Strategy for master BibTeX file using Git submodules

Andrius Velykis

This is my approach to managing references in a single *master* BibTeX file. By putting the bibliography file in a Git repository and then using it as a *submodule* of Git repositories for my papers, I get a portable, versioned and centralised solution to reference management. Here is how I do it.

Using master bibliography file

As I mentioned earlier, I use BibTeX to manage my research references. My whole bibliography is stored in one master BibTeX file. I keep it up to date with new papers, correcting mistakes in references, etc.

Now, when writing a paper, I have a couple requirements about the references, which is not quite straightforward to achieve using a central BibTeX file:

- **Paper 'project' should be portable, with local references to the `.bib` file.**

  I avoid absolute references to the master BibTeX file. Furthermore, system-wide configuration option (e.g. via *texmf*) is also out of the question.

- **BibTeX references should be versioned.**

  When I come back to the paper (e.g. for minor edits) I want to have the same references as when I originally wrote it. My master BibTeX file may have been updated since then—even its reference keys may have changed! I would like to have access to that specific version of my bibliography.

My solution that works for these requirements is to use separate Git repositories for the master BibTeX file and for each of the papers.

Versioning the bibliography & papers

Having a version control system (VCS) for your papers is good in general. It gives a historical view on the writing process, allows reverting to earlier versions if needed and tagging important milestones, e.g. 'submitted', 'camera-ready', etc. When it comes to writing in collaboration with other authors, a VCS is absolutely indispensable!

For similar reasons, I also put my bibliography file under version control. My VCS of choice is Git and its quick repositories lend themselves nicely to have separate per-paper and bibliography repositories. There is no problem in creating single-file repositories.

I can have local repositories on my computer, or better, put them in a central location online. For example, Bitbucket offers unlimited free private Git repositories.

For this example, consider that we have two repositories:

- *myrefs.git* (e.g. accessible online at `git://example.org/user/myrefs.git`) bibliography repository with a single BibTeX `.bib` file:

  `myrefs.bib`

- *mypaper.git* my paper repository with a single LaTeX `.tex` file:

  `paper.tex`

Linking bibliography with the paper

To use the bibliography in my paper, I use Git *submodule* functionality to "nest" a specific version of *myrefs.git* repository under *mypaper.git* repository:

```
git submodule add git://example.org/user/myrefs.git refs
```

This command attaches the current version of the bibliography repository under `refs` directory. Now *mypaper.git* files look like this:

```
.gitmodules
paper.tex
refs/myrefs.bib
```

In my LaTeX file I can use the BibTeX file as a local reference: `\bibliography{refs/myrefs}`. I can move the directory to another place, I can check it out in another computer—the submodule directory will be local to the paper.

Updating the bibliography

The good thing about the *submodule* command is that it points to a specific version (*changeset*) of the submodule repository. Therefore the submodules are not affected when I update the master BibTeX file. When I checkout the paper repository, it will always use the same old version of the bibliography repository.

I can change this version if needed. If I commit changes the master BibTeX repository, I can update the submodule accordingly. Furthermore, if I need to do some destructive changes to the BibTeX file for the specific paper, I just branch the repository and use the branched version as the submodule. This is useful if, for example, I need to remove some conflicting BibTeX fields for certain bibliography style. I will not get into technical details here—documentation on Git *submodules* is available online.

I can also edit the BibTeX file directly in the submodule (`refs/myrefs.bib`) and push to the master repository. Note that it may require the submodule to point to a branch.

Benefits

With such configuration, I have just one versioned bibliography file in a Git repository. By using Git *submodules*, I can have local references to my bibliography in the papers, and can point to specific versions of that bibliography.

This approach would be valid when collaborating with multiple authors as well—and multiple bibliographies. In that case, just have different submodules for each author's BibTeX repository, e.g. `/refs`, `/refs2`, etc.

Finally, for cases when I need to send a "cleaned up" version of my paper, I can extract a subset of the full bibliography using various tools available. Bibliography management software usually allows that, e.g. using the bibtools scripts or `Database > Select Publications From .aux File` menu item in BibDesk. Then I copy the LaTeX files and replace the `refs/myrefs.bib` file with the `.bib` file of references subset and send the "cleaned up" version.