supported. Because **termcap** strings are not as flexible, it is not always possible to convert a **terminfo** string capability into an equivalent **termcap** format. A subsequent conversion of the **termcap** file back into **terminfo** format will not necessarily reproduce the original **terminfo** source.

Some common **terminfo** parameter sequences, their **termcap** equivalents, and some terminal types which commonly have such sequences, are:

| **terminfo** | **termcap** | Representative Terminals |
|---|---|---|
| **%p1%c** | **%.** | adm |
| **%p1%d** | **%d** | hp, ANSI standard, vt100 |
| **%p1%'x'%+%c** | **%+x** | concept |
| **%i** | **%i** | ANSI standard, vt100 |
| **%p1%?%'x'%>%t%p1%'y'%+%;** | **%>xy** | concept |
| **%p2** is printed before **%p1** | **%r** | hp |

**−u** Produce a **terminfo** source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals' *termname*s. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with **use=** fields for the other terminals. In this manner, it is possible to retrofit generic **terminfo** entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times, or by different people so that each description is a full description, using **infocmp** will show what can be done to change one description to be relative to the other.

A capability is displayed with an at-sign (@) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value is displayed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability gives a different value for that capability.

The order of the other *termname* entries is significant. Since the **terminfo** compiler **tic** does a left-to-right scan of the capabilities, specifying two **use=** entries that contain differing entries for the same capabilities will produce different results, depending on the order in which the entries are given. **infocmp** will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability *after* a **use=** entry that contains, it will cause the second specification to be ignored. Using **infocmp** to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying superfluous **use=** fields. **infocmp** will flag any superfluous **use=** fields.

**−s** Sorts the fields within each type according to the argument below:

- **d** Leave fields in the order that they are stored in the **terminfo** database.
- **i** Sort by **terminfo** name.
- **l** Sort by the long C variable name.
- **c** Sort by the **termcap** name.

If the −**s** option is not given, the fields are sorted alphabetically by the **terminfo** name within each type, except in the case of the −**C** or the −**L** options, which cause the sorting to be done by the **termcap** name or the long C variable name, respectively.

**−v** Print out tracing information on standard error as the program runs.

**−V** Print out the version of the program in use on standard error and exit.

**−1** Print the fields one to a line. Otherwise, the fields are printed several to a line to a maximum width of 60 characters.

**−w** *width* Changes the output to *width* characters.

The location of the compiled **terminfo** database is taken from the environment variable **TERMINFO**. If the variable is not defined, or the terminal is not found in that location, the system **terminfo** database, usually in **/usr/share/lib/terminfo**, is used. The options −**A** and −**B** may be used to override this location.

**−A** *directory* Set **TERMINFO** for the first *termname*.

**−B** *directory* Set **TERMINFO** for the other *termname*s. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

## FILES

**/usr/share/lib/terminfo/?/*** Compiled terminal description database.

## SEE ALSO

**captoinfo**(1M), **tic**(1M), **curses**(3X), **terminfo**(4)