

The checkcites* script

Enrico Gregorio

Enrico dot Gregorio at univr dot it

Paulo Roberto Massa Cereda

cereda at users dot sf dot net

Contents

1 Introduction	1
2 How the script works	2
3 Usage	3
4 License	8

1 Introduction

checkcites is a Lua script written for the sole purpose of detecting unused or undefined references from both L^AT_EX auxiliary or bibliography files. We use the term *unused reference* to refer to the reference present in the bibliography file – with the .bib extension – but not cited in the .tex file. The term *undefined reference* is exactly the opposite, i.e., the item cited in the .tex file, but not present in the .bib file.

The original idea came from a question posted in the T_EX community at Stack Exchange about [how to check which bibliography entries were not used](#). We decided to write a script to check references. We opted for Lua, since it is a very straightforward language and it has an interpreter available on every modern T_EX distribution.

Attention!

checkcites is known to run with the most recent texlua and lua interpreters. Unfortunately, the code is incompatible with interpreters prior to the Lua 5.1 language specification.

*Version 2.1 from July 26, 2019.

2 How the script works

checkcites uses the generated auxiliary files to start the analysis. From version 2.0 on, the scripts supports two backends:

bibtex Default behavior, the script checks `.aux` files looking for citations, in the form of `\citation{a}`. For every `\citation` line found, checkcites will extract the citations and add them to a table, even for multiple citations separated by commas, like `\citation{a,b,c}`. The citation table contains no duplicate values. At the same time checkcites also looks for bibliography data, in the form of `\bibdata{a}`. Similarly, for every `\bibdata` line found, the script will extract the bibliography data and add them to a table, even if they are separated by commas, like `\bibdata{d,e,f}`. Again, no duplicate values are allowed. Stick with this backend if you are using BibTeX or BibLaTeX with the `backend=bibtex` package option.

biber With this backend, the script checks `.bcf` files (which are XML-based) looking for citations, in the form of `bcf:citekey` tags. For every tag found, checkcites will extract the corresponding values and add them to a table. The citation table contains no duplicate values. At the same time checkcites also looks for bibliography data, in the form of `bcf:datasource` tags. Similarly, for every tag found, the script will extract the bibliography data and add them to a table. Again, no duplicate values are allowed. Stick with this backend if you are using BibLaTeX with the default options or with the `backend=biber` option explicitly set.

Attention!

If `\citation{*}` (BibTeX) or simply `*` (BibLaTeX) is found, checkcites will issue a message telling that `\nocite{*}` is in the `.tex` document, but the script will do the check nonetheless.

Now, checkcites will extract all entries from the bibliography files found in the previous steps, regardless of which backend was used. For every element in the bibliography data table, the script will look for entries like `@BOOK`, `@ARTICLE` and so forth – we actually use pattern matching for this – and add their identifiers to a table. No duplicate values are allowed.

Attention!

If checkcites cannot find a certain bibliography file, the script ends. Make sure to put the correct name of the bibliography file in your `.tex` file.

Let there be A and B the sets of citations and references, respectively. In order to get all unused references in the `.bib` files, we compute the set difference:

$$B - A = \{x : x \in B, x \notin A\}.$$

Similarly, in order to get all undefined references in the `.tex` file, we compute the set difference:

$$A - B = \{x : x \in A, x \notin B\}.$$

If there are either unused or undefined references, checkcites will print them in a list format. In Section 3 there is a more complete explanation on how to use the script.

3 Usage

checkcites is very easy to use. First of all, let us define two files that will be used here to explain the script usage. Here is our sample bibliography file `example.bib`, with five fictional entries.

Bibliography file

```
@BOOK{foo:2012a,  
  title = {My Title One},  
  publisher = {My Publisher One},  
  year = {2012},  
  editor = {My Editor One},  
  author = {Author One}  
}  
  
@BOOK{foo:2012b,  
  title = {My Title Two},  
  publisher = {My Publisher Two},  
  year = {2012},  
  editor = {My Editor Two},  
  author = {Author Two}  
}  
  
@BOOK{foo:2012c,  
  title = {My Title Three},  
  publisher = {My Publisher Three},  
  year = {2012},  
  editor = {My Editor Three},  
  author = {Author Three}  
}  
  
@BOOK{foo:2012d,  
  title = {My Title Four},  
  publisher = {My Publisher Four},  
  year = {2012},  
  editor = {My Editor Four},  
  author = {Author Four}  
}  
  
@BOOK{foo:2012e,  
  title = {My Title Five},  
  publisher = {My Publisher Five},  
  year = {2012},  
  editor = {My Editor Five},  
  author = {Author Five}  
}
```

The second file is our main L^AT_EX document, `document.tex`. Observe that we will stick with Bib_TE_X for now and check BibL^AT_EX later on.

Main document

```
\documentclass{article}

\begin{document}

Hello world \cite{foo:2012a,foo:2012c},
how are you \cite{foo:2012f},
and goodbye \cite{foo:2012d,foo:2012a}.

\bibliographystyle{plain}
\bibliography{example}

\end{document}
```

Open a terminal and run `checkcites`:

```
$ checkcites
```

```

  _ _ _ _ _
 | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ |
 | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ |
 | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ |
```

```
checkcites.lua -- a reference checker script (v2.1)
Copyright (c) 2012, 2019, Enrico Gregorio, Paulo Roberto Massa Cereda
```

```
-----
I am sorry, but you have not provided any command line argument, including
files to check and options. Make sure to invoke the script with the actual
arguments. Refer to the user documentation if you are unsure of how this
tool works. The script will end now.
```

If you do not have `checkcites` installed with your T_EX distribution, you can run the standalone script `checkcites.lua` with either `texlua` or `lua`. We recommend to use `texlua`, as it is shipped with all the modern T_EX distributions:

```
$ texlua checkcites.lua
```

When you run `checkcites` without providing any argument to it, the a message error will appear. Do not panic! Try again with the `--help` flag:

```
$ checkcites --help
```

```

  _ _ _ | _ _ _ _ _ | _ _ _ | _ _ _ _ _
| _ _ | _ _ _ | _ _ ' _ _ | _ _ | _ _ _ _ |
| _ _ | _ _ | _ _ | _ _ | _ _ | _ _ | _ _ |
```

checkcites.lua -- a reference checker script (v2.1)

Copyright (c) 2012, 2019, Enrico Gregorio, Paulo Roberto Massa Cereda

Usage: checkcites.lua [[--all | --unused | --undefined] [--backend
<arg>] <file> [<file 2> ... <file n>] | --help | --version]

-a,--all list all unused and undefined references
-u,--unused list only unused references in your bibliography files
-U,--undefined list only undefined references in your TeX source file
-b,--backend <arg> set the backend-based file lookup policy
-h,--help print the help message
-v,--version print the script version

Unless specified, the script lists all unused and undefined references by default. Also, the default backend is set to "bibtex". Please refer to the user documentation for more details.

Since we are using BibTeX, we do not need to set up the backend! Simply provide the auxiliary file – the one with the .aux extension – which is generated when you compile your main .tex file. For example, if your main document is named foo.tex, you probably have a foo.aux file too. Let us compile our sample document document.tex:

```
$ pdflatex document.tex
```

After running pdflatex on our .tex file, there is now a document.aux file in our work directory.

Auxiliary file

```
\relax
\citation{foo:2012a}
\citation{foo:2012c}
\citation{foo:2012f}
\citation{foo:2012d}
\citation{foo:2012a}
\bibstyle{plain}
\bibdata{example}
```

Now we can run checkcites on the document.aux file:

[illegible]

Copyright (c) 2012, 2019, Enrico Gregorio, Paulo Roberto Massa Cereda

Fantastic, I found 5 references in 1 bibliography file. Please wait a moment while the reports are generated.

=> foo:2012e

=> foo:2012f

checkcites allows a couple of command line flags that will tell it how to behave. For example, check this command line:

The `--unused` flag will make the script only look for unused references in the `.bib` file. The argument order does not matter, you can also run:

The script will behave the same. Similarly, you can use:

6

The `--undefined` flag will make the script only look for undefined references in the `.tex` file. If you want `checkcites` to look for both unused and undefined references, run:

```
$ checkcites --all document.aux
```

If no special argument is provided, the `--all` flag is set as default.

Observe that our example relied on the default backend, which uses Bib \TeX . Let us change our document a bit to make it Bib \LaTeX -compliant:

Main document

```
\documentclass{article}

\usepackage{biblatex}
\addbibresource{example.bib}

\begin{document}

Hello world \cite{foo:2012a,foo:2012c},
how are you \cite{foo:2012f},
and goodbye \cite{foo:2012d,foo:2012a}.

\printbibliography

\end{document}
```

As usual, let's compile our sample document `document.tex`:

```
$ pdflatex document.tex
```

After running `pdflatex` on our `.tex` file, there is now a `document.aux` file in our work directory, as expected. However, since we are using Bib \LaTeX as well, there is another file of interest in our working directory, one that has a `.bcf` extension! In order to run `checkcites` on that specific file, we need to provide the `biber` backend:

```
$ checkcites --backend biber document.bcf
```

We can even omit the file extension, the script will automatically assign one based on the current backend:

```
$ checkcites --backend biber document
```

Now, let us run `checkcites` on the `.bcf` file, providing the `biber` backend:

