SyncTeX

Hans Hagen

1. Introduction

Some users like the SyncTeX feature that is built in the TeX engines. Personally I never use it because it doesn't work well with the kind of documents I maintain. If you have one document source, and don't shuffle around (reuse) text too much it probably works out okay but that is not our practice. Here I will describe how you can enable a more ConTeXt specific SyncTeX support so that aware PDF viewers can bring you back to the source.

2. What we want

The SyncTeX method roughly works as follows. Internally TeX constricts linked lists of glyphs, kerns, glue, boxes, rules etc. These elements are called nodes. Some nodes carry information about the file and line where they were created. In the backend this information gets somehow translated in a (sort of) verbose tree that describes the makeup in terms of boxes, glue and kerns. From that information the SyncTeX parser library, hooked into a PDF viewer, can go back from a position on the screen to a line in a file. One would expect this to be a relative simple rectangle based model, but as far as I can see it's way more complex than that. There are some comments that ConTeXt is not supported well because it has a layered page model, which indicates that there are some assumptions about how macro packages are supposed to work. Also the used heuristics not only involve some specific spot (location) but also involve the corners and edges. It is therefore not so much a (simple) generic system but a mechanism geared for a macro package like LaTeX.
Because we have a couple of users who need to edit complex sets of documents, coded in TeX or XML, I decided to come up with a variant that doesn't use the SyncTeX machinery but manipulates the few SyncTeX fields directly[1] and eventually outputs a straightforward file for the editor. Of course we need to follow some rules so that the editor can deal with it. It took a bit of trial and error to get the right information in the support file needed by the viewer but we got there.
The prerequisites of a decent ConTeXt "click on preview and goto editor" are the following:

- It only makes sense to click on text in the text flow. Headers and footers are often generated from structure, and special typographic elements can originate in macros hooked into commands instead of in the source.
- Users should not be able to reach environments (styles) and other files loaded from the (normally read-only) TeX tree, like modules. We don't want accidental changes in such files.
- We not only have TeX files but also XML files and these can normally flush in rather arbitrary ways. Although the concept of lines is sort of lost in such a file, there is still a relation between lines and the snippets that make out the content of an XML node.
- In the case of XML files the overhead related to preserving line numbers should be minimal and have no impact on loading and memory when these features are not used.
- The overhead in terms of an auxiliary file size and complexity as well as producing that file should be minimal. It should be easy to turn on and off these features. (I'd never turn them on by default.)

[1] This is something that in my opinion should have been possible right from the start but it's too late now to change the system and it would not be used beyond ConTeXt anyway.

It is unavoidable that we get more run time but I assume that for the average user that is no big deal. It pays off when you have a workflow when a book (or even a chapter in a book) is generated from hundreds of small XML files. There is no overhead when SYNCTEX is not used.
In CONTEXT we don't use the built-in SYNCTEX features, that is: we let filename and line numbers be set but often these are overloaded explicitly. The output file is not compressed and constructed by CONTEXT. There is no benefit in compression and the files are probably smaller than default SYNCTEX anyway.

3. Commands

Although you can enable this mechanism with directives it makes sense to do it using the following command.

```
\setupsynctex[state=start]
```

The advantage of using an explicit command instead of some command line option is that in an editor it's easier to disable this trickery. Commenting that line will speed up processing when needed. This command can also be given in an environment (style). On the command line you can say

```
context --synctex somefile.tex
```

A third method is to put this at the top of your file:

```
% synctex=yes
```

Often an XML file is very structured and although probably the main body of text is flushed as a stream, specific elements can be flushed out of order. In educational documents flushing for instance answers to exercises can happen out of order. In that case we still need to make sure that we go to the right spot in the file. It will never be 100% perfect but it's better than nothing. The above command will also enable XML support.
If you don't want a file to be accessed, you can block it:

```
\blocksynctexfile[foo.tex]
```

Of course you need to configure the viewer to respond to the request for editing. In Sumatra combined with SciTE the magic command is:

```
c:\data\system\scite\wscite\scite.exe
  "%f" "-goto:%l"
```

Such a command is independent of the macro package so you can just consult the manual or help info that comes with a viewer, given that it supports this linking back to the source at all.

4. Methods

Contrary to the native SYNCTEX we only deal with text which gives reasonable efficient output. If you enable tracing (see next section) you can see what has become clickable. Instead of words you can also work with ranges, which not only gives less runtime but also much smaller `.synctex` files. Just try:

```
\setupsynctex[state=start,method=min]
```

to get words clickable and

```
\setupsynctex[state=start,method=max]
```

to get the more efficient ranges. The overhead for `min` is some 10 percent while `max` slows down around 5 percent.

5. Tracing

In case you want to see what gets synced you can enable a tracker:

```
\enabletrackers
  [system.synctex.visualize]
\enabletrackers
  [system.synctex.visualize=real]
```

The following tracker outputs some status information about XML flushing. Such trackers only make sense for developers.

```
\enabletrackers[system.synctex.xml]
```

6. Warning

Don't turn on this feature when you don't need it. This is one of those mechanism that hits performance badly.
Depending on needs the functionality can be improved and/or extended. Of course you can always use the traditional SyncTeX method but don't expect it to behave as described here.