NAME

bibclean – prettyprint and syntax check BibTeX and Scribe bibliography data base files

SYNOPSIS

bibclean [**–author**] [**–error-log** *filename*]
[**–help**] [**'–?'**] [**–[no-]check-values**] [**–[no-]delete-empty-fields**]
[**–[no-]file-position**] [**–[no-]fix-initials**] [**–[no-]fix-names**]
[**–[no-]par-breaks**] [**–[no-]print-patterns**] [**–[-no]read-init-files** *filename*]
[**–[no-]remove-OPT-prefixes**] [**–[no-]scribe**] [**–[no-]trace-file-opening**]
[**–[no-]warnings**] [**–version**]
<infile or *bibfile1 bibfile2 bibfile3 . . .*
>outfile

All options can be abbreviated to a unique leading prefix.

An explicit file name of "–" represents standard input; it is assumed if no input files are specified.

On VAX VMS and IBM PC DOS, the leading "–" on option names may be replaced by a slash, "/"; however, the "–" option prefix is always recognized.

DESCRIPTION

bibclean prettyprints input BIBTEX files to *stdout*, and checks the brace balance and bibliography entry syntax as well. It can be used to detect problems in BIBTEX files that sometimes confuse even BIBTEX itself, and importantly, can be used to normalize the appearance of collections of BIBTEX files.

Here is a summary of the formatting actions:

- BIBTEX items are formatted into a consistent structure with one *key = "value"* pair per line, and the initial @ and trailing right brace in column 1.
- Tabs are expanded into blank strings; their use is discouraged because they inhibit portability, and can suffer corruption in electronic mail.
- Long string values are split at a blank and continued onto the next line with leading indentation.
- A single blank line separates adjacent bibliography entries.
- Text outside BIBTEX entries is passed through verbatim.
- Outer parentheses around entries are converted to braces.
- Personal names in *author* and *editor* field values are normalized to the form "P. D. Q. Bach", from "P.D.Q. Bach" and "Bach, P.D.Q.".
- Hyphen sequences in page numbers are converted to en-dashes.
- Month values are converted to standard BIBTEX string abbreviations.
- In titles, sequences of upper-case characters at brace level zero are braced to protect them from being converted to lower-case letters by some bibliography styles.
- ISBN (International Standard Book Number) and ISSN (International Standard Serial Number) entry values are examined to verify the checksums of each listed number.

The standardized format of the output of **bibclean** facilitates the later application of simple filters, such as **bibextract**(1), **bibindex**(1), **biblook**(1), **bibsort**(1), **citefind**(1), and **citetags**(1), to process the text, and also is the one expected by the GNU Emacs BIBTEX support functions.

OPTIONS

Command-line switches may be abbreviated to a unique leading prefix, and letter case is *not* significant. All options are parsed before any input bibliography files are read, no matter what their order on the command line. Options that correspond to a yes/no setting of a flag have a form with a prefix "no-" to set the flag to *no*. For such options, the last setting determines the flag value used. This is significant when options are also specified in initialization files (see the **INITIALIZATION FILES** manual section).

–author
Display an author credit on *stderr*. Sometimes an executable program is separated from its documentation and source code; this option provides a way to recover from that.

–error-log *filename*
Redirect *stderr* to the indicated file, which will then contain all of the error and warning messages. This option is provided for those systems that have difficulty redirecting *stderr*.

–help or **–?**
Display a help message on *stderr*, giving a usage description, similar to this section of the manual pages.

–init-file *filename*
Provide an explicit value pattern initialization file. It will be processed *after* any system-wide and job-wide initialization files found on the **PATH** (for VAX VMS, **SYS$SYSTEM**) and **BIBINPUTS** search paths, respectively, and may override them. It in turn may be overridden by a subsequent file-specific initialization file. The initialization file name can be changed at compile time, but defaults to *.bibcleanrc* on UNIX, and to *bibclean.ini* elsewhere. For further details, see the **INITIALIZATION FILES** manual section.

–[no-]check-values
With the positive form, apply heuristic pattern matching to value fields in order to detect possible errors (e.g. “*year = "192"*” instead of “*year = "1992"*”), and issue warnings when unexpected patterns are found.

This checking is usually beneficial, but if it produces too many bogus warnings for a particular bibliography file, you can disable it with the negative form of this option. Default: *yes*.

–[no-]delete-empty-fields
With the positive form, remove all key/value pairs for which the value is an empty string. This is helpful in cleaning up bibliographies generated from text editor templates. Compare this option with **–[no-]remove-OPT-prefixes** described below. Default: *no*.

–[no-]file-position
With the positive form, give detailed file position information in warning and error messages. Default: *no*.

–[no-]fix-initials
With the positive form, insert a space after a period following author initials. Default: *yes*.

–[no-]fix-names
With the positive form, reorder *author* and *editor* name lists to remove commas at brace level zero, placing first names or initials before last names. Default: *yes*.

–[no-]par-breaks
With the negative form, a paragraph break (either a formfeed, or a line containing only spaces) is not permitted in value strings, or between key/value pairs. This may be useful to quickly trap runaway strings arising from mismatched delimiters. Default: *yes*.

–[no-]print-patterns
With the positive form, print the value patterns read from initialization files as they are added to internal tables. Use this option to check newly-added patterns, or to see what patterns are being used.

When **bibclean** is compiled with native pattern-matching code (the default), these patterns are the ones that will be used in checking value strings for valid syntax, and all of them are specified in initialization files, rather than hard-coded into the program. For further details, see the **INITIALIZATION FILES** manual section. Default: *no*.

–[no-]read-init-files
With the negative form, suppress loading of system-, user-, and file-specific initialization files. Initializations will come *only* from those files explicitly given by **–init-file** *filename* options. Default: *yes*.

–[no-]remove-OPT-prefixes	With the positive form, remove the "OPT" prefix from each key name where the corresponding value is *not* an empty string. The prefix "OPT" must be entirely in upper-case to be recognized. This option is for bibliographies generated with the help of the GNU Emacs BIBTEX editing support, which generates templates with optional fields identified by the "OPT" prefix. Although the function *M-x bibtex-remove-OPT* normally bound to the keystrokes *C-c C-o* does the job, users often forget, with the result that BIBTEX does not recognize the key name, and ignores the value string. Compare this option with **–[no-]delete-empty-fields** described above. Default: *no*.
–[no-]scribe	With the positive form, accept input syntax conforming to the SCRIBE document system. The output will be converted to conform to BIBTEX syntax. See the **SCRIBE BIBLIOGRAPHY FORMAT** manual section for further details. Default: *no*.
–[no-]trace-file-opening	With the positive form, log in the error log file the names of all files which **bibclean** attempts to open. Use this option to identify where initialization files are located.
–[no-]warnings	With the positive form, allow all warning messages. The negative form is *not* recommended since it may mask problems that should be repaired. Default: *yes*.
–version	Display the program version number on *stderr*. This will also include an indication of who compiled the program, the host name on which it was compiled, the time of compilation, and the type of string value matching code selected, when that information is available to the compiler.

ERROR RECOVERY AND WARNINGS

When **bibclean** detects an error, it issues an error message to both *stderr* and *stdout*. That way, the user is clearly notified, and the output bibliography also contains the message at the point of error.

Error messages begin with a distinctive pair of queries, ??, beginning in column 1, followed by the input file name and line number. If the **–file-position** option was specified, they also contain the input and output positions of the current file, entry, and value. Each position includes the file byte number, the line number, and the column number. In the event of a runaway string argument, the entry and value positions should precisely pinpoint the erroneous bibliography entry, and the file positions will indicate where it was detected, which may be rather later in the files.

Warning messages identify possible problems, and are therefore sent only to *stderr*, and not to *stdout*, so they never appear in the output file. They are identified by a distinctive pair of percents, %%, beginning in column 1, and as with error messages, may be followed by file position messages if the **–file-position** option was specified.

For convenience, the first line of each error and warning message sent to *stderr* is formatted according to the expectations of the GNU Emacs *next-error* command. You can invoke **bibclean** with the Emacs *M-x compile<RET>bibclean filename.bib >filename.new* command, then use the *next-error* command, normally bound to *C-x ‘* (that's a grave, or back, accent), to move to the location of the error in the input file.

If error messages are ignored, and left in the output bibliography file, they will precipitate an error when the bibliography is next processed with BIBTEX.

After issuing an error message, **bibclean** then resynchronizes its input by copying it verbatim to *stdout* until a new bibliography entry is recognized on a line in which the first non-blank character is an at-sign (@). This ensures that nothing is lost from the input file(s), allowing corrections to be made in either the input or the output files. However, if **bibclean** detects an internal error in its data structures, it will terminate abruptly without further input or output processing; this kind of error should never happen, and if it does, it should be reported immediately to the author of the program. Errors in initialization files, and running out of dynamic memory, will also immediately terminate **bibclean**.

INITIALIZATION FILES

bibclean can be compiled with one of three different types of pattern matching; the choice is made by the installer at compile time:

- The original version uses explicit hand-coded tests of value-string syntax.
- The second version uses regular-expression pattern-matching host library routines together with regular-expression patterns that come entirely from initialization files.
- The third version uses special patterns that come entirely from initialization files.

The second and third versions are the ones of most interest here, because they allow the user to control what values are considered acceptable. However, command-line options can also be specified in initialization files, no matter which pattern matching choice was selected.

When **bibclean** starts, it searches for initialization files, finding the first one in the system executable program search path (on UNIX and IBM PC DOS, **PATH**) and the first one in the **BIBINPUTS** search path, and processes them in turn. Then, when command-line arguments are processed, any additional files specified by **–init-file** *filename* options are also processed. Finally, immediately before each *named* bibliography file is processed, an attempt is made to process an initialization file with the same name, but with the extension changed to *.ini*. This scheme permits system-wide, user-wide, session-wide, and file-specific initialization files to be supported.

When input is taken from *stdin*, there is no file-specific initialization.

For precise control, the **–no-init-files** option suppresses all initialization files except those explicitly named by **–init-file** *filename* options, either on the command line, or in requested initialization files.

Recursive execution of initialization files with nested **–init-file** options is permitted; if the recursion is circular, **bibclean** will finally get a non-fatal initialization file open failure after opening too many files. This terminates further initialization file processing. As the recursion unwinds, the files are all closed, then execution proceeds normally.

An initialization file may contain empty lines, comments from percent to end of line (just like TEX), option switches, and key/pattern or key/pattern/message assignments. Leading and trailing spaces are ignored. This is best illustrated by a short example:

```
% This is a small bibclean initialization file

-init-file /u/math/bib/.bibcleanrc %% departmental patterns

chapter = "\"D\""                  %% 23

pages   = "\"D--D\""               %% 23--27

volume  = "\"D \\an\\d D\""        %% 11 and 12

year    = \
   "\"dddd, dddd, dddd\"" \
   "Multiple years specified."      %% 1989, 1990, 1991

-no-fix-names   %% do not modify author/editor lists
```

Long logical lines can be split into multiple physical lines by breaking at a backslash-newline pair; the backslash-newline pair is discarded. This processing happens while characters are being read, before any further interpretation of the input stream.

Each logical line must contain a complete option (and its value, if any), or a complete key/pattern pair, or a key/pattern/message triple.

Comments are stripped during the parsing of the key, pattern, and message values. The comment start symbol is not recognized inside quoted strings, so it can be freely used in such strings.

Comments on logical lines that were input as multiple physical lines via the backslash-newline convention must appear on the *last* physical line; otherwise, the remaining physical lines will become part of the comment.

Pattern strings must be enclosed in quotation marks; within such strings, a backslash starts an escape mechanism that is commonly used in UNIX software. The recognized escape sequences are:

\a	alarm bell (octal 007)
\b	backspace (octal 010)
\f	formfeed (octal 014)
\n	newline (octal 012)
\r	carriage return (octal 015)
\t	horizontal tab (octal 011)
\v	vertical tab (octal 013)
\ooo	character number octal *ooo* (e.g **\012** is linefeed)
\0xhh	character number hexadecimal *hh* (e.g. **\0x0a** is linefeed) **\0Xhh** is character number hexadecimal *hh* (e.g. **\0X0A** is linefeed)

Backslash followed by any other character produces just that character. Thus, \% gets a literal percent into a string (preventing its interpretation as a comment), \" produces a quotation mark, and \\ produces a single backslash.

Use of an ASCII NUL *(\0)* in a string will terminate it; this is a feature of the C programming language in which **bibclean** is implemented.

Key/pattern pairs can be separated by arbitrary space, and optionally, either an equals or colon functioning as an assignment operator. Thus, the following are equivalent:

```
pages="\"D--D\""
pages:"\"D--D\""
pages "\"D--D\""
  pages = "\"D--D\""
  pages : "\"D--D\""
pages   "\"D--D\""
```

Each key name can have an arbitrary number of patterns associated with it; however, they must be specified in separate key/pattern assignments.

An empty pattern string causes previously-loaded patterns for that key name to be forgotten. This feature permits an initialization file to completely discard patterns from earlier initialization files.

Patterns for value strings are represented in a tiny special-purpose language that is both convenient and suitable for bibliography value string syntax checking. While not as powerful as the language of regular-expression patterns, its parsing can be portably implemented in less than 3% of the code in a widely-used regular-expression parser (the GNU **regexp** package).

The patterns are represented by the following special characters:

<space>	one or more spaces
a	exactly one letter
A	one or more letters
d	exactly one digit
D	one or more digits
w	exactly one word (one or more letters and digits)

W	one or more space-separated words, beginning and ending with a word
.	one ‘special’ character, one of the characters <space> ! # ( ) * + , - . / : ; ? [ ] ˜, a subset of punctuation characters that are typically used in string values
:	one or more ‘special’ characters
X	one or more ‘special’-separated words, beginning and ending with a word
\x	exactly one x (x is any character), possibly with an escape sequence interpretation given earlier
x	exactly the character x (x is anything but one of these pattern characters: a A d D w W . : <space> \ )

The **X** pattern character is very powerful, but generally inadvisable, since it will match almost anything likely to be found in a BibTeX value string. The reason for providing pattern matching on the value strings is to uncover possible errors, not mask them.

There is no provision for specifying ranges or repetitions of characters, but this can usually be done with separate patterns. It is a good idea to accompany the pattern with a comment showing the kind of thing it is expected to match. Here is a portion of an initialization file giving a few of the patterns used to match *number* value strings:

```
number  =       "\"D\""          %% 23
number  =       "\"A AD\""       %% PN LPS5001
number  =       "\"A D(D)\""     %% RJ 34(49)
number  =       "\"A D\""        %% XNSS 288811
number  =       "\"A D\\.D\""    %% Version 3.20
number  =       "\"A-A-D-D\""    %% UMIAC-TR-89-11
number  =       "\"A-A-D\""      %% CS-TR-2189
number  =       "\"A-A-D\\.D\"" %% CS-TR-21.7
```

For a bibliography that contains only *article* entries, this list should probably be reduced to just the first pattern, so that anything other than a digit string fails the pattern-match test. This is easily done by keeping bibliography-specific patterns in a corresponding file with extension *.ini*, since that file is read automatically.

You should be sure to use empty pattern strings in this pattern file to discard patterns from earlier initialization files.

The value strings passed to the pattern matcher contain surrounding quotes, so the patterns should also. However, you could use a pattern specification like "\"D" to match an initial digit string followed by anything else; the omission of the final quotation mark \" in the pattern allows the match to succeed without checking that the next character in the value string is a quotation mark.

Because the value strings are intended to be processed by TeX, the pattern matching ignores braces, and TeX control sequences, together with any space following those control sequences. Space around braces are preserved. This convention allows the pattern fragment *A-AD-D* to match the value string *TN-K\slash 27-70*, because the value is implicitly collapsed to *TN-K27-70* during the matching operation.

bibclean’s normal action when a string value fails to match any of the corresponding patterns is to issue a *warning* message something like this: *"Unexpected value in “year = "192"”*. In most cases, that is sufficient to alert the user to a problem. In some cases, however, it may be desirable to associate a different message with a particular pattern. This can be done by supplying a message string following the pattern string. Format items *%%* (single percent), *%e* (entry name), *%k* (key name), *%t* (citation tag), and *%v* (string value) are available to get current values expanded in the messages. Here is an example:

```
chapter = "\"D:D\"" "Colon found in ``%k = %v''" %% 23:2
```

To be consistent with other messages output by **bibclean**, the message string should *not* end with punctuation.

If you wish to make the message an error, rather than just a warning, begin it with a query (?), like this:

```
chapter = "\"D:D\"" "?Colon found in ``%k = %v''" %% 23:2
```

The query will not be included in the output message.

Escape sequences are supported in message strings, just as they are in pattern strings. You can use this to advantage for fancy things, such as terminal display mode control. If you rewrite the previous example as

```
chapter = "\"D:D\"" \
         "?\033[7mColon found in ``%k = %v''\033[0m" %% 23:2
```

the error message will appear in inverse video on display screens that support ANSI terminal control sequences. Such practice is not normally recommended, since it may have undesirable effects on some output devices. Nevertheless, you may find it useful for restricted applications.

For some types of bibliography keys, **bibclean** contains special-purpose code to supplement or replace the pattern matching:

- *ISBN* and *ISSN* fields are handled this way because their validation requires evaluation of checksums that cannot be expressed by simple patterns; no patterns are even used in these two cases.
- When **bibclean** is compiled with pattern-matching code support, *chapter*, *number*, *pages*, and *volume* values are checked only by pattern matching.
- *month* values are first checked against the standard BIBTEX month abbreviations, and only if no match is found are patterns then used.
- *year* values are first checked against patterns, then if no match is found, the year numbers are found and converted to integer values for testing against reasonable bounds.

Values for other keywords are checked only against patterns. You can provide patterns for *any* keyword you like, even ones **bibclean** does not already know about. New ones are simply added to an internal table that is searched for each string to be validated.

The special keyword, *tag*, represents the bibliographic citation tag. It can be given patterns, like any other keyword. Here is an initialization file pattern assignment that will match an author name, a colon, an alphabetic string, and a two-digit year:

```
tag = "A:Add"                  %% Knuth:TB86
```

Notice that no quotation marks are included in the pattern, because the citation tags are not quoted. You can use such patterns to help enforce uniform naming conventions for citation tags, which is increasingly important as your bibliography data base grows.

SCRIBE BIBLIOGRAPHY FORMAT

bibclean's support for the SCRIBE bibliography format is based on the syntax description in the SCRIBE Introductory User's Manual, 3rd Edition, May 1980. SCRIBE was originally developed by Brian Reid at Carnegie-Mellon University, and is now marketed by Unilogic, Ltd.

The BIBTEX bibliography format was strongly influenced by SCRIBE, and indeed, with care, it is possible to share bibliography files between the two systems. Nevertheless, there are some differences, so here is a summary of features of the SCRIBE bibliography file format:

(1) Letter case is not significant in keywords and entry names, but case is preserved in value strings.

(2) In key/value pairs, the key and value may be separated by one of three characters: =, /, or space. Space may optionally surround these separators.

(3) Value delimiters are any of these seven pairs: { } [ ] ( ) < > ' ' " " ‘ ‘

(4) Value delimiters may not be nested, even though with the first four delimiter pairs, nested balanced delimiters would be unambiguous.

(5) Delimiters can be omitted around values that contain only letters, digits, sharp (#), ampersand (&), period (.), and percent (%).

(6) Outside of delimited values, a literal at-sign (@) is represented by doubled at-signs (@@).

(7) Bibliography entries begin with @name, as for BIBTEX, but any of the seven SCRIBE value delimiter pairs may be used to surround the values in key/value pairs. As in (4), nested delimiters are forbidden.

(8) Arbitrary space may separate entry names from the following delimiters.

(9) @Comment is a special command whose delimited value is discarded. As in (4), nested delimiters are forbidden.

(10) The special form

@Begin{comment}
...
@End{comment}

permits encapsulating arbitrary text containing any characters or delimiters, other than "@End{comment}". Any of the seven delimiter pairs may be used around the word "comment" following the "@Begin" or "@End"; the delimiters in the two cases need not be the same, and consequently, "@Begin{comment}"/"@End{comment}" pairs may *not* be nested.

(11) The *key* keyword is required in each bibliography entry.

(12) A backslashed quote in a string will be assumed to be a TEX accent, and braced appropriately. While such accents do not conform to SCRIBE syntax, SCRIBE-format bibliographies have been found that appear to be intended for TEX processing.

Because of this loose syntax, **bibclean**'s normal error detection heuristics are less effective, and consequently, SCRIBE mode input is not the default; it must be explicitly requested.

SEE ALSO

bibextract(1), bibindex(1), biblook(1), bibsort(1), bibtex(1), citefind(1), citetags(1), latex(1), scribe(1), tex(1)

FILES

**.bib*	BIBTEX and SCRIBE bibliography data base files.
**.ini*	File-specific initialization files.
.bibcleanrc	UNIX system and user initialization files.
bibclean.ini	Non-UNIX system and user initialization files.
BIBINPUTS	Search path for user initialization files.
PATH	Search path for system initialization files on UNIX and IBM PC DOS>
SYS$SYSTEM	Search path for system initialization files on VAX VMS.

AUTHOR

Nelson H. F. Beebe
Center for Scientific Computing
Department of Mathematics
University of Utah
Salt Lake City, UT 84112
USA
Tel: +1 801 581 5254
FAX: +1 801 581 4148
Email: <beebe@math.utah.edu>