A newbie's experiences with Lilypond, Lilypond-book, LaTeX and Perl

Joe McCool
Southern Regional College, Ireland
mccoolj (at) src dot ac dot uk
http://benburb.demon.co.uk/apache2-default/joe.html

Abstract

The author is an active Irish traditional musician. He is also a keen inland boater. He is having a lot of fun composing a book on "Traditional Music for Boaters".

In this paper he describes his successes and frustrations using Lilypond, Lilypond-book, LaTeX and ABC musical notation. Lilypond and LaTeX have a lot in common. Neither are WYSIWYG, neither demand GUI's. Both compile simple flat files to produce beautiful graphical output.

Lilypond's original manifestations produced output directly for LaTeX, but of late users writing books have been encouraged to use Lilypond-book. This looks for Lilypond code within LaTeX source files and produces graphics and associated instructions which can then be processed by LaTeX.

Most joy has come from automating these processes via GNU/Linux and Perl.

1 What's out there

By definition classical music has an inherent connection with books and text. Traditional music does not. Classical music is written down in scores. Historically, traditional music is not. It is largely an aural medium and tunes are learnt by ear. Having said that, musical scores do now have an important role to play. More and more young traditonal musicians are learning how to read scores. Printed material allows us to store pieces that would otherwise be lost and pencil and paper is a useful aid to composition.

Musicians like to share pieces and the arrival of the personal computer has made this easier than ever. But, where years ago we popped manuscripts in the post, we now need to share scores electronically. This has brought about a plethora of programs and systems that enable us to do so.

1.1 ABC notation

In the 1980's Chris Walshaw, then at the University of Cambridge, began writing out fragments of folk/traditional tunes using letters to represent the notes. This became gradually formalised into what is known as the ABC "standard".[1] Numerous small programs have appeared to convert ASCII files of ABC to printed scores.[2] There are also programs to convert ABC code to midi.

Here's an example of an ABC file:

```
X: 1
T:The trout
C:Franz Schubert
O:Austria
M:C|
L:1/8
Q:1/4=160
K:C
G2|"C"c2c2e2e2|"C"c4G2G2|"G"G3G dcBA|"G"
G4 z2G2|"C"c2c2 e2e2|"C"c4G2c2|"G"B2AB
....
```

The K: field represents the tune's key. All lines below this contain the music. Lines above are header fields, with X: representing an index for this particular tune in a file of other ABC's. Notes in quotes represent accompaniment chords.

At first sight ABC seems ideal for what I wish to do. It has a simple input format. It can output PostScript files that I can incorporate into LaTeX documents. ABC programs are open source and, most importantly, there are huge collections of ABC source files available on the internet.

But, I have a few issues with ABC:

- there is a gross lack of standardisation. What standardisation exists is often ignored by the authors of ABC files.
- I need a system comparable to LaTeX in terms of typesetting quality. ABC does not have the fine grained control of LaTeX.
- It is difficult to avoid clashing problems: note heads clashing with bar numbers, or grace notes clashing with accidentals.
- At the time I started my project, the ABC mailing list seemed to vanish!

[1] www.walshaw.plus.com/abc
[2] for example, moinejf.free.fr

1.2 Commercially available software

Under Microsoft Windows, several commercial programs are available for typesetting music. Finale,[3] Sibelius,[4] and Cakewalk[5] are well known. Noteworthy Composer is available as shareware.[6]

1.3 MusicTeX and MusiXTeX

Both of these, based on what I could find out about them,[7] appeared too complex for me. They did not seem to have an active development or support community.

1.4 Lilypond

I rejected the commercial products and Noteworthy because they use proprietary file formats and they rely on GUI interfaces. I am a traditionalist in more ways than one. Long ago I realised the power, elegance and beauty of plain ASCII files under GNU/Linux. Hence my final choice of Lilypond.[8][9]

- Lilypond's originators have objectives very similar to those of LaTeX: "to print music in the best traditions of classical engraving with minimum fuss".
- Lilypond uses plain ASCII, not dissimilar to ABC.
- Lilypond enjoys ongoing development.
- Its documentation is excellent.
- Very active user support via mailing lists.
- Very fast keying of source files:
  - Note durations need stating once only. In the input `a4 b`, the notes `a` and `b` have the same duration.
  - Notes can be raised an octave using a following `'` or lowered by a following `,`. Lilypond also provides a `relative` mode in which it will position notes on the scale in a common sense, reasonable fashion.

    ```
    \relative c'' {
     b c d c b c bes a
    }
    ```

    These notes will all be placed within the scale, rather than climbing higher upward.
- key transposition is easy: `\transpose d e ...`
- The excitement and delight I found putting together my first Lilypond scripts was matched only by that of my first LaTeX scripts. Lilypond and LaTeX really are first cousins!

[3] `www.finalemusic.com`
[4] `www.sibelius.com`
[5] `www.cakewalk.com`
[6] `www.noteworthysoftware.com`
[7] `www.tex.ac.uk/cgi-bin/texfaq2html?label=music`
[8] `www.lilypond.org`
[9] In rural Ireland we have a saying: "If you think a donkey will do the job, use a horse!"

1.5 Example Lilypond file

```
\version "2.11.33"
\header {
  composer = "Joe Mc Cool"
  title = "The Eight Lock"
  dedication = ""
}
voicedefault = {
  \relative c'
  \clef treble
  \key g \major
  \time 4/4
  \repeat volta 2 {
    \time 4/4
    \clef treble
    d'4 d8 e d c b4
    ....
  }
  \repeat volta 2 {
    a'4 a8 a b c4.
    ....
  }
}
\include "../new.score.ly"
```

Notice the indentation of code, similar to programming languages. *voicedefault* is a musical object that will subsequently process as a score (see below), or a midi file.[10]

As my project gathered weight I got tired of having to edit individual Lilypond files in order to change the overall look of my book, hence my use of the include statement. `new.score.ly` consists of:

```
\score{
<<
{
\voicedefault
}
>>
  \layout{
  #(layout-set-staff-size 20)
  }
}
```

In order to change the overall look of all my pieces, for example to change the staff size, all I have to do is edit the above.

2 My approach

My approach is constrained by the following goals:

- Each page should contain an integer number of tunes. Classical musicians are happy (or at least willing) to turn a page in the middle of a piece, traditional musicians are not.
- Traditional music is often played in sets. Two or three jigs will be played one after the other,

[10] though midi support is regrettably not good in Lilypond.

or a group of hornpipes. When new tunes are added to the collection, they must be kept as close together as possible to other members of their set, ideally on the same page, or on the same spread.

- Brief texts and footnotes must appear on the page of the tune to which they refer.
- Index entries must have the form '*name : type : page number*'. This reveals the page number and the tune type for each entry.
- The build process should produce midi files.

3 Combining Lilypond and LaTeX

Here is a simple example of a LaTeX file (`small.ly`) containing Lilypond code:

```
\documentclass{article}
\begin{document}
\noindent
Some text before a musical snippet.\\
\begin[quote,fragment]{lilypond}
{
  c' e' g' e'
}
\end{lilypond}
Another snippet:\\
\begin[quote,fragment]{lilypond}
{
  f' g' a' b'
}
\end{lilypond}
Some more text.\\
\end{document}
```

This is processed by the command line:

```
lilypond-book -f latex --psfonts
              --output OUTPUT small.tex
```

And in the `OUTPUT` directory Lilypond creates the following files:

```
lily-2b589ef505-1.eps
lily-2b589ef505.eps
lily-2b589ef505.ly
lily-2b589ef505-systems.tex
lily-2b589ef505-systems.texi
lily-2b589ef505.txt
lily-eb070afcf7-1.eps
lily-eb070afcf7.eps
lily-eb070afcf7.ly
lily-eb070afcf7-systems.tex
lily-eb070afcf7-systems.texi
lily-eb070afcf7.txt
small.dep
small.tex
snippet-map.ly
snippet-names
tmpMQ9ShM.aux
```

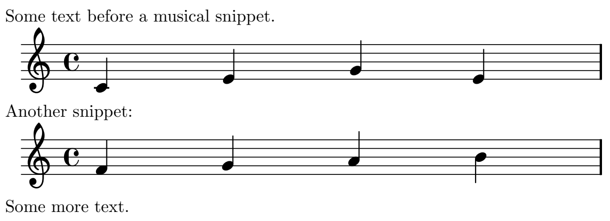

Figure 1: `small.pdf`

Here the file `small.tex` constitutes the final output. This is then processsed by LaTeX in the normal way, with the result shown in Fig. 1.

Lilypond-book creates a graphic for each line of music. It also creates a graphic for the whole snippet — in our small example, `2b589ef505-1.eps` and `2b589ef505.eps`. My project has currently gathered about 200 tunes and the number of small files in `OUTPUT` hovers around 3500!

Contents of `OUTPUT/small.tex`:

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
\noindent
Some text before a musical snippet.\\
{%
\parindent 0pt%
\ifx\preLilyPondExample \undefined%
 \relax%
\else%
 \preLilyPondExample%
\fi%
\def\lilypondbook{}%
\input lily-2b589ef505-systems.tex%
\ifx\postLilyPondExample \undefined%
 \relax%
\else%
 \postLilyPondExample%
\fi%
}
Another snippet:\\
{%
\parindent 0pt%
\ifx\preLilyPondExample \undefined%
 \relax%
\else%
 \preLilyPondExample%
\fi%
\def\lilypondbook{}%
\input lily-eb070afcf7-systems.tex%
\ifx\postLilyPondExample \undefined%
 \relax%
\else%
 \postLilyPondExample%
\fi%
}
Some more text.\\
\end{document}
```

If LaTeX can fit the snippet into a page in its entirety it uses the whole graphic, otherwise it uses individual lines, placing some of the lines on the following page. This breaks my first requirement: pages should contain only an integer number of snippets.

4 Overcoming the first limitation

My first approach to this problem was to wrap the snippet in a Figure environment:

```
\begin{figure}
....
lilypond code
....
\end{figure}
```

An integer number of tunes then appeared on a page, but I found that the positioning of the graphics was inconsistent, particularly at the end of chapters. Google reported that lots of people had suffered from this same problem, but I could find no solutions.

Indeed the suggestion was that I abandon LaTeX altogether and use only lilypond and a particular stylesheet.[11]

I also tried using the standard utility *grep* to find a relation between my LaTeX file, the Lilypond file and the *lilypond-book*-generated `eps` files. I intended then to use conventional `\includegraphics` commands to position the graphics manually. This proved too cumbersome.

My final code ended up as:

```
\noindent
\begin{minipage}{\columnwidth}
\index{mytune:reels}
\lilypondfile{mytune.ly}\\\\
\include{mytune.tex}
\end{minipage}
```

Here *mytune.tex* contains notes pertaining to this particular piece. Wrapped within the minipage, it was guaranteed to appear on the same page. It might also contain footnotes.

This example also shows that if the Lilypond script is long, it can be stored in a file and referred to with `\lilypondfile`; *lilypond-book* then processes its argument.

5 Clever includes

Ideally I would have liked code such as:

```
\newcommand{\lily}[1]{
  \lilypondfile{#1}{#1}}
.....
\lily{lilys/my.tune.ly}
```

[11] `lsr.dsi.unimi.it/LSR/Item?id=368`

but *lilypond-book* complained about not being able to find files. It is just not that clever. It is not able to process my `\newcommand`.

6 Source collections

Ironically the largest collections of traditional music from all over the world are held in ABC files and there are quite a few search engines tuned specifically for searching ABC sites.[12] There is also a Python script available that converts ABC to Lilypond (*abc2ly*).

Again, possibly because of the lack of ABC standards, *abc2ly* does not produce very tidy code and sometimes gets the repeats plain wrong. It is often brought to its knees by idiosyncratic ABC.

7 And then there was Perl

Perl is ideal for processing text. Both LaTeX and Lilypond are text based, so the marriage is obvious. My collecting of ABC files and their subsequent placement in my book is now almost completely automatic:

- ABC file arrives in target folder (often via email)
- A Perl daemon:
  1. makes a backup
  2. cleans up ABC code
  3. creates an index entry
  4. merges text to precede or follow this item
  5. runs *abc2ly*
  6. adds name of lily file to compilation list
- a *make* invocation puts together book version

I think of this process as resembling a trout: the Perl daemon watches for an ABC file arriving as a result of an Internet search — just as a trout watches for minnows! It is not perfect, but I have good error reporting in place and mistakes are easily fixed by hand.

When sufficient new tunes have been added to the repository, another Perl script employs *lilypond-book*, *latex*, *dvips* and *ps2pdf* to produce the final copy.

8 Subversion

Small changes are made to this project daily and sometimes the editing is done using machines on different sites. A small change can have a disastrous effect on the end product. For this reason the whole project is controlled using the *Subversion*[13] version control system.

[12] for example, `trillian.mit.edu/~jc/cgi/abc/tunefind`
[13] `subversion.tigris.org`