Multipub: Managing a LaTeX Manuscript for Concurrent Print and Electronic Formats

Paolo Prandoni

January 18, 2016

Abstract

The following document describes a set of tools and the workflow to generate multiple publication formats from a single LaTeX manuscript; targeted formats are a printer-friendly PDF, a PDF version for small-screen eReaders such as the Kindle, an HTML version and an EPUB3 version. Multi-target compilation is achieved using a variety of applications, including LaTeXML, a style file and some custom scripts[1]; installation of the toolchain is described in detail for a Linux machine. With a few tweaks to your original LaTeXdocument's structure and inclusions, no changes to the source manuscript are required to switch between targets.

1 Introduction

So, you just put the finishing touches to your LaTeX manuscript, which compiles to a beautiful camera-ready PDF file: your book is finally complete... or is it? Aren't you forgetting about the *electronic version* of the book?

While there are plenty of applications that can convert pure-text manuscripts into one of the many ebook formats, there are few complete toolchains that properly handle works containing mathematical formulas and drawings. Most attempts to include math into HTML pages rely on bitmap graphics to encapsulate formulas as images. However both HTML and ebooks are reflowable, i.e. there is no fixed page formats and the font size can be changed by the user; therefore, pre-rendered bitmap images either scale rather badly or do no fit at all. Similar problems affect illustrations, especially figures generated programmatically by using PostScript primitives (or PsTricks).

In the rest of this memo I will describe the toolchain and the workflow I use to manage the concurrent PDF and eBook rendering of LaTeX manuscripts. Starting with a properly prepared master document in LaTeX, a simple compilation script can be used to generate a printable PDF, a PDF for the Kindle, an HTML

version and an EPUB version. Changing the compilation target does not require any editing of the master text but different style files can be conditionally included according to target. For a (verbose) explanation of the decision process that went into the selection of the tools, see Appendix A.

The main link in the toolchain is LaTeXML [3], that will be used to compile LaTeX to XHTML; however, a set compilation scripts is provided so that all the gory details are conveniently hidden [1]. The full toolchain uses a variety of third-party applications besides LaTeXML, so it is not easily portable[1]. I will assume a Linux environment from now on, and details on how to set up your environment are described in Section 4. For other operating systems, I recommend installing a virtual machine such as VirtualBox [6], install a quick and easy Linux distribution and proceed from there.

2 Preparing the Manuscript

The tools described in this document can generate four compilation targets:

- PRINT: standard compilation leading to a printable PDF output.
- KINDLE: specially-formatted PDF to fit the screen of small e-paper readers like the Kindle.
- HTML: compilation with LaTeXML for electronic publishing (browsers with MathML or MathJax support).
- EPUB: EPUB3 version for e-readers.

After having installed the required files as described in Section 4, multitarget compilation is achieved via the following steps:

1. include the `multipub` package in the master LaTeX document
2. conditionally include external packages using `multipub`'s target-specific macros
3. (optionally) set up the file structure for multi-file documents
4. handle the PsTricks illustrations
5. compile via the provided `mpbuild` shell script.

[1] For instance, LaTeXML is written in Perl and installing Perl under Windows is far from trivial.

2.1 The `multipub` Package

To enable multi-target compilation, include[2] the `multipub` package right after the `documentclass` declaration in your master document and specify your compilation target immediately below; then, just before `\begin{document}`, invoke the `\mutipubbegin` macro like so:

```
\documentclass[...]{...}
\input{multipub}
\multipub{PRINT}
...
...
\multipubbegin
\begin{document}
...
```

All target names can be used to define conditional portions of the LaTeX manuscript that will be active only when the target is selected; for instance

```
\begin{PRINT}
  % custom page formats for the PRINT version
  \include{styles/printlayout}
  \usepackage[utopia]{mathdesign}
  \usepackage{pstricks}
\end{PRINT}

\begin{EPUB}
  % custom page formats for the EPUB version
  \include{styles/epublayout}
\end{EPUB}
```

2.2 External Packages

Since we'll be using LaTeXML, the first step in preparing a manuscript for electronic publishing is to **make sure that all critical packages have available LaTeXML bindings** (see Figure 1). A critical package is a functional package that will be needed for all compilation targets. Please note that any package affecting the page layout only (e.g. specific font packages) will be irrelevant when compiling for HTML or EPUB and so it can be conditionally included for PRINT or KINDLE as explained above. Another exception is the inclusion of PSTricks-derived packages, for which a different strategy is provided below.

[2] I couldn't package `multipub` as a style file because LaTeXML only accepts style files for which it has bindings.

a0size, a4, a4wide, aasms, aaspp, aastex, acronym, ae,
afterpage, alltt, amsbsy, amscd, amsfonts, amsmath,
amsopn, amsppt, amsrefs, amssymb, amstex, amstext,
amsthm, amsxtra, apjfonts, array, avant, babel,
bbm, bbold, beton, bm, bookman, booktabs, braket,
calc, cancel, caption, ccfonts, chancery, charter,
circuitikz, cite, citesort, cmbright, color, colordvi,
colortbl, comment, concmath, courier, crop, cropmark,
dcolumn, deluxetable, doublespace, dsfont, ellipsis,
elsart, emulateapj, emulateapj5, enumerate, epigraph,
epsf, epsfig, epstopdf, eucal, eufrak, euler, eulervm,
eurosym, euscript, exscale, fancyhdr, fix-cm, fixltx2e,
flafter, fleqn, float, floatfig, floatflt, floatpag,
fontenc, fontspec, footmisc, fourier, framed, fullpage,
geometry, german, graphics, graphicx, grffile, helvet,
here, hhline, html, hyperref, hyperxmp, ifpdf, ifthen,
ifvtex, ifxetex, import, indentfirst, inputenc,
iopams, keyval, latexml, latexsym, lineno, listings,
listingsutf8, longtable, lscape, luximono, lxRDFa,
makeidx, mathpazo, mathpple, mathptm, mathptmx,
mathrsfs, multicol, multido, multirow, nameref,
natbib, newcent, ngerman, nicefrac, ntheorem, numprint,
palatino, paralist, parskip, pdflscape, pdfsync, pgf,
pgfplots, pifont, placeins, preview, psfig, pslatex,
pspicture, pst-grad, pst-node, pstricks, pxfonts,
relsize, revsymb, revtex, revtex4, rotate, rotating,
rsfs, scalefnt, setspace, slashed, srcltx, subfig,
subfigure, subfloat, supertabular, svg, tabularx,
textcomp, texvc, theorem, threeparttable, tikz-3dplot,
tikz, times, tocbibind, transparent, txfonts, type1cm,
ulem, units, upgreek, upref, url, utopia, verbatim,
wrapfig, xargs, xcolor, xspace, xunicode, yfonts

Figure 1: Packages with available bindings in LateXML.

2.3 File Structure for Multi-File Documents

With multi-part documents (such as the chapters in a book or different sections in a longer text) it is often convenient to have a master document and include the relevant files from subfolders; e.g. if you have a `book.tex` master file in the top-level directory and a series of subfolders containing chapters you may find yourself writing something like:

```
\documentclass[...]{book}
...

\begin{document}

\title{My Book}
\maketitle

\include{ch01/chapter1.tex}
\include{ch02/chapter2.tex}

\end{document}
```

The rationale behind this file structure is to organize the material for each chapter in separate subfolders, possibly creating another level of folders in each subfolder to collect figures or data files. The problem, however, is that if you now, say, include a figure in a chapter file using a path relative to the chapter's subfolder, the figure will not be found since LaTeX assumes that all paths are relative to the master document. To work around this problem without using absolute paths, `multipub` defines two specific macros. In your master document, use the `\includefile` macro to include chapters, separating the path from the file name:

```
\documentclass[...]{book}
...
\includefile{ch01}{chapter1.tex}
\includefile{ch02}{chapter2.tex}
```

then, in each *included* file, you can use the `\localpath` macro to include files relative to the included file itself. For example, assume the folder `ch01/figs/` exists; then in `chapter1.tex` you can use things like:

```
\includegraphics{\localpath{figs/fig.eps}}
```

and the document will compile. **Using the `\includefile` macro is critical with respect to PSTricks figures**, as explained in the following section.

As a final note, *do NOT use underscores in the filenames or path names* as this seems to confuse LaTeXML.

2.4 Figures and PSTricks

LaTeXML supports picture inclusion via `\includegraphics`; in other words, figures defined as in the code snippet below should present no problem across compilation targets:

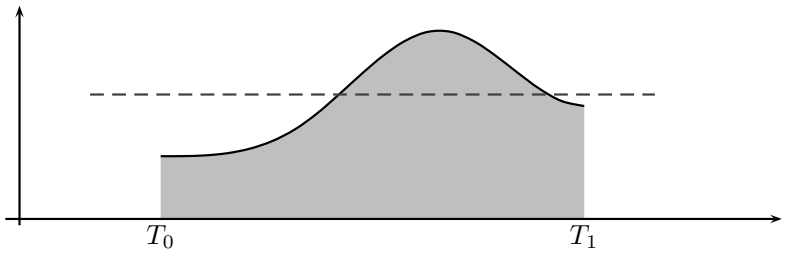
```
\begin{figure}
  \includegraphics{figs/figure.eps}
\end{figure}
```

On the other hand, although LaTeXML has some basic PSTricks bindings, it certainly does not cover the multitude of useful PSTricks-based packages in use today. Therefore the strategy for multitarget compilation is the following:

- in the main document folder (or in each chapter folder) create a subfolder called `ps`
- isolate the PSTricks-specific code for each figure in the tex file and save it in separate small file in the `ps` folder (e.g. `ps/fig3.tex`)
- use the `\pstf` macro to include the PsTricks figure using the file name without extension

When compiling for the EPUB target, **the `mpbuild` compilation script will pre-render all the figures** in all the `ps` subfolders of your manuscript.

For example, consider the following figure:



obtained with the following code:

```
\begin{figure}
  \center
  %%%% BEGIN PSTricks %%%
```

```
  \savedata{\smooth}[
    1.0000 1.0000 1.1579 1.0011 1.3158 1.0130 1.4737 1.0484
    1.6316 1.1201 1.7895 1.2408 1.9474 1.4232 2.1053 1.6777
    2.2632 1.9852 2.4211 2.3076 2.5789 2.6068 2.7368 2.8445
    2.8947 2.9824 3.0526 2.9828 3.2105 2.8418 3.3684 2.6079
    3.5263 2.3338 3.6842 2.0725 3.8421 1.8769 4.0000 1.8000]
  \psset{xunit=1.8cm,yunit=0.8cm,linewidth=0.8pt}
  \begin{pspicture}(-.5,-.5)(5.5,3.5)
    \pscustom{
      \dataplot[plotstyle=curve,linewidth=1.1pt]{\smooth}
      \gsave
      \psline(4,0) \psline(1,0)
      \fill[fillstyle=solid,fillcolor=lightgray]
      \grestore}
    \psline{->}(0,-.1)(0,3.4)
    \psline{->}(-.1,0)(5.4,0)
    \rput[t](1,-.1){$T_0$}
    \rput[t](4,-.1){$T_1$}
    \psline[linestyle=dashed]{-}(0.5,1.983)(4.5,1.983)
  \end{pspicture}
  %% END PSTricks %%%%
\end{figure}
```

Cut all the lines between the `BEGIN PSTricks` and `END PSTricks` markers and save them as the file `ps/fig1.tex`. Now you can include the figure with

```
\begin{figure}
  \pstf{fig1}
\end{figure}
```

When compiling for the PRINT target, the macro will include the pre-rendered version of the figure if it is available, otherwise it will include the PSTricks code directly (obviously in this case the PRINT version must include all necessary PSTricks packages; you can do that within the conditional inclusion section in the preamble of the document). When compiling for `EPUB` the pre-rendered figures *must* be available and a compilation error is raised otherwise.

Normally, the size of the figures is automatically adjusted to the compilation target; however, the `\pstf` macro takes an optional argument that can be used to specify the width of the included pre-rendered figures.

3 Compilation

Compilation is performed using the provided `mpbuild` script. In the following examples assume you're compiling the `mybook.tex` master file.

3.1 PRINT Version

```
> mpbuild PRINT mybook
```

Compilation in this case is a simple LaTeX→ dvips → ps2pdf sequence and it will generate a `mybook.pdf` output.

3.2 KINDLE Version

```
> mpbuild KINDLE mybook
```

The KINDLE version also produces a `mybook.pdf` file. In this case, the following typesetting parameters are modified:

- the page size is scaled to an 800x600 aspect ratio, with minimal borders
- the font is changed to a sans-serif family (epigrafica)
- the math font is scaled down

Because of the small page size in the KINDLE version, please bear the following in mind:

- "long" equations will have to be manually split across lines. Use the conditional inclusion macros to provide alternate versions of long equations.
- when including figures, always specify the size in terms of textwidth, in order to automatically rescale the graphics.

3.3 HTML Version

```
> mpbuild HTML mybook [destination]
```

HTML compilation takes place in two steps, first the LaTeX manuscript is converted to XML and then the XML is converted to HTML; both operation are performed by LateXML. Important notes:

- the script (optionally) pre-renders all the PsTricks figures in the manuscript's file tree;
- if specified, the resulting XHTML files will be placed in the `destination` folder;
- the main file will be `mybook.xhtml`
- the look and feel of the resulting web pages can be controlled by setting up a CSS file called `mybook.css` in the same folder as the main LaTeX file; see Section 3.5 for a brief discussion.

3.4 EPUB Version

```
> mpbuild EPUB mybook
```

EPUB compilation requires first to compile the manuscript as HTML, and then to package all the necessary files into an `mybook.epub` file. The conversion HTML to EPUB is performed using Calibre. As in the HTML case, you can control the look and feel of the resulting eBook by providing a custom CSS `mybook.css`.

3.5 Custom CSS

The HTML produced by LaTeXML contains CSS markers for all the meaningful parts of the text. Most of the time, the CSS class names are related to the LaTeX sectioning names, prepended by the suffix `ltx_`. So, for instance, you can change the attributes of a section title by specifying a set of CSS directives for the class `.ltx_title_section`:

```
.ltx_title_section {
  background: lightgray;
  padding: 10px 0px 10px 10px;
  border-radius: 6px;
}
```

or you can change captions like so:

```
.ltx_caption {
  width: 80%;
  margin: auto;
  font-style: italic;
}
```

Changes in the style once again address the relevant CSS tag; for instance, to eliminate the initial indentation in paragraphs use

```
.ltx_para > .ltx_p:first-child {
  text-indent: 0;
}
```

Please look at the provided examples and consult the LaTeXML documentation for more details.

4 Setting Up the Toolchain

If you start from a fresh Linux installation (e.g. Ubuntu), first install TeX and some additional components (do all these steps as superuser, obviously):

```
> apt-get install texlive
> apt-get install texlive-latex-extra
> apt-get install texlive-fonts-extra
> apt-get install texlive-lang-greek
> apt-get install texlive-generic-extra
```

In order to have up-to-date EPUB support, we need to compile LaTeXML from the Perl sources. First install the necessary dependencies:

```
> apt-get install xzdec
> apt-get install libxml2-dev
> apt-get install libxslt1-dev
> apt-get install zlib1g-dev
> apt-get install lib32z1-dev
> apt-get install libgdbm-dev
```

Install the necessary Perl libraries using `cpan` (you may need to configure `cpan` on first run, all default options are OK):

```
> cpan install XML::LibXML
> cpan install XML::LibXSLT
> cpan install Image::Size
> cpan install JSON::XS
> cpan install Archive::Zip
> cpan install File::Which
```

Now download the LaTeXML sources, unzip the archive and compile:

```
> mkdir latexml
> cd latexml
> git clone https://github.com/brucemiller/LaTeXML.git
> cd LaTeXML
> perl Makefile.PL
> make
> make test
> make install
```

Finally, for EPUB support, install Calibre:

```
> apt-get install calibre
```

Now download the `multipub` package from [1] and unpack it somewhere in your filesystem. Make sure that the `toolbox/scripts` subfolder is in your `PATH` by adding the following line to your `.bashrc`

```
PATH=$PATH:<path-to-multipub-package>/toolbox/scripts/
```

Finally, to test the HTML compilation, you can install a simple webserver such as `lighttpd`; also set your account as a member of the `www-data` group:

```
> apt-get install lighttpd
> usermod -G www-data <your-login-name>
```

Now you can place the HTML files generated by LaTeXML in a subfolder of `/var/www/` and browse to your ebook at `hhtp://<ip-address>/mybook/mybook.xhtml`:

```
> mkdir /var/www/mybook
> mpbuild HTML mybook /var/www/mybook
```

A Choosing the Links in the Toolchain

This section explains the criteria behind the toolchain selection, which resulted in

- EPUB as the chosen eBook format
- MathML for math support

$$\langle \varphi^{(n)}(t), \varphi^{(m)}(t) \rangle = \frac{\pi}{2\Omega_N^2} \int_{-\Omega_N}^{\Omega_N} e^{j\Omega(n-m)T_s} d\Omega$$

Figure 2: Example of mathematical formula. This should look good in all formats.

- LateXML as the XHTML compiler
- Calibre for EPUB3

A.1 Render Mathematical Formulas and eReaders

Bitmap formulas look awful, so the only two approaches to reflowable, good quality math available today for browsers and eReaders are MathML and MathJax. MathML is a standard supported natively by most browsers (although, surprisingly, not by Chrome) in which equations are encoded in XML and rendered on the fly. The second approach, MathJax, allows for math to be written in a variety of formats (including LaTeX and MathML itself) and the rendering is performed by Javascript; obviously, this requires a browser (or a reader) equipped with Javascript and sufficient computational power. MathML is "understood" by MathJax, so I chose **MathML**.

A.2 eReaders and eBook Formats

Today, the most common devices to read eBooks are:

1. Kindle Paperwhite (or similar e-paper devices)
2. iPad or Android tablets
3. PCs

On the other hand, there are about *thirty* eBook formats in use worldwide. Many of these are either specialized formats (e.g. comic books) or proprietary formats (e.g. Sony reader format). The most important items for our purposes are:

- EPUB: an open standard now in its 3rd version based on HTML+CSS. Supports MathML and Javascript.
- iBook: a non-standard EBOOK subset by Apple; not compatible with EPUB readers. Supports MathML and Javascript.

- AZW: Kindle's proprietary format (now in version 8). Does NOT support MathML.
- PDF: although not an ebook standard per se, still very relevant. Supports Javascript.

Each standard should be evaluated according to two independent criteria: how beautiful the books look and what devices these book can be read on; additionally, one should consider DRM and marketing channels.

- iBook produces very high quality books since the books only work on Apple devices and therefore the environment is fully controlled; however, books only work on Apple devices.
- AZW works natively on Kindle but Amazon has Kindle reader software for all platforms (including Apple devices)
- EPUB is an open format and readers exist for tablets and PCs (but not for the Kindle). However, your mileage may vary with each reader since the standard is open and potentially very broad.
- PDFs are supported by all devices but reflowing is not possible.

AZW (the Kindle format used by the "classic" Kindles) does not support MathML so equations must be rendered as graphics. However, and this is critical, the AZW format *does not support inline graphics*, so inline equations look terrible. For classic Kindles, the best approach is simply to produce a fixed-layout PDF with a size compatible with the Kindle screen. (This is what Amazon does already for existing textbooks).

iBook is a closed standard and Apple handles the sales channel with an iron fist. Since iBook is basically a proprietary dialect of EPUB and since excellent EPUB readers exist for iPad, iBook would be difficult to recommend over EPUB.

EPUB therefore seems to be the obligatory choice although, being an open standard, one should not expect the flawless level of device support offered by the proprietary formats.

A.3 LaTeX Compilation

Since the chosen target format, EPUB, is based on XHTML, the first order of business is finding a good tool to convert LaTeX to HTML. Since TeX is a Turing-complete programming language, conversion is anything but trivial and, indeed, independently of the chosen tool, there will be limitations on the external packages that can be used in a convertible a manuscript.

Amongst the available conversion tool, I chose **LaTeXML** [3] since it formats equations in MathML and it supports a reasonable number of external packages via

so called “bindings” (see Figure 1 and [4] for the up-to-date list). A binding is a piece of custom code that makes the package understandable by LateXML; writing a custom binding seems to be a daunting task so we will not attempt that here.

LaTeXML proceeds in two passes: first it converst LaTeX to XML, then it formats the XML via style sheets into XHTML.

A.4 Conversion to EPUB

The EPUB format is simply a compressed archive containing a set of XHTML pages (the chapters of the book) plus CSS files and a variety of indexing files. Unfortunately, direct conversion to EPUB from LaTeXML's XML file is still experimental and undocumented. Until this changes, I found that the most robust way to package the XHTML file into an EPUB compliant document is to use Calibre's `ebook-convert` utility [7]

References

[1] `http://www.sp4comm.org/multipub`

[2] `http://idpf.org/epub`

[3] `http://dlmf.nist.gov/LaTeXML/`

[4] `http://dlmf.nist.gov/LaTeXML/manual/included.bindings/`

[5] `https://www.ctan.org/pkg/dsptricks`

[6] `https://www.virtualbox.org/`

[7] `http://calibre-ebook.com/`