A pdfLaTeX-based automated journal production system

THIERRY BOUCHE
Cellule MathDoc
UMS 5638 (Université Joseph Fourier & CNRS)
100, rue des Maths
Domaine Universitaire
38402 St-Martin-d'Hères, France
thierry dot bouche (at) ujf dash grenoble dot fr
http://www.cedram.org/

Abstract

We present the recent development of a production system for mathematical serials with both an electronic and paper version. The challenges were many: (i) no house style layout should be imposed, as the journals come from different publishing houses and may have very different typographical options; (ii) produce screen-optimised and printer-friendly output at once; (iii) avoid any duplication of information so that all aspects of the publications are always in sync (Web site metadata, table of contents…), thus (iv) generate on the fly article's page numbers, XML metadata at the published volume level from one master LaTeX source file tree. Using available technology (pdflatex, pdfpages.sty and `\write18`*), the proposed solution to these problems appeared amazingly simple and easy to use. However, we'll show that there is quite some room left for improvement.*

Introduction

At the end of fall 2003, discussions began in the French mathematical community about a consolidated effort for high-quality online publishing of our academically (meaning: independent and learned society) published research journals. One driving force of this project was the achievements of the NUMDAM digitisation program, which has more or less settled standards for delivery and navigation of a significant part of the mathematical literature (`http://www.numdam.org`).

Among the prominent features of NUMDAM, we have the rich set of metadata for each article, including tagged bibliographies, and the powerful search engine associated with it, written by Claude Goutorbe of Cellule MathDoc. Thanks to various matching tools provided by the AMS or developed internally, whatever sensible link can be provided is added to the Web interface, which is something our users very much appreciate.

It appeared after some investigations that what was almost straightforward to achieve in a retrodigitisation process could become a nightmare to produce in a natively digital environment:

1. Although all journals under consideration were produced with some flavour of TeX, each had a specific format with a primarily paper-only approach to the publication process.
2. Although all of them had a Web site, none of them had reliable processes to control whether the metadata exposed on this site was consistent with the reality of the paper issues.
3. Although bibliographies are such a routine object in the learned publication business, we could count over 20 ways of "structuring" them in the TeX files.

It turned out that, although we're now in the 21st century, the rather quaint copy-paste operation (a typical late 20th century hobby) was the main procedure on which all these journals relied for the most typical aspect of serials publishing: exposing the same data in many formats and contexts. Let's think for a moment about the starting page number of an article, which is a rather critical datum if you hope to find it somewhere. It will be printed within the article itself (where it is determined only at the last step of production, as it depends on the lengths of all the same volume's articles before it), probably an inner table of contents, possibly a back cover table of contents, presumably a Web table of contents, not to mention

an eventual annual index, or third party uses of the data, as current contents or indexing databases services, that could nowadays be fed through OAI-PMH or RSS feeds...

For instance, let me give the following (anonymous) example: a respected journal once published a paper which, for some obscure (possibly scientific!) reason, was ultimately shortened by a paragraph or two in the proof reviewing stage. It was the first paper of the first issue of its year of publication in that journal, and was paginated 1–27 (hard coded in the TEX source) although its final form had 26 pages. The next article was thus paginated starting at p. 29 but the printer saved the 2 white pages. Thus, all the 2000 printed page numbers in this volume are wrong except the first 26.

Last minute changes and copy-paste are the two devils in journal production. A more obvious example: an accepted paper happens to have a serious scientific failure which is found after all the publishing process has been done. The author informs the journal in a hurry that they have to cancel it, of course they do. Now, all pages numbers are wrong for the following articles, go figure where they have already been disseminated!

In a retrodigitisation process, these issues are just annoying, but all you aim at is to create accurate and structured metadata describing an existing paper collection. Moreover, as it is a batch process on a large amount of similar data, high quality can be achieved at a reasonable cost. Production cost and complexity is an issue for our small journals, which heavily rely on voluntary effort by researchers in their spare time (as well as *Cahiers GUTenberg* which will enter the scene later on).

Good solutions to complex problems are simple

So. How do you produce a journal in such a way that you have detailed, accurate metadata in a versatile format, a powerful Web site with screen optimised versions of the articles, and yet the same paper version?

After some time spent in reviewing the existing more or less full answers to this question, mostly based on scripts, heuristics, external programs or auxiliary files, I happened to find one so simple that I think it deserves to be detailed here. In fact, it is so simple that I feel a little embarrassed to show the main steps in the small `\includearticle` macro on p. 48 while I spend the rest of the text discussing the troubles. At the time of this writing, 11 issues from 3 journals (including the latest *Cahiers GUTenberg*) were made using this tool.

This solution has been made possible rather recently thanks to the collaborative effort of many talented developers, and although I could achieve this because of the power of TEX macro language, I must confess that I never use TEX itself, but engines that understand an extension of TEX's primitives, yet have a full macro interpreter onboard, namely: pdfTEX with `\write18` enabled and (soon) Tralics.

Here is a short description of the system.

Principles

1. Any metadata is input at most once in the system, preferably in the relevant file.
2. Anything that is not deterministically determined by a given file, should stay away from that file.
3. Anything that can be computed, *should* be computed!
4. Do not reinvent the wheel, do not invent exotic formats that no one will master, stay pragmatic but avoid bottle-necks that would impact versatility of future use or quality of the output.

Implications

1. A journal is a set of volumes, made of issues, made of articles, plus various other material, mostly constant. The journal description belongs to the journal file, the volume description to the volume file, etc. Notice that a page number is essentially a by-product of a completed issue; except for the first page of an issue, it should be set nowhere.
2. As it is the de facto standard of math writing, AMSLATEX was chosen as the input format, with the minmal set of extensions as required by the subsequent processing. BibTEX was chosen for the bibliographies.

User interface

Of the relevant parts of a journal, I didn't implement the volume level (this would save copying *one* number) but tried to define an *issue*.

Claim 1 *An issue is entirely determined by*

- *its bibliographic data (journal, year, month, volume, issue),*
- *its first page number,*
- *the ordered list of the articles,*
- *and optional additional material such as advertising, disclaimers, obituaries...*

I write this (and only this) in the issue file:

```
\documentclass[francais,CG,Volume,
                        Couverture]{cedram}
\IssueInfo{}{46-47}{avril}{2006}
\SetFirstPage{1}
\SpecialNo{Les fontes (Brest 2003)}
\begin{document}
\makefront
   \includearticle{edito}
   \includearticle{atanasiu}
  [...]
   \includearticle{devroye}
   \includepub{pubyannis}
\makeback
\end{document}
```

The `\makefront` command makes the front matter of the paper volume (including the table of contents), `\includearticle` includes the corresponding article, `\makeback` makes the back matter, etc.

Each article obeys an AMSLaTeX structure:

```
\documentclass[CG,francais]{cedram}
\usepackage{x,y}
\title[Formatons les formats de fonte]
       {Formatons\\ les formats de fonte !}
\author{\firstname{Luc} \lastname{Devroye}}
\address{McGill University,\\
etc.}
\thanks{L'auteur...}
```

Assuming that all the articles and other material are in final form (which means that they are in a directory of their own, and that an error-free source master file compiled with pdflatex has been compiled successfully with all cross-references resolved), when we compile (twice) the issue file, it will produce one big PDF comprising all inner pages of the paper volume; this is sent to the printer. It will also produce the pages of the cover, and an XML file with all the metadata for this volume. In fact, as a side effect, you'll also find in each article subdirectory a hyperlinked PDF with a first page added, so that everything is ready at once for shipping both the paper and electronic editions of that issue.

Architecture

LaTeX is a "document preparation system"; it operates at the document level. I'm not convinced by systems that address the abovementioned issues by considering articles as subdocuments of a master document: they require a high level of normalisation of the sources to avoid conflicts (different macros with same names, cross references, etc.), many redefinitions of standard user commands which is very risky since users like shortcuts, and would yield broken links when you provide an article on its own. We can't expect that mathematicians will obey such strict rules, nor TeXnicians! Thus the relevant document unit in a journal is an article, preferably isolated in a specific directory in order to avoid conflicts with input of figure names, etc. It should be compiled individually and produce a nicely hyperlinked and searchable vector PDF. The metadata relevant to the article are standard: authors' data, title, abstract, keywords, subject classification, dates, bibliography. The volume, issue, page numbers are not part of the article itself, as it can be moved at any time without affecting its scientific content, thus without edits. Of course, an article is prepared for a journal, so that info should be present in the article file, and determine the layout and many typographical options.

Starting from the obvious observation that nobody but TeX knows how long an article is, when considering its source, I eventually understood that the only reliable solution for setting error-free page numbers was to ask TeX to do so. Of course, you could compile a volume with a perl script that would compute things, compile articles, examine the produced PDF to deduce page numbers, modify the articles, recompile, etc. These are heuristics, and will be broken at the first discrepancy between the paper volume and the model volume assumed by the script. In some sense, as long as articles have a "bibliographical" reference, we're still in the retrodigitisation paradigm when producing an electronic edition: it is the paper model that endows the article with its metadata, so it is by assembling the paper volume that we can deduce the required data to get the final articles. But, more generally, the same applies to purely electronic serial publications: even the table of contents of an incrementally growing online volume is something that is generated as the last step when the latest article is added. Only flat repositories like arXiv can bypass this question.

Implementation

Articles

As far as articles are concerned, the `cedram` class is simply `amsart`, with a few features added. Namely:

- some extra metadata fields (provision for bilinguism, journal dates, ...);
- the automatic inclusion of a configuration file at `\begin{document}`;
- a 'lastpage' trick;
- the facility to store the literal TEX code of a macro argument or an environment's contents and write it to auxiliary files in various formats (by overloading standard macros);
- hooks added in the presentation code so that all known layout options can be easily implemented;
- some ad hoc definitions for various theorem styles;
- a journal option to load the journal file defining all constant metadata and make-up for that journal;
- some more class options, mostly for compatibility (by default, the class requires hyperref, pdflatex, T1 encoding, Latin Modern fonts, ...).

There is a light version called 'special' for things that should look like an article but do not have its full features (editorial statements, e.g.).

When compiling an article at its final stage, it reads a possible configuration file that might override options and provide the needed metadata (issue info, first page), writes out the screen-optimised PDF (with a first page added, kind of an offprint cover, meant to identify more clearly the article origin when it will travel the net on its own), a `.cdrsom` file which contains a TEX command providing all the data pertaining to this article that could be used to generate the corresponding TOC line in whatever format, and a `.cdrxml` file that contains an XML-like snippet with all the metadata for this article.

Volumes

A volume is made using the same `cedram` class, with an option 'Volume', so that all the typographical options are the ones of the journal. There are some specific options to this mode of operation, such as 'Couverture' which will generate the cover, 'CouvTires' the covers for author's (paper) offprints ...

Let me explain what it does line by line, which will show how it works, and why it is so simple and reliable.

```
\documentclass[francais,CG,Volume,Couverture]{cedram}
```

This sets the volume mode for *Cahiers GUTenberg* (CG), with French hyphenation patterns for the editorial material surrounding articles, and will generate a cover.

```
\IssueInfo{}{46-47}{avril}{2006}
\SetFirstPage{1}
\SpecialNo{Les fontes (Brest 2003)}
```

These set variables: issue number (CG has no volumes), month and year of publication, starting page number of the first article, title of the issue when relevant. These variables are available during the whole LATEX run, as well as written to auxiliary files.

```
\begin{document}
\makefront
```

In article mode, many things happen at the point of `\begin{document}` — but not in volume mode, as far as I can tell! The `\makefront` call could have been automated here. In any case, this command sets `\pagestyle{empty}`, and inputs `CG-front.tex`, which in turn inputs the definition files for the front matter (title page, administrative data, summary). It also inputs a void file that can be populated locally for special occasion issues. The summary is a container constant source file making use of the issue variables and inputting a summary data file with some fixed name which is generated later on (thus the necessity of two runs to complete an issue). In fact, the summary is a 'special' item, thus a complete LATEX file which is compiled during the run, in a subprocess similar to the 'article' case below. The `\makefront` macro ejects all remaining material to be printed, goes to the next odd page, and sets the page counter of the master document to the value given by `\SetFirstPage`.

```
\includearticle{devroye}
```

This is the main operation, but maybe the simplest one. It is so simple that I include its entire definition here:

```
\def\includearticle#1{%
  \IncludeArticle[2]{#1/}{#1}%
  \ifx\@empty\articlesXML
    \gdef\articlesXML{#1/#1.cdrxml}%
  \else
    \g@addto@macro\articlesXML{ #1/#1.cdrxml}%
  \fi
  \ifx\@empty\articlesSOM
    \gdef\articlesSOM{#1/#1.cdrsom}%
  \else
    \g@addto@macro\articlesSOM{ #1/#1.cdrsom}%
  \fi
```

As we can see, \includearticle is just a shorthand for the more general \IncludeArticle that assumes that the article's master TeX file resides in a subdirectory with the same basename. Moreover, it stores in a macro the list of .cdrxml and .cdrsom files that will be dealt with at the end of the run. Going back a few lines in cedram.cls, we have:

```
\def\IssueInfo#1#2#3#4{%
 \tkkv={\ScreenMode\issueinfo{#1}{#2}{#3}{#4}}%
 \issueinfo{#1}{#2}{#3}{#4}}
\let\articlesXML\@empty
\tkkp={\setpage}
\def\pdflatex{%
    pdflatex --shell -interaction=nonstopmode }
\newcommand\IncludeArticle[3][2]{%
 \cleararticlepage
 \immediate\write18{echo '\the\tkkv
             \the\tkkp{\thepage}' > #2#3.cfg}%
 \immediate\write18{cd #2 && \pdflatex  #3}%
 \ifcdr@redoBibtex
 \immediate\write18{cd #2 && bibtex #3}%
 \immediate\write18{cd #2 && \pdflatex  #3}%
 \immediate\write18{cd #2 && \pdflatex  #3}%
 \fi
 \immediate\write18{cd #2 && \pdflatex  #3}%
 \includepdf[pages={#1-},noautoscale]{#2#3.pdf}%
}
```

The main article operation is thus the following.

1. The last page is ejected and, depending on the journal style, we go to the next odd page before dealing with the article.
2. The issue info has been stored globally and is written to the auxiliary file for the article, together with the current page number (a traditional \write could have been used here as well).
3. Then, the article is recompiled; this will use the given information because it reads the just created .cfg file at \begin{document}. Optionally bibtex and further pdflatex calls are executed.
4. Finally, the newly generated PDF is included (except, of course, for its first page) in the master volume being produced.

The trick here is that we can trust the page counter of the master document: this is the actual paper volume to be printed! Thus we can reasonably be sure that the value of \thepage is the correct value for the first page of the next article, which will be included precisely at this page. And this will remain true next time as we input the final PDF of the article right away.

```
\includepub{pubyannis}
\makeback
\end{document}
```

These last lines show that we can add advertising, or anything else. The counterpart of \makefront is \makeback: it includes almost static pages (instructions to authors, subscription info, etc.). In fact, many things happen at \end{document}, which is the only place where everything is known about the issue in final form: an XML file is written by processing the master's and all articles' XML snippets, summary data is generated by concatenation of all articles' summary lines, and the cover is built by compiling the adequate template.

Metadata and format questions

As long as printer and screen (Web) PDF files are considered, the described system has proved to work quite effectively. But, when you wish to produce versatile metadata from LaTeX source, you can expect troubles. All typeset material is done by LaTeX, thus the abovementioned .cdrsom files are perfect, thanks to the possibility of redefining any necessary macro on the fly to have different views on the same data (for instance, one journal has three summaries in it: one in French, with corresponding abstracts, another in English, both at the beginning of the paper volume, and another one set differently on the back cover, where actual titles are used: this is why I store nine fields in the .cdrsom files).

Apart from pragmatic reasons discussed earlier, there is a fundamental reason to prefer TeX source as the master for all metadata: math authors write their papers with TeX, and validate their scientific content on the printed result, which is where last minute corrections happen. With a full XML process, outputting TeX code after automated transformations, the correction process would be much more difficult to control, and could yield cases where there is simply no way to obtain the desired physical representation of the article, which is at the present time still the only long-term reference for the scientific content of the paper.

The first version of the cedram tools assumed a lot of postprocessing of the 'pseudo' XML files output by LaTeX. We had the TeX code somewhat sanitised,

textual material converted to UTF-8 with variable success, and math expressions exposed as GIFs on our Web interface, thanks to `latex2html`.

My first idea in this respect was to use the kind of trick that is exploited in hyperref to produce properly encoded PDF bookmarks in order to write Unicode files. I was not able to achieve this myself. I also had a look at TEX4Ht which sounds like a good conversion tool from TEX to XML or HTML+GIF as an alternate presentation format. I gave up because of the huge number of parameters and files necessary to understand before producing output only somewhat similar to my expectations.

I am currently experimenting with Tralics [1], which might be the killer application: instead of asking pdflatex to write out a pseudo XML snippet for each article, that will need further processing, it can easily write structured code tweaked for Tralics, where all the data is the literal unexpanded TEX string, leaving all the conversion process from TEX data to Tralics, which is very good at doing so.

It even parses BibTEX files, but might also easily be used to structure `thebibliography` environments! For example, here is an excerpt from the file generated by the compilation of our example article.

```
\begin{xmlelement}{article}
\xmlbibcite {b8}{8}
\xbox{pagedeb}{149}
\xbox{pagefin}{166}
 \begin{xmlelement}{auteur}
   \xbox{nomcomplet}{\firstname {Luc}
     \lastname {Devroye}}
   \xbox{prenom}{Luc}
   \xbox{nom}{Devroye}
{\killparcode\begin{xmlelement}{adresse}{McGill
      University,\\  etc.\end{xmlelement}}
   \end{xmlelement}
{\killparcode\begin{xmlattelement}[fr]{titre}%
 Formatons\\ les formats de
 fonte !\end{xmlelement}}
   \begin{biblio}
    \bibitem{b8}J.~\textsc{Andr\'e}
     \pointir « Ligatures \& informatique »,
     \emph{Cahiers GUTenberg}, \no22,
     p.~61--86, 1995. \end{biblio}
 \end{xmlelement}
```

After some minor configuration, thanks to the fact that Tralics rather deeply understands TEX macros, Tralics will generate a wonderful, valid, XML, with all text converted to Unicode, and math to MathML. This XML can be exploited at once on our Web sites.

The only remaining question is whether the world is ready for MathML. Recent tests show that the quality of the display of MathML expressions in current browsers has drastically increased: it is now similar to TEX in readability (which relies a lot on fine positioning of sub- or superscripts), and it considerably enhances accessibility to the math content on the Net. The only remaining difficulty is that, as Tralics is a full TEX interpreter, it cannot generate easily a mixed format sanitising the text strings to well-formed Unicode XML but keeps a verbatim copy of the math formulas in TEX, which might be the most practical for many of our users in the near future ...

```
<?xml version='1.0' encoding='iso-8859-1'?>
<article>
 <pagedeb>149</pagedeb>
 <pagefin>166</pagefin>
 <auteur>
  <nomcomplet>Luc Devroye</nomcomplet>
  <prenom>Luc</prenom>
  <nom>Devroye</nom>
  <adresse>McGill University,\\ etc.</adresse>
 </auteur>
 <titre xml:lang="fr">Formatons les formats
                          de fonte !</titre>
 <biblio type='flat'>
  <bib_entry crossref='cite:b8'>
   <reference>8</reference>
   <bibitemdata>J. <hi rend='sc'>André</hi>
     « Ligatures &amp; informatique »,
     <hi rend='it'>Cahiers GUTenberg</hi>,
     no 22, p.~61&#x2013;86,
     1995.</bibitemdata>
  </bib_entry>
 </biblio>
</article>
```

References

[1] José Grimm, "Tralics, a LATEX to XML Translator", *TUGboat* 24:3 (2003), Proceedings of EuroTEX 2003, pp. 377–388.