

L^AT_EX for usicians

Guido Gonzato, PhD

Version 1.0.1
January 19, 2019



Abstract

This guide shows how to create L^AT_EX documents that include several kinds of music elements, from very simple to quite complex. Music features may consist of music symbols, song lyrics, guitar chords diagrams, lead sheets, music excerpts, guitar tablatures, multi-page scores.

Music can be produced directly by T_EX and L^AT_EX using packages, and also by external scorewriters. Major packages and programs are listed and briefly described, providing ready-to-use examples.

Contents

1	Introduction	3
1.1	Conventions	4
1.2	Preliminaries	5
1.3	Adding Packages	5
1.4	Including PDF files	6
2	Music Symbols	8
2.1	Using Packages	8
2.1.1	Package musicography	9

2.1.2	Package <code>leadsheets</code>	9
2.1.3	Package <code>lilyglyphs</code>	10
2.2	Using Music Fonts	11
3	Song Lyrics	13
3.1	Package <code>guitar</code>	13
3.2	Package <code>gtrcrd</code>	14
3.3	Package <code>songs</code>	15
3.4	Package <code>musixguit</code>	16
3.5	Package <code>leadsheets</code>	17
3.6	Package <code>songbook</code>	18
3.7	Program: <code>chordii</code>	19
4	Guitar Chord Diagrams	20
4.1	Package <code>gchords</code>	20
4.2	Package <code>songs</code>	21
4.3	Package <code>guitarchordschemes</code>	22
5	Sheet Music	23
5.1	Which Notation Software?	24
5.2	Packages <code>musixtex</code> , <code>m-tx</code>	24
5.3	Package <code>gregoriotex</code>	29
5.4	Program: <code>LilyPond</code>	30
5.4.1	<code>lilypond-book</code>	32
5.4.2	Package <code>lilyuatex</code>	33
5.5	Program: <code>abcm2ps</code>	35
5.6	Program: <code>abc2svg</code>	35
5.7	Program: <code>PMW</code>	36
5.8	Program: <code>MUP</code>	37
5.9	Program: <code>MuseScore</code>	38
5.10	Package <code>abc</code>	38
6	Guitar Tablatures	40
6.1	Program: <code>LilyPond</code>	40
6.2	Program: <code>abc2xml.py</code>	41
6.3	Package <code>musixtex</code>	43
6.4	Package <code>guitartabs</code>	44
6.5	Using Guitar Pro Tablatures	45

7 Bits and Pieces	45
7.1 Multimedia Files	45
7.2 Notation Source Files	46
8 Putting It All Together	47
8.1 Package <code>abc</code> , Revisited	47
8.2 Using <code>make</code>	49
9 The End	52
A List of Packages and Programs	53
B Examples	55
B.1 A Complete <code>abc</code> Example	55
B.2 A Complete Songbook Example	59
B.3 A sample <code>Makefile</code>	64
B.4 Verses and Guitar Chords Diagrams	64



1 Introduction

[And] there is no such hobby that it cannot be combined with \LaTeX .

— Clemens Niederberger

I'm a long-time and enthusiastic \LaTeX user, and I'm also an amateur musician; I play folk music on wind instruments. Years ago, I used \LaTeX to typeset my [ABC notation tutorial](#) that I'm still maintaining. Writing that tutorial, I included many PDF music snippets along with the corresponding ABC sources, and I was very satisfied with the result. (Hopefully, other people were satisfied, too.)

Since then, I have become interested in other music notation languages. I have come across many excellent programs and many great packages I wasn't aware of; I have collected many music snippets and I have taken notes. I must say that I'm impressed; there are plenty of options for the musician who is also a \LaTeX user. In fact, \TeX can typeset music by itself using extensions, such as the MusixTeX family. Besides, \LaTeX can easily include music produced by other scorewriters.

So, to write \LaTeX documents that include music we have to solve two problems: how do we make the music in a suitable format? And how do we combine the music with \LaTeX ?

For my own self-training, and in the hope of doing something useful for other musicians, I have decided to write this guide that explains how to solve these problems in many ways. In particular, it shows how to create documents that include many types of music information, from very simple to quite complex:

- **music symbols:** $\sharp \flat$ 

Imagine (John Lennon)

Intro, $\times 2$

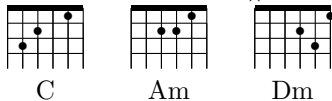
C Cmaj7 F

- **song lyrics:**

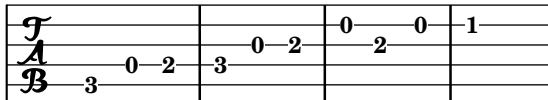
```

C           Cmaj7 F
Imagine there's no heaven
C           Cmaj7 F
It's easy if you try
C           Cmaj7 F
No hell below us
C           Cmaj7 F
Above us only sky

```

- **guitar chord diagrams:**


- **sheet music:**


- **guitar tablatures:**


- **multimedia files:** [play me!](#)

- **notation sources:**

```

X: 1
L: 1/4
K: C
c G/G/ _AG|z !>!B!>!c z|]

```

This guide aims to provide a roundup of available options, along with ready-to-use examples. We will examine a few $\text{\TeX}/\text{\LaTeX}$ packages and several programs that produce high-quality music output; then we will learn how to include the music (as PDF files, sound files, or notation sources) in \LaTeX documents.

1.1 Conventions

This document is meant to be consulted in its PDF version (online, if possible), rather than printed on paper. Hence, it uses many external links.

Links are written in short form whenever possible; for instance, the ABC package will be referred to as [abc](#) instead of <https://www.ctan.org/pkg/abc>.

Sources (\LaTeX or notation languages) are shown in a frame with a light green background; the resulting PDF output, when applicable, has a light gray background. Command line sessions are shown in a frame; user commands are emphasised in boldface.

All software described in this guide is [free and open source](#).

1.2 Preliminaries

To begin with, we need a working \LaTeX system. I strongly recommend that you install the latest release of [TeX Live](#), which I used to typeset this manual and all included examples on a GNU/Linux system (TeX Live 2018). Please note that on Debian-based GNU/Linux systems, and maybe other distributions, the default TeX Live packages may be older. I suppose that other \TeX distributions like [MiKTeX](#) or [MacTeX](#) are just as good, but I have no first-hand experience with them.

Secondly, we need a Bash-based command line environment; this is provided by default in GNU/Linux and macOS systems. Windows users should install the excellent [MSYS2](#) platform.

In most cases, typesetting is done with the common `pdflatex` command. Some packages, though, require the $X_{\text{T}}\LaTeX$ or $\text{Lua}\LaTeX$ engines. In this manual, `pdflatex` is the default command, unless otherwise specified.

This manual is not meant to replace the documentation of the packages and programs that it briefly describes. A few examples are provided to get you started and to whet your appetite, but in most cases they don't cover all available features. The package or program documentation is the primary source of information you should refer to.

Finally, I assume that you are reasonably familiar with \LaTeX . Should you need some information, fine manuals and tutorials are available at the [\$\LaTeX\$ info page](#).

1.3 Adding Packages

TeX Live provides a large number of packages, but in the following sections we will deal with packages that may not be included in less recent releases. Also, some features may be available in package releases that are newer than those included in TeX Live. In these cases, you will have to install the missing (or newer) package manually. The procedure is simple:

1. create this directory structure:

```
$ mkdir -p $HOME/texmf/tex/latex
```

new packages will be installed in this directory tree.

2. get the package (typically as a zip-compressed directory) from your favourite CTAN mirror; let's call it `foo.zip`
3. unpack it in the right place:

```
$ mkdir $HOME/texmf/tex/latex/foo
$ mv foo.zip $HOME/texmf/tex/latex/foo
$ cd $HOME/texmf/tex/latex/foo ; unzip foo.zip
```
4. if no `.sty` file exists, run the command `latex foo.ins` or `latex foo.dtx` to create it;
5. run the command `texhash $HOME/texmf`

Package FOO is now accessible.

If you are installing a package that is newer than that in TeX Live and want L^AT_EX to use it, add this line to your `.bashrc`:

```
export TEXINPUTS=$HOME/texmf/tex/:
```

Please note the double `/` at the end. This line makes T_EX search recursively in `$HOME/texmf/tex/`, then it adds the current value of `TEXINPUTS` (if any) to the search path.

1.4 Including PDF files

It's reasonable to assume that PDF is the most sensible format for final output. PostScript, SVG, PNG and other formats will not be considered, but it's easy to convert PDF files to these formats using applications like [Inkscape](#) or [ImageMagick](#).

Unless you create music directly using MUSIX_TE_X and related packages (Section 5.2), your L^AT_EX document will include music as PDF files. Such files can be short excerpts, i.e. smaller than a page, or span several pages. These files will be included with the `\includegraphics` command (package [graphicx](#)) or with the `\includepdf` command (package [pdfpages](#)), respectively:

```
\documentclass[oneside]{article}
\usepackage{graphicx}
\usepackage{pdfpages}
\usepackage[a4paper,margin=1.5cm]{geometry}
```

```

\begin{document}

This is a short excerpt:

\includegraphics[width=0.8\textwidth]{sample.pdf}

Let's now include several pages:

% pages=- means "all pages"
\includepdf[pages=-,pagecommand={},width=\textwidth]{music.pdf}

\end{document}

```

I suggest that you read the excellent [epslatex](#) guide that explains many details on alignment, size, rotation etc. of included graphics files.

In the case of short excerpts, we need some means of cropping the PDF to its actual contents (bounding box); PDF files, in fact, are usually created as whole pages. Cropping the PDF is accomplished with the free program [pdfcrop](#), a very useful Perl script that depends on [Ghostscript](#) and [PDFedit](#).

Given a file called `music.pdf`, we crop it with these commands:

```

$ pdfcrop music.pdf
PDFCROP 1.38, 2012/11/02 - Copyright (c) 2002-2012 by Heiko Oberdiek.
==> 1 page written on `music-crop.pdf'.
$ mv -f music-crop.pdf music.pdf
$ _

```

Pdfcrop doesn't work on MSYS2; moreover, MSYS2's Ghostscript package lacks the `bbox` device. However, if you install the official release of Ghostscript for Windows, you can use the following shell script, `pdfcrop.sh`. It only works with single-page PDF files:

```

#!/bin/sh

# pdfcrop.sh - for MSYS2 and GhostScript
# Guido Gonzato, PhD. GPL 2 or later.

MYSELF=$(basename $0)

if [ $# -eq 0 ] ; then
  printf "Usage: ${MYSELF} <file.pdf>\n"
  printf "This script uses 'gs' to crop a one-page pdf file.\n\n"

```

```

    exit 1
fi

# GhostScript for Windows is installed in C:\Gs
GS=/c/gs/gs9.26/bin/gswin64c.exe
# GNU/Linux and others:
# GS=/usr/bin/gs
INPUT=$1
PDF=$(basename $1 .pdf)
OUTPUT=$PDF-crop.pdf
GSOPTS="-q -sDEVICE=bbox -dBATC H -dNOPAUSE"

# find out the bounding box
$GS $GSOPTS $INPUT 2>&1 | grep "%B" > $PDF.bbox

# read bbox coordinates in variables
read tmp X1 Y1 X2 Y2 < $PDF.bbox

# write the output, cropped to bbox
$GS -q -o $OUTPUT \
  -sDEVICE=pdfwrite \
  -c "[ /CropBox [$X1 $Y1 $X2 $Y2] /PAGES pdfmark" \
  -f $INPUT

/bin/rm -f $PDF.bbox

echo "$INPUT cropped to $OUTPUT"

```



2 Music Symbols

The simplest music elements we may want to include in our documents are music symbols (*glyphs*). Standard L^AT_EX only provides the math mode commands `\sharp`, `\flat`, and `\natural`: ♯ ♭ ♮. Additional glyphs are provided by several packages; moreover, glyphs provided by music fonts are accessible via X_YL^AT_EX and LuaL^AT_EX.

2.1 Using Packages

The impressive [Comprehensive LaTeX Symbol List](#), Section 7, lists packages that provide a handful of music symbols: `textcomp`, `mnsymbol`, `fdsymbol`, `boisik`, `wasysym`, `stix`, and `arev`.


```

\thispagestyle{empty}

\begin{document}




Leadsheets makes the music symbol font provided by MusiX\TeX\
available as text font and then uses it to define a number
of symbols:

\sharp\ \doublesharp\ \flat\ \doubleflat\ \natural\
\trebleclef\ \altoclef\ \bassclef\ \meterC\ \allabreve\
\meter{12}{8}\ \wholereast\ \halfrest\ \quarterrest\
\eighthrest\ \sixteenthrest\ \Break\ \normalbar\
\leftrepeat\ \rightrepeat\ \leftrightrepeat\ \doublebar\
\stopbar

\end{document}

```

Leadsheets makes the music symbol font provided by MusiX_{TEX} available as text font and then uses it to define a number of symbols:

x b bb q    C C ¹²/₈ - - z y z | : : : ||

This package also provides the `\writechord` command to typeset chords:

```

\documentclass[12pt]{article}
\usepackage[minimal]{leadsheets}
\useleadsheetslibraries{chords}
\thispagestyle{empty}

\begin{document}

Let's typeset some chords: \writechord{Bb7(#9)} \writechord{Bbb6}
\writechord{C#7(b9)} \writechord{C##13}

\end{document}

```

Let's typeset some chords: $B\flat^{7(\#9)}$ $B\flat\flat^6$ $C\sharp^{7(\flat 9)}$ $C\sharp\sharp^{13}$

2.1.3 Package [lilyglyphs](#)

One of the most complete sources of music glyphs is the [lilyglyphs](#) package. It provides all the symbols available in the [Emmentaler](#) music font, which is used by the [LilyPond](#) scorewriter (Section 5.4). LILYGLYPHS only works

with $\text{Xe}\text{L}\text{A}\text{T}\text{E}\text{X}$ and $\text{Lua}\text{L}\text{A}\text{T}\text{E}\text{X}$, and is incompatible with LEADSHEETS and $\text{MUSIX}\text{T}\text{E}\text{X}$. Available glyphs and corresponding commands are listed in Section 3 of the package [documentation](#).

This is a very small example of what LILYGLYPHS provides:

```

\documentclass{article}
\usepackage{fontspec}
\usepackage{lilyglyphs}
\thispagestyle{empty}

\begin{document}

Lilyglyphs makes the music symbol font provided by LilyPond available
as text font and then uses it to define a number of symbols, some
\clefGInline\ of which \clefCInline\ can be used \clefFInline\ inline:

clefs:~ \clefG\ \clefC\ \clefF\ ~time signatures:~
\lilyTimeC\ \lilyTimeCHalf\ \lilyTimeSignature{7}{8}\
~accidentals:~ \sharp\ \flat\ \natural\ \doublesharp\ \flatflat\
~rests:~ \wholeNoteRest\ \halfNoteRest\ \crotchetRest\
~notes:~ \wholeNote\ \halfNote\ \halfNoteDown\ \quarterNote\
\quarterNoteDotted\
~and much, much more.

\end{document}

```

Lilyglyphs makes the music symbol font provided by LilyPond available as text font and then uses it to define a number of symbols, some G of which C can be used F inline:

clefs: G C F time signatures: C C 7 accidentals: \sharp \flat \natural \times $\flat\flat$ rests: — — z
notes: \circ J P J J and much, much more.

2.2 Using Music Fonts

The $\text{Xe}\text{L}\text{A}\text{T}\text{E}\text{X}$ and $\text{Lua}\text{L}\text{A}\text{T}\text{E}\text{X}$ engines use Unicode input by default and support OTF/TTF fonts. We can use these engines to print any character provided by locally installed fonts; these can be listed with the `fc-list` command, provided by the [Fontconfig](#) software.

Some fonts are especially useful for music. Unicode-encoded fonts provide [music symbols](#), which we can find for instance in [GNU FreeFont](#). However, a font standard for music applications called [SMuFL](#) has been developed.

Specifically, [Bravura](#) is a free, SMuFL-compliant music font that provides thousands of high-quality music glyphs. Bravura is available in OpenType format as **Bravura.otf** and **BravuraText.otf**; the first is used for drawing music symbols in scores, the second for inserting music symbols in text. Each glyph is mapped to a numerical code called *code point*; a comprehensive list of glyphs and the corresponding code points is available at <https://www.smufl.org/version>.

The following source shows how to use the glyphs provided by Bravura. We can define new commands for commonly used glyphs, or directly use the `\char"XXXX` syntax to print Unicode characters specifying their code point. The fonts are expected to be installed in `/usr/share/fonts/`; if you install the fonts in a different directory, you'll also have change the following source accordingly:

```
\documentclass{article}
\usepackage{fontspec}
\thispagestyle{empty}
\newfontfamily\brtxt{BravuraText.otf}[Path=/usr/share/fonts/]
\newfontfamily\brv{Bravura.otf}[Path=/usr/share/fonts/]

\newcommand{\clefGi}      {{\brtxt \char"E050}}
\newcommand{\clefCi}      {{\brtxt \char"E05C}}
\newcommand{\clefFi}      {{\brtxt \char"E062}}

\newcommand{\clefG}       {{\brv \char"E050}}
\newcommand{\clefC}       {{\brv \char"E05C}}
\newcommand{\clefF}       {{\brv \char"E062}}
\newcommand{\timeC}       {{\brv \char"E08A}}
\newcommand{\timeCHalf}   {{\brv \char"E08B}}
\renewcommand{\flat}      {{\brv \char"E260}}
\renewcommand{\natural}   {{\brv \char"E261}}
\renewcommand{\sharp}     {{\brv \char"E262}}
\newcommand{\wholeNote}   {{\brv \char"E1D2}}
\newcommand{\halfNote}    {{\brv \char"E1D3}}
\newcommand{\halfNoteDown}{{\brv \char"E1D4}}
\newcommand{\quarterNote} {{\brv \char"E1D5}}

\begin{document}

The Bravura and BravuraText Music fonts provide thousands of music
symbols. BravuraText glyphs are specifically \clefGi\ designed
\clefFi\ to be used \clefCi\ inline:

clefs:~ \clefG\ \clefC\ \clefF\ ~time signatures:~
```










```

\timeC\ \timeCHalf\ ~accidentals:~ \sharp\ \flat\ \natural\
~notes:~ \wholeNote\ \halfNote\ \halfNoteDown\ \quarterNote\
~and much, much more.

\end{document}

```

The Bravura and BravuraText Music fonts provide thousands of music symbols. BravuraText glyphs are specifically designed to be used inline:

clefs:   time signatures:   accidentals:   notes:    and much, much more.



3 Song Lyrics

Several packages make it possible to typeset song lyrics, with varying degrees of complexity. For instance, we may want to write lyrics only; other useful features include chords above lyrics, guitar chords diagrams, transposition, index generation, etc.

3.1 Package **guitar**

This is a very basic and simple to use package that provides a **guitar** environment to add chords above lyrics. Customisation options include alignment, spacing, font, and other low-level details.

A minimal example:

```

\documentclass{article}
\thispagestyle{empty}
\usepackage{guitar}

\begin{document}

\begin{guitar}
  \textbf{Imagine (John Lennon)}

  \emph{Intro, $\times$ 2}
  % use tilde to add some space
  [C] ~ [Cmaj7] ~ [F] ~
  % if chords overlap, add | at the end of the chord, like:

```

```

[C]Imagine there's [Cmaj7|]{no} [F]heaven
[C]It's easy if [Cmaj7|]{you} [F]try
[C]No hell [Cmaj7|]{below} [F]us
[C]Above us [Cmaj7|]{only} [F]sky
\end{guitar}

\end{document}

```

Imagine (John Lennon)

```

Intro, × 2
C Cmaj7 F

C Cmaj7 F
Imagine there's no heaven
C Cmaj7 F
It's easy if you try
C Cmaj7 F
No hell below us
C Cmaj7 F
Above us only sky

```

3.2 Package `gtrcrd`

This is another basic and simple to use package. Customisation options include chord location and name (Do-Re-Mi instead of C-D-E), font, transposition, and spacing. Chords overlaps need manual adjustment.

```

\documentclass{article}
\thispagestyle{empty}
\usepackage{gtrcrd}

\setlength{\crdheight}{2ex} % reduce spacing
\def\crdfont{\footnotesize \itshape \sffamily} % setchord font
\setlength{\parindent}{0pt} % no indentation

\begin{document}

\textbf{Imagine (John Lennon)}

\emph{Intro, $\times$ 2}

% if chords overlap, use \hspace:
\C {\hspace{3mm}} \C[maj7] {\hspace{10mm}} \F ~

\C Imagine there's \C[maj7] no {\hspace{5mm}} \F heaven

\C It's easy if \C[maj7] you {\hspace{2mm}} \F try

```

```
\C No hell \C[maj7] below \F us
\C Above us \C[maj7] only {\hspace{2mm}} \F sky
\end{document}
```

```
Imagine (John Lennon)
Intro, × 2
C Cmaj7 F

C Cmaj7 F
Imagine there's no heaven
C Cmaj7 F
It's easy if you try
C Cmaj7 F
No hell below us
C Cmaj7 F
Above us only sky
```

3.3 Package `songs`

This is a very powerful package that provides many features: chords above lyrics, guitar chords diagrams (Section 4.2), transposition, index generation, multiple columns, and more. Overall, it allows for the creation of complete songbooks.

```
\documentclass{article}
\thispagestyle{empty}
\usepackage[chorded]{songs}

\begin{document}

\renewcommand{\lyricfont}{\small}
\renewcommand{\printchord}{\it\small}
\afterpreludeskip=-18pt
\beforepostludeskip=-8pt

\begin{songs}{}
\beginsong{Imagine}[by={John Lennon}]
\beginverse
\emph{Intro, $\times$ 2}
\C \Cmaj7 \F
\C Imagine there's \Cmaj7no \Fheaven
\C It's easy if \Cmaj7you \Ftry
\C No hell \Cmaj7below \Fus
\C Above us \Cmaj7only \Fsky
\endverse
\end{songs}
```

```
\endsong
\end{songs}

\end{document}
```

1 *Imagine*
John Lennon

1. *Intro, × 2*
C Cmaj7 F

C Cmaj7 F
Imagine there's no heaven

C Cmaj7 F
It's easy if you try

C Cmaj7 F
No hell below us

C Cmaj7 F
Above us only sky

3.4 Package `musixguit`

This package is integrated with `musixtex` (Section 5.2), and is also capable of producing sheet music and guitar chord diagrams. Its documentation is written in German; if you can't read it don't worry, the provided examples are easy to understand. Chords overlaps need manual adjustment.

```
\documentclass{article}
\usepackage{musixguit}
\thispagestyle{empty}

\begin{document}

\textbf{Imagine (John Lennon)}

\begin{song}

\emph{Intro, $\times$ 2}

\chord{C} ~ \chord{Cmaj7} {\hspace{8mm}} \chord{F}

\chord{C}Imagine there's \chord{Cmaj7}no~~~ \chord{F}heaven

\chord{C}It's easy if \chord{Cmaj7}you~~ \chord{F}try
```



```

\chord{C}No hell \chord{Cmaj7}below \chord{F}us
\chord{C}Above us \chord{Cmaj7}only~ \chord{F}sky
\end{song}
\end{document}

```

Imagine (John Lennon)

Intro, × 2

C Cmaj7 F

C Cmaj7F
Imagine there's no heaven

C Cmaj7 F
It's easy if you try

C Cmaj7 F
No hell below us

C Cmaj7 F
Above us only sky

3.5 Package `leadsheets`

This package provides many features: music and symbols, chords, MuseJazz style, a **song** and a **verse** environment, transposition, a **leadsheet** class, and templates. Overall, this package allows for the creation of complete songbooks.

```

\documentclass{article}
\usepackage[full]{leadsheets}
\thispagestyle{empty}

\begin{document}

\begin{song}{title={Imagine}, music={John Lennon}}
\begin{verse}

Intro, $\times$ 2\\
\chord{C}~ \chord{Cmaj7}~ \chord{F}~

% The ^ character is a shortcut for \chord
\chord{C}Imagine there's ^{Cmaj7}no ^{F}heaven \\
^{C}It's easy if ^{Cmaj7}you ^{F}try \\
^{C}No hell ^{Cmaj7}below ^{F}us \\
^{C}Above us ^{Cmaj7}only ^{F}sky \\

```

```

\end{verse}
\end{song}

\end{document}

```

Imagine

Intro, × 2
C C^{maj7} F

C C^{maj7} F
Imagine there's no heaven
C C^{maj7} F
It's easy if you try
C C^{maj7} F
No hell below us
C C^{maj7} F
Above us only sky

3.6 Package *songbook*

This is another powerful package that provides support for chords, songs, overhead transparencies, and index generation.

```

\documentclass{article}
\usepackage[chordbk]{songbook}
\thispagestyle{empty}

\begin{document}

\textbf{Imagine (John Lennon)}

% \medskip

\emph{Intro, $\times$ 2}

\Ch{C}~ \Ch{Cmaj7}~ \Ch{F}~

\Ch{C}{Imagine} there's \Ch{Cmaj7}{no} \Ch{F}heaven

\Ch{C}{It's} easy if \Ch{Cmaj7}{you} \Ch{F}{try}

\Ch{C}No hell \Ch{Cmaj7}{below} \Ch{F}us

```

```
\Ch{C}{Above} us \Ch{Cmaj7}{only} \Ch{F}{sky}
\end{document}
```

Imagine (John Lennon)

Intro, × 2

C Cmaj7 F

C Cmaj7 F
Imagine there's no heaven

C Cmaj7 F
It's easy if you try

C Cmaj7 F
No hell below us

C Cmaj7 F
Above us only sky

3.7 Program: *chordii*

Chordii is a free command-line program, released under the [GNU GPL](#). It uses a simple text notation to typeset songs in PostScript format, complete with chords and guitar chord grid.

This is the source of a song written in Chordii format:

```
{titles:left}
{title:Imagine}
{st:John Lennon}

(Intro, x 2)
[C] [Cmaj7] [F]
[C]Imagine there's [Cmaj7]no [F]heaven
[C]It's easy if [Cmaj7]you [F]try
[C]No hell [Cmaj7]below [F]us
[C]Above us [Cmaj7]only [F]sky
```

We typeset the score with these commands:

```
$ chordii -a imagine.cho > imagine.ps
$ ps2pdf imagine.ps
$ _
```

The `-a` switch means “Automatic single space lines without chords”. The resulting song is:

Imagine
John Lennon

(Intro, x 2)
C Cmaj7 F

C Cmaj7 F
Imagine there's no heaven
C Cmaj7 F
It's easy if you try
C Cmaj7 F
No hell below us
C Cmaj7 F
Above us only sky



The image shows three guitar chord diagrams. The first is for C major, with an 'x' on the 6th string and open strings on 1-5. The second is for C major 7, with an 'x' on the 6th string and open strings on 1-5, and a half note on the 2nd string. The third is for F major, with a half note on the 1st string and full notes on 2-4.



A musical staff showing a sequence of notes: G4, A4, B4, C5, B4, A4, G4.

4 Guitar Chord Diagrams

Guitar players may need to print *guitar chords diagrams* and *guitar tablatures*; both can be made with L^AT_EX packages and external programs. As a matter of fact, these music features are not limited to the guitar; chord diagrams and tablatures apply to other stringed instruments as well.

Tablatures are a form of music notation that is usually employed to show how a *melody* should be fingered on the fretboard. Since tablatures are just a special form of music notation, we will deal with them after the section about sheet music. For the moment, let’s see how to do guitar chord diagrams.

For an authoritative list of standard guitar chords diagrams, I suggest that you visit [this useful page](#).

4.1 Package `gchords`

This package makes it possible to print guitar chord diagrams using the `\chord` command that employs a simple syntax:

```
\chord{fret number}{fingering}{chord name}
```

- *fret number* can be `{t}`, which means top fret, or `{t}` followed by a digit that denotes the fret;
- *fingering* is explained in the example below;
- *chord name* is the given chord name.

The `\chords` command prints a row of chords, each defined by a `\chord` command:

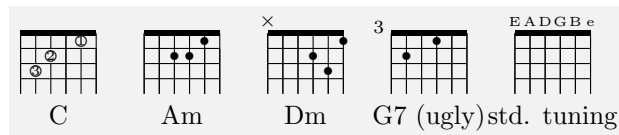
```
\documentclass{article}
\usepackage{gchords}
\thispagestyle{empty}

\begin{document}

\def\numfrets{4}

\chords{ % print a row of chords
% fingers: n, x, o, p{n}, f{n}
% C chord: finger 3 pos. 3, finger 2 pos. 2, finger 1 pos. 1
\chord{t}{n,f3p3,f2p2,n,f1p1,n}{C}
\chord{t}{n,n,p2,p2,p1,n}{Am}
\chord{t}{x,n,n,p2,p3,p1}{Dm}
\chord{t3}{n,p2,n,p1,n,n}{G7 (ugly)}
{\tiny % font size for string labels, t{X}
\chord{t}{t{E}n,t{A}n,t{D}n,t{G}n,t{B}n,t{e}n}
{std. tuning} }
}

\end{document}
```



4.2 Package `songs`

We met this package in Section 3.3. It provides an easy way to print guitar chord diagrams (referred to as “guitar tablatures” in the package). The `\gtab` command defines chord diagrams using a very simple syntax:

```
\gtab{chord name} {fret:strings:fingering}
```

- *chord name* is the given chord name;

- *fret* is an optional fret number;
- *strings* is a string of digits denoting the strings that compose the chord;
- *fingering* is an optional string of digits denoting the fingers to use.

```

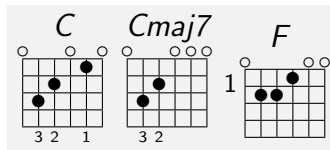
\documentclass{article}
\thispagestyle{empty}
\usepackage[chorded]{songs}

\begin{document}

% \gtab{chord name}{fret:strings:fingering}
\gtab{C}{032010:032010} \gtab{Cmaj7}{032000:032000}
\gtab{F}{1:022100}

\end{document}

```



4.3 Package `guitarchordschemes`

This package enables the creation of fully customisable guitar chord diagrams and scales. The main command is `\chordscheme`, and is quite self-explanatory:

```

\documentclass{article}
\usepackage{guitarchordschemes}
\thispagestyle{empty}

\begin{document}

% general parameters
\setchordscheme{
  rotate=-90,
  x-unit=2.5mm,           % chord size, x
  y-unit=3mm,            % chord size, y
  name-format=\bfseries, % chord name font
  finger-format=\bfseries%
}

```

```

    \footnotesize           % fingering font
}

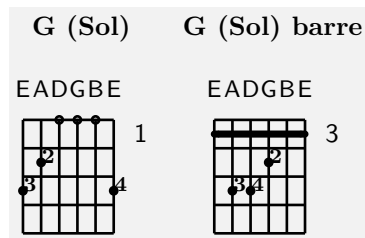
\newsavebox{\Gstd}
\savebox{\Gstd}{
\chordscheme[           % define the chord
  name = G (Sol),       % chord name
  position = 1,         % first fret position
  finger = {2/5:2} ,    % fret, string, finger
  finger = {3/6:3} ,    % fret, string, finger
  finger = {3/1:4} ,    % fret, string, finger
  ring = {2,3,4}       % open strings
]}

\newsavebox{\Gbarre}
\savebox{\Gbarre}{
\chordscheme[
  name = G (Sol) barre,
  position = 3,
  barre = 1/1-6,       % fret, string range
  finger = {2/3:2},
  finger = {3/5:3},
  finger = {3/4:4}
]}

\usebox{\Gstd}\usebox{\Gbarre}

\end{document}

```



5 Sheet Music

In this section, we will learn how to produce sheet music in PDF format for inclusion in \LaTeX documents. Sheet music can be typeset by \TeX itself, but we can also use the output produced by one of the several free and open source scorewriters.

In the next few sections, we will briefly examine each option. Music notation languages will not be explained, for obvious space reasons. Minimal sources will be listed to give minimal working examples and to get acquainted with the notation complexity.

5.1 Which Notation Software?

I'm not going to suggest any particular music notation software. I do have my personal preferences; I find some notations easier than others; some programs have more features than others; and so on. Nevertheless, I think that all packages and programs described in the following sections have their advantages, and all of them can typeset beautiful scores. Unless you need specific features, choosing a typesetting software is mostly a matter of personal preference. Let's put it this way: the best program is the one that lets you obtain the results you need, with the least effort, in the least amount of time, and the most fun.

All software described below is capable of typesetting complex Western music: multiple voices and staves, symbols, dynamics, lyrics, text annotations; some of them can also do tablatures and more.

5.2 Packages `musixtex`, `m-tx`

In the beginning, `TEX` could not typeset music, and everyone was unhappy. Then the `mtex` (aka `MUTEX`) package was created, but it was limited and not very easy to use. Then `MUTEX` begat `MUSICTEX`, which begat `pmtex`; then `MUSICTEX` begat `musixtex`, which begat `pmx`, which begat `m-tx`. I hope I got the storyline right. Each package was a simpler interface to its predecessors.

In reverse order: `M-TX` is a preprocessor to `PMX`, which in turn is a preprocessor for `MUSIXTEX`, which does the actual music typesetting via `TEX`. It goes without saying that `M-TX` is the simplest to use, while `MUSIXTEX` is the most difficult to use but also the most powerful. Other packages related to `MUSIXTEX` are the following:

- `autosp` generates note-spacing commands for `MUSIXTEX` scores;
- `bagpipe` provides support for typesetting bagpipe music;
- `bizantinemusic` facilitates the writing of Byzantine music;
- `figbas` provides mini fonts for figured bass notation in `MUSIXTEX`;
- `gregoriotex` provides engraving of Gregorian Chant (Section 5.3);

- [lyluatex](#) provides commands to include LilyPond scores in Lua \LaTeX documents (Section 5.4.2);
- [musixtnt](#) is an extension library that enables transformations of the effect of notes commands;
- [pmxchords](#) produces chord information to go with pmx output;
- [snote](#) provides shape notes for MUSIX\TeX ;
- [texmuse](#) is a music typesetting system using \TeX and Metafont.

From [musixtex](#) home page:

MusiX \TeX provides a set of macros, based on the earlier Music \TeX , for typesetting music with \TeX .

MUSIX \TeX is quite low-level, and the user must take care of such details as beam slope and note spacing; several examples are available [here](#). Notably, it works with plain \TeX . (Ever tried plain \TeX ?) A minimal example (`sample-mtex.tex`):

```
% bare MusiXTeX example

\input musixtex
\nopagenumbers

\setstaves1{1}          % a single stave
\setclef1{\treble}      % with a treble clef
\generalmeter{\meterC} % common time
\nobarnumbers          % what it says
\startextract           % a short music piece
% \qu = quarter note, stem up;
% \ql = quarter note, stem down;
% \Notes, \en = start and end of note line
\Notes \qu c \qu d \qu e \qu f \en
\bar
\Notes \qu g \qu{'a} \ql b \ql c \en
\endextract

\end
```

We typeset the score with these commands:

```
$ tex sample-mtex.tex
$ dvips sample-mtex.dvi
$ ps2pdf sample-mtex.ps
$ _
```



MUSIXTEX input can also be embedded in L^AT_EX documents (`sample-m-latex.tex`):

```
\documentclass{article}
\usepackage{musixtex}
\thispagestyle{empty}

\begin{document}

A short music excerpt in MusiX\TeX:

\medskip

\begin{music}
  \smallmusicsize
  \instrumentnumber{1}
  \setstaves1{1}
  \generalmeter{\meterC}
  \nobarnumbers
  \startextract
  % bar 1
  \Notes \qu c \en % C
  \notes \ibu1d2\qb1d\tbu1\qb1e \en % beamed DE
  \notes \ibu1g2\qb1f\qb1g%
    \qb1{'a}\tbu1\qb1b \en % beamed FGAB
  \bar % bar 2
  \Notes \ql{'c} \en % c
  \notes \ibu1{'b}{-3}%
    \qb1b\tbu1\qb1a \en % beamed BA
  \notes \ibu1{g}{-3}%
    \qb1g\qb1f\qb1e\tbu1\qb1d \en % beamed GFED
  \bar % bar 3
  \notes \ibu1f0\qb1c\qb1g\qb1e\tbu1\qb1g% % beamed CGEG
    \ibu1f0\qb1c\qb1g\qb1e\tbu1\qb1g \en % beamed CGEG
  \bar % bar 4
```

```

\Notes \qu c\qu e\qu c\qp \en      % CEC
\endextract

\end{music}

\end{document}

```

A short music excerpt in MusiX_{TEX}:



Admittedly it looks a bit arcane, and I'll point you to the [documentation](#) for explanations. I suggest that you learn at least the basics of it; several useful packages are based on this syntax.

M-Tx employs a much simpler notation than MUSIX_{TEX}. This is a standalone music sample (`sample-mtx.mtx`) that produces the same music as the above MUSIX_{TEX} score:

```

% music sample in M-Tx notation

Title: \bigtype Music sample in M-tx
Style: Solo
Meter: C
Width: 140mm

c4 d8 e f g a b | c4 b8 a g f e d | c8 g+ e g c- g+ e g | c4- e c r |

```

We typeset the score with this command:

```

$ musixtex sample-mtx.mtx
This is musixtex.lua version 0.21.
==> This is M-Tx 0.63a (Music from TeXt) <8 April 2018>
==>> Input from file sample-mtx.mtx
...
sample-mtx.pdf generated by ps2pdf.
$ _

```

which produces `sample-mtx.pdf`. In older versions of M-Tx, the command was `m-tx`; it has now been retired and replaced by `musixtex`. This is the resulting score:

Music sample in M-Tx



Music in M-TX format can be easily included in L^AT_EX documents. M-TX provides the `Score`, `excerpts`, and `mus` environments to include complete pieces, short excerpts, and inline short excerpts respectively. Let's see how to use `excerpts` and `mus`; a few steps are required.

First of all, the L^AT_EX source that includes the M-TX music files must not have a `.tex` extension; `.ltx` or `.latex` are ok. Let's call our sample file `sample-latexmtx.ltx`.

Secondly, we need the `mtxlatex.sty` style file, which is not installed by default but is found in the M-TX documentation directory. In TeX Live, `mtxlatex.sty` is located in directory `/usr/share/doc/texlive-doc/generic/m-tx`. This file must be copied to the same directory as the source file; all included M-TX files must be copied there too.

We already met `sample-mtx.mtx`; the following is another short excerpt, `scale.mtx`. It produces a 30mm wide scale:

```
Style: Solo
Meter: C
Size: 13pt
Width: 30mm

c8 d e f g2 |
```

Finally, this is the main file `sample-latexmtx.ltx`:

```
\documentclass[12pt]{article}
\usepackage{mtxlatex} % usually not installed
\thispagestyle{empty}

\mtxlatex

\begin{document}

This \LaTeX{} document includes music written in M-Tx. The
\texttt{mus} environment includes music inline:
```


This is a sample GABC source, `kyrie.gabc`, taken from the package documentation:

```
name:Kyrie XVII;
%%
(c4)KY(f)ri(gfg)e(h.) *()
e(ixjvIH'GhvF'E)lé(ghg')i(g)son.(f.)
<i>bis</i>(::)
```

We also need an auxiliary \LaTeX source, `kyrie.tex`:

```
\documentclass{article}
\thispagestyle{empty}
\usepackage[autocompile]{gregoriotex}

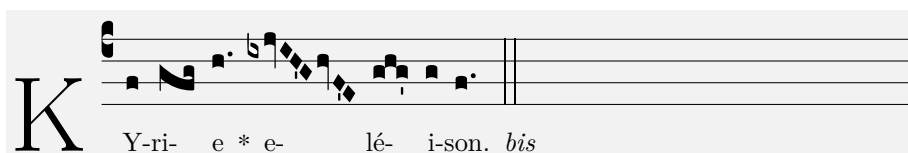
\begin{document}

\gregorioscore{kyrie}

\end{document}
```

We typeset the score with this command:

```
$ lualatex kyrie.tex
...
Output written on kyrie.pdf (1 page, 11290 bytes).
Transcript written on kyrie.log.
$ _
```



5.4 Program: *LilyPond*

LilyPond is a free and multiplatform scorewriter, released under the GNU GPL. From the LilyPond home page:

LilyPond is a music engraving program, devoted to producing the highest-quality sheet music possible. It brings the aesthetics of traditionally engraved music to computer printouts. LilyPond is free software and part of the [GNU Project](#).

LilyPond uses a simple text notation for music input; output is PDF by default. This is our usual music sample in LilyPond format, `sample-lilypond.ly`:

```
% music sample in LilyPond notation

\version "2.18.2"

\paper{
  indent = 0 \mm
}

\header {
  title = "Music sample in LilyPond"
  tagline = "" % no footer
}

\relative c' {
  \time 4/4
  \clef treble
  c4 d8 e f8 g a b | c4 b8 a g8 f e d |
  c8 g' e g c,8 g' e g | c,4 e c r \bar "|."
}
```

We typeset the document with this command:

```
$ lilypond sample-lilypond.ly
GNU LilyPond 2.18.2
Processing `sample-lilypond.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Finding the ideal number of pages...
Fitting music on 1 page...
Drawing systems...
Layout output to `sample-lilypond.ps'...
Converting to `./sample-lilypond.pdf'...
Success: compilation successfully completed
$ _
```

LilyPond is capable of typesetting many different kinds of music, and it is among the most complete and powerful scorewriters available. If you want to find out more, the documentation page is [here](#).

5.4.1 `lilypond-book`

LilyPond also provides the `lilypond-book` command that can be used to easily embed LilyPond sources in special \LaTeX documents. These should have a `.lytex` extension, like the following source (`sample-lilybook.lytex`):

```
\documentclass{article}
\thispagestyle{empty}

\begin{document}

This is a LilyPond snippet \begin{lilypond} {c' e' g'}
\end{lilypond} embedded in the \LaTeX{} source.

This is another LilyPond excerpt that uses the \texttt{lilypond}
environment:

\medskip

\begin{lilypond}
  \score {
    \relative c' {
      \time 4/4
      \clef treble
      c4 d8 e f8 g a b | c4 b8 a g8 f e d |
      c8 g' e g c,8 g' e g | c,4 e c r \bar "|."
    }
  } % end of score
\end{lilypond}

\medskip

End of document.


\end{document}
```

We typeset the score with these commands:



```

$ lilypond-book -f latex -o /tmp \
  --lily-output-dir=/tmp sample-lilybook.lytex
lilypond-book (GNU LilyPond) 2.18.2
...
Writing `/tmp/sample-lilybook.tex'...
$ cd /tmp
$ pdflatex sample-lilybook.tex
...
Output written on sample-lilybook.pdf (1 page, 60934 bytes).
Transcript written on sample-lilybook.log.
$ _

```

This is a LilyPond snippet  embedded in the \LaTeX source.

This is another LilyPond excerpt that uses the `lilypond` environment:



End of document.

5.4.2 Package `lyluatex`

This package may be missing in less recent releases of TeX Live, so you might have to install it manually. `LYLUATEX` provides a native `\Lua\text{\LaTeX}` environment that offers the same functionality as `lilypond-book`, even though the output is different. This source is `sample-lyluatex.tex`:

```

\documentclass{article}
\usepackage{lyluatex}
\thispagestyle{empty}

\begin{document}

```

This is a LilyPond snippet `\lilypond{c' e' g'}` embedded in the text using the `\verb|\lilypond|` command.

This is another LilyPond excerpt that uses the `\texttt{lilypond}` environment:

```
\medskip

\begin{lilypond}
  \score {
    \relative c' {
      \time 4/4
      \clef treble
      c4 d8 e f8 g a b | c4 b8 a g8 f e d |
      c8 g' e g c,8 g' e g | c,4 e c r \bar "|."
    }
  } % end of score
\end{lilypond}

\medskip

End of document.

\end{document}
```

We typeset the score with this command:

```
$ lualatex --shell-escape sample-lyluatex.tex
...
Output written on sample-lyluatex.pdf (1 page, 22334 bytes).
  Transcript written on sample-lyluatex.log.
$ _
```

This is a LilyPond snippet  embedded in the text using the `\lilypond` command.

This is another LilyPond excerpt that uses the `lilypond` environment:



End of document.

5.5 Program: [abcm2ps](#)

abcm2ps is a free and multiplatform scorewriter, released under the GNU GPL.

This program is currently one of the best implementations of the [ABC notation](#), which describes itself as:

the text-based music notation system and the *de facto* standard for folk and traditional music.

In fact, this notation is specifically designed to meet the needs of traditional musicians; hundred of thousands (really!) of tunes in ABC formats are available. ABC is a simple text notation, originally designed for single-voice music but currently capable of producing complex polyphonic scores.

This is our usual music sample in ABC notation (**sample-abc.abc**):

```
% music sample in ABC notation
X: 1
T: Music sample in ABC
M: C
L: 1/4
K: C
%
C D/E/ F/G/A/B/|c B/A/ G/F/E/D/|C/G/E/G/ C/G/E/G/|CECz|]
```

We typeset the score with these commands:

```
$ abcm2ps -c -O= sample-abc.abc
abcm2ps-8.14.1 (2018-11-15)
File sample-abc.abc
Output written on sample-abc.ps (1 page, 1 title, 20503 bytes)
$ ps2pdf sample-abc.ps
$ _
```

5.6 Program: [abc2svg](#)

abc2svg is a free and multiplatform scorewriter, released under the GNU GPL. It's basically **abcm2ps** rewritten in JavaScript.

Although **abc2svg** can be integrated in a [web-based editor](#), it's a command-line program. It reads an ABC source file and turns it to **xhtml**:

```
$ abc2svg file.abc > file.xhtml
$ _
```

The resulting `.xhtml` can then be loaded into any web browser and printed to PDF. [Google Chrome](#) or [Chromium](#) are the recommended browsers.

We can run the whole procedure non-interactively, entirely in the command line. This only works in GNU/Linux and macOS:

```
$ abc2svg tunes.abc > tunes.xhtml
$ chromium-browser --headless --print-to-pdf=tunes.pdf tunes.xhtml
... many log messages ...
... Written to file tunes.pdf.
$ _
```

However, in this case Chromium will add headers and footers to every page. To remove them and obtain a clean PDF file, run the command:

```
$ pdfcrop --margins "0 -9 0 -9" --clip tunes.pdf
PDFCROP 1.38, 2012/11/02 - Copyright (c) 2002-2012 by Heiko Oberdiek.
==> 79 pages written on `tunes-crop.pdf'.
$ _
```

The same result can be obtained using [pdffpages](#) directly:

```
\documentclass{article}
\usepackage{pdffpages}

\def\tunes{tunes.pdf} % PDF file to trim

\begin{document}

\includepdf[pages=-,pagecommand={},%
width=\paperwidth,trim={0 0.9cm 0 0.9cm},clip]{\tunes}

\end{document}
```

5.7 Program: *PMW*

Philip's Music Writer (PMW) is a free and multiplatform scorewriter, released under the GNU GPL. From the PMW home page:

Philip's Music Writer (PMW) is a computer program for high quality music typesetting.

PMW uses a simple text notation for music input and produces output in PostScript.

This is our usual music sample in PMW format (`sample-pmw.pmw`):

```
@ music sample in PMW

Heading "|Music sample in PMW"
Key C
Time 4/4

[stave 1 treble 1]
c d- e-; f-g-a-b-; | c' b- a-; g-f-e-d-; |
c-g-e-g-; c-g-e-g-; |c e c r |
[endstave]
```

We typeset the score with these commands:

```
$ pmw -includefonts sample-pmw.pmw
$ ps2pdf sample-pmw.ps
$ _
```

5.8 Program: *MUP*

MUP is a free and multiplatform scorewriter, released under a free license. From the *MUP* home page:

Mup takes a text file as input and produces very high quality PostScript output for printed music. It can handle both regular notation and tablature notation. It can also produce MIDI output.

This is our usual music sample in *MUP* format (`sample-mup.mup`):

```
// music sample in MUP notation

header
  title "Music sample in MUP"
```

```
score
  time=4/4

music
  1: 4c; 8d bm; e ebm; f bm; g; a; b ebm;
  bar
  1: 4c+; 8b bm; a ebm; g bm; f; e; d ebm;
  bar
  1: 8c bm; g; e; g ebm; c bm; g; e; g ebm;
  bar
  1: 4c; e; c; r;
endbar
```

We typeset the score with these commands:

```
$ mup -F sample-mup.mup
Mup - Music Publisher Version 6.6
Copyright (c) 1995-2017 by Arkkra Enterprises.
Mup is free software. Use -l option to see license terms.
$ ps2pdf sample-mup.ps
$ _
```

5.9 Program: *MuseScore*

MuseScore is a free and multiplatform scorewriter, released under the GNU GPL. Unlike the previous programs, it's a desktop application; however, it can be conveniently used from the command line.

MuseScore uses its own file formats (*.mscz*, *.mscx*) but it can also import several other file formats; *MusicXML* is probably the most important.

We convert any supported file to PDF with this command:

```
$ musescore file.xml -o file.pdf
initScoreFonts 0x30d33c0
convert <file.xml> to <file.pdf>
setFirstInstrument: no instrument found for part 'P1'
$ _
```

5.10 Package *abc*

This package enables the inclusion of music in ABC notation in \LaTeX sources; it's similar to *lilypond-book* or *lyluatex*, but targeting ABC.

ABC provides the `abc` environment and the `\abcinput` commands. The first embeds ABC music in the source, while the second includes an external ABC file. This is (`sample-abc.tex`):

```
\documentclass{article}
\usepackage[generate,ps2eps]{abc}
\thispagestyle{empty}

\begin{document}

This is an Irish reel:

\begin{abc}[name=julia,program={abcm2ps -O=}]
X:63
T: Julia Delaney's
M: C|
L: 1/8
R: reel
K: Ddor
|: dcAG ~F2EF|~E2 DE FD D2|dcAG FGAA|Addc d2 fe :|
   f2fe fagf |ecgc acgc   |f2fe fagf|edcG Add2 :|
\end{abc}

This is an Irish polka, slightly smaller:

\abcinput[program={abcm2ps -O=},width=0.9\abcwidth]
{breeches}

End of document.

\end{document}
```

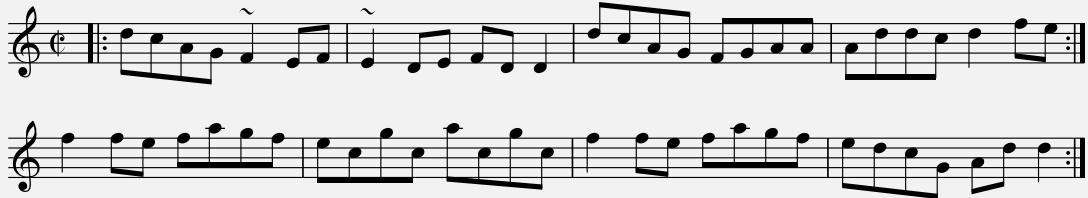
The `generate` option (default) indicates that the ABC music should be generated by the external ABC typesetter. If the ABC music is not modified and it has already been generated, we could specify the `nogenerate` option. This is what we want to do when only the text is changed.

We typeset the source with this command:

```
$ pdflatex --shell-escape sample-abc.tex
...
Output written on sample-abc.pdf (1 page, 30937 bytes).
Transcript written on sample-abc.log.
$ _
```

This is an Irish reel:

Julia Delaney's



This is an Irish polka, slightly smaller:

Breeches Full of Stitches



End of document.



6 Guitar Tablatures

Let's go back to guitar stuff. Tablature, as already explained, is a form of music notation for stringed instruments; making tablatures is basically the same process as making sheet music.

We can distinguish between *automatic tablatures*, where fret positions are generated by the program, and *manual tablatures*, where fret positions are specified by the user.

6.1 Program: LilyPond

In addition to sheet music, LilyPond can also easily typeset automatic guitar tablatures. A minimal example (`lilytab.ly`):

```
\version "2.18.2"
\header { tagline = "" } % no footer
\paper { left-margin = 0\cm }
```



```

music = {
  \time 3/4
  c4 d e f g a b a b c'2 r4 \bar "|."
}

\score {
  <<
  \new Staff { \clef "G_8" \music } % sheet music
  \new TabStaff { \tabFullNotation \music } % tablature
  >>
}

```

We typeset the score with this command:

```

$ lilypond lilytab.ly
...
Converting to `./lilytab.pdf'...
Success: compilation successfully completed
$ _

```

The image displays a musical score for a single line of music in 3/4 time. The top staff is a treble clef with a G-clef (8) and the bottom staff is a guitar tablature with a G-clef (8) and fret numbers (3, 0, 2, 3, 0, 2, 0, 2, 0, 1). The music consists of a sequence of notes: C4, D, E, F, G, A, B, A, B, C'2, followed by a rest for 4 beats. The tablature shows the corresponding fret positions for each note: 3, 0, 2, 3, 0, 2, 0, 2, 0, 1.

As we can see, the very same music line can be typeset as sheet music and as guitar tablature. Fretboard positions are automatically generated by LilyPond.

6.2 Program: [abc2xml.py](#)

This Python program converts an ABC file to a [MusicXML](#) file containing automatic tablature information; this file can then be typeset with any MusicXML-enabled application, like MuseScore.

A minimal tablature example ([abctab.abc](#)):

```

X: 1
M: 3/4
L: 1/4
K: C
%
V:1
CDE | FGA | BBB | c2z |]
V:2 clef=tab octave=-1
CDE | FGA | BBB | c2z |]

```

We typeset the score with these commands (error messages can be safely ignored):

```

$ abc2xml.py -f abctab.abc > abctab.xml
-- decoded from utf-8
-- skipped header: (field X,1)
-- done in 0.02 secs
$ musescore abctab.xml -o abctab.pdf
Jack appears to be installed on this system, so we'll use it.
initScoreFonts 0x25ba100
libpng warning: iCCP: known incorrect sRGB profile
convert <abctab.xml> to <abctab.pdf>
Error at line 18 col 16: no instrument found for part 'P1'
Error at line 138 col 16: no instrument found for part 'P2'
$ _

```

The image displays a musical score for a piece in 3/4 time. The top staff is a treble clef staff with a key signature of one sharp (F#). The melody consists of four measures: C4 (quarter), D4 (quarter), E4 (quarter), F#4 (quarter), G4 (quarter), A4 (quarter), B4 (quarter), and C5 (quarter). The bottom staff is a guitar tablature staff with three strings labeled T (Treble), A (Middle), and B (Bass). The fret numbers for each string are: T: 0, 0, 0, 1; A: 3, 0, 2, 3; B: 3, 0, 2, 3. Vertical bar lines separate the measures, and a double bar line is at the end of the fourth measure.

As we can see, the very same music line can be typeset as sheet music and as guitar tablature. Fretboard positions are automatically generated by `abc2xml.py`.

We can also use `lilypond` to typeset the ABC tablature. The ancillary program `musicxml2ly` converts MusicXML files to LilyPond format:

```

$ abc2xml.py -f abctab.abc > abctab.xml
-- decoded from utf-8
-- skipped header: (field X,1)
-- done in 0.02 secs
$ musicxml2ly abctab.xml
$ musicxml2ly: Reading MusicXML from abctab.xml ...
musicxml2ly: Converting to LilyPond expressions...
musicxml2ly: Converting to LilyPond expressions...
musicxml2ly: Output to `abctab.ly'
$ lilypond abctab.ly
GNU LilyPond 2.18.2
Processing `abctab.ly'
...
Converting to `./abctab.pdf'...
Success: compilation successfully completed
$ _

```

Regrettably, `musicxml2ly` is not as robust as MuseScore's MusicXML import filter, and it may fail on complex music.

6.3 Package *musixtex*

Starting from version 1.29, `MUSIXTEX` can also make manual tablatures for several stringed instruments. A minimal example, to be typeset with `musixtex` (`sample-musixtex-tab.tex`):

```

% bare MusiXTeX example

\input musixtex
\nopagenumbers

This is a MusiXTeX tablature example:

\setlines16           % six lines
\setstaves1{1}
\setclefsymbol1{\tabclef} % and a TAB clef
\nobarnumbers
\let\extractline\leftline % left aligned

\startextract
  \Notes \tab{5}{3} \tab{4}{0} \tab{4}{2} \en
  \bar
  \Notes \tab{4}{3} \tab{3}{0} \tab{3}{2} \en
  \bar

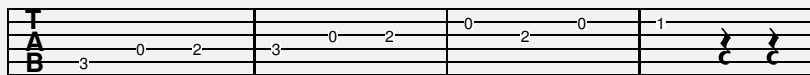
```

```

\Notes \tab{2}{0} \tab{3}{2} \tab{2}{0} \en
\bar
\Notes \tab{2}{1} \qp \qp \en
\endextract
\end

```

This is a MusiXTeX tablature example:



6.4 Package `guitartabs`

This package may be missing in less recent releases of TeX Live, so you might have to install it manually. It provides a `guitartabs` document class and a very simple syntax to manually specify strings, fret positions, and note length:

```

\documentclass{guitartabs}
\thispagestyle{empty}

\begin{document}

\Large

\begin{tabline}{4}{4}{4}{E,A,D,G,B,e}
% bar 1
% note 1 of 3, string 5, fret 4, note length 1/4
\notel{1}{3}{5}{3}{4}
% note 2 of 3, string 4, fret 0, note length 1/4
\notel{2}{3}{4}{0}{4}
\notel{3}{3}{4}{2}{4}
% bar 2
\nextbar
\notel{1}{3}{4}{3}{4}
\notel{2}{3}{3}{0}{4}
\notel{3}{3}{3}{2}{4}
% bar 3
\nextbar
\notel{1}{3}{2}{0}{4}
\notel{2}{3}{3}{2}{4}
\notel{3}{3}{2}{0}{4}
% bar 4

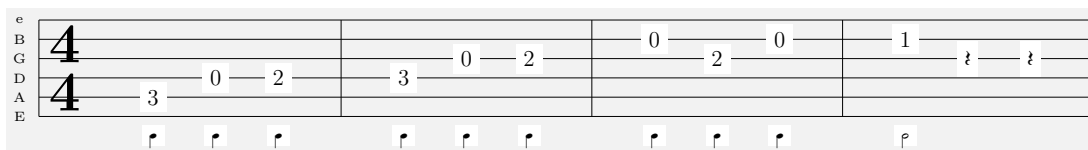
```

```

\nextbar
\note1{1}{3}{5}{3}{2}
% rests
\restquarter{2}{3}
\restquarter{3}{3}
\end{tabline}

\end{document}

```



If you don't need the note length indications at the bottom of the tablature, use `\note` instead of `\note1`.

6.5 Using Guitar Pro Tablatures

A *de facto* standard for guitar tablatures is the [Guitar Pro](#) format, or more precisely a set file formats: `.gpX` (where X is 3, 4, 5, 6), `.gpx`, `.gtp`. Many sites offer free tablatures in one of these formats.

To print them, we can use [TuxGuitar](#) or MuseScore. TuxGuitar is a free and multiplatform tablature editor; it's a desktop application that can export tablatures in PDF format. Regrettably, it cannot work as a command-line program.

To import GuitarPro tablatures into MuseScore and export them as PDF, we can use the `-P` command line switch. It exports all parts to the PDF file; then it's up to the user to find out the pages that contain the tablature.



7 Bits and Pieces

7.1 Multimedia Files

In addition to sheet music, our documents can also include sound files. To be more precise, we can click on a link to an external sound file that is referred to in the PDF document; the default player will be run.

This feature is made possible by the [hyperref](#) package, which provides the `\href` command:

```
\href{run:jingle.midi}{this midi file}
```

The first parameter opens the specified file with the default multimedia player; the second parameter creates a link to click on. For example, the above line lets the reader click on [this midi file](#) and listen to it. The operating system must know how to play the specified sound file.

The link need not be just text; we may as well apply the link to included graphics.

7.2 Notation Source Files

If we need to include notation source files, we could use a simple `verbatim` environment, or `alltt` (package `alltt`) if we also want to include L^AT_EX commands in the source:

```
\textit{\% This is a comment!}
\textbf{X:} 1
\textbf{L:} 1/4
\textbf{K:} C
\textit{\% These are notes:}
c G/G/ _AG|z !>!B!>!c z|]
```

```
% This is a comment!
X: 1
L: 1/4
K: C
% These are notes:
c G/G/ _AG|z !>!B!>!c z|]
```

But if you want something fancier, like the sources included in this document, you could also use the `tcolorbox` and define a new environment like the following:

```
\newenvironment{source}
{ % beg def
  \medskip
  \small
  \begin{margins}{-0.3cm}{-0.3cm}
    \begin{spacing}{0.9}
      \begin{tcolorbox}[breakable,boxrule=0.2pt,%
        left=0pt,right=0pt,colback=green!7!white,arc=2pt]
        \begin{alltt}

```

```
\end{margins}
}
```



8 Putting It All Together

We have solved the first problem; now we know how to make music files in several different ways. Now it's time to solve the second problem: how to combine music files and \LaTeX .

We have two possible approaches: keeping all music in the \LaTeX source, or keeping the music in external files. Both approaches have advantages and disadvantages. Basically, the first method is slower, because any change in the source may require recompilation of music excerpts. The second method requires a little more work, because we need to maintain an additional file that takes care of music conversion.

8.1 Package `abc`, Revisited

We met this package in Section 5.10, where it was used to include ABC music in a \LaTeX source. This package, however, is not limited to ABC. User-defined environments can provide support for virtually any external scorewriter; LilyPond support, however, is slightly bugged.

The following source defines the `mtx` and `pmw` environments. For each, the definition specifies the external program to run, its command line switches, and the file extension:

```
\documentclass{article}
\thispagestyle{empty}
\usepackage[generate,ps2eps]{abc}

% --- M-Tx support
\newenvironment{mtx}[1][ ]
{\renewcommand{\normalabcoutputfile}{out-mtx}%
\abc[program=musixtex,options={-g},extension=mtx,#1]}
{\endabc}
\newcommand{\mtxinput}[2][ ]{%
\abcinput[program=musixtex,options={-g},extension=mtx,#1]{#2}}

% --- PMW support
```

```

\newenvironment{pmw}[1][]
{\renewcommand{\normalabcoutputfile}{out-pmw}%
\abc[program=pmw,options={-includefont},extension=pmw,#1]}
{\endabc}
\newcommand{\pmwinput}[2][]{%
\abcinput[program=pmw,options={-includefont},extension=pmw,#1]{#2}}

\begin{document}

This document includes music excerpts written in different formats. It
uses \texttt{abc.sty} and defines new environments.

This is a short piece, typeset by M-Tx:

\mtxinput{sample-mtx}

The same piece, typeset by PMW:

\pmwinput{sample-pmw}

End of document.

\end{document}

```

This document includes music excerpts written in different formats. It uses `abc.sty` and defines new environments.

This is a short piece, typeset by M-Tx:

Music sample in M-Tx



The same piece, typeset by PMW:

Music sample in PMW



End of document.

A complete template that implements all environments is presented in Section [B.1](#).

8.2 Using **make**

Another way to make a \LaTeX document that includes music in different formats is by using a developer's tool called **make**. It's a program that takes care of what needs what, what needs to be converted first, what should be done if you modify something, and so on. **make** is normally used to compile programs.

Let's make a practical example. We have a \LaTeX document, **main.tex**, which includes three PDF files, **music1.pdf**, **music2.pdf**, and **music3.pdf**:

```
\documentclass{article}
\usepackage{graphicx}
\thispagestyle{empty}

\begin{document}

This document includes three music excerpts:

\includegraphics{music1}

\includegraphics{music2}

\includegraphics{music3}

End of document.

\end{document}
```

Let's suppose that the three PDF files are obtained from an M-Tx file, a LilyPond file, and an ABC file. We should convert the PDF files manually, then typeset **main.tex**. It's not a big deal, but what if you have dozens of music files, each of which must be converted with different commands? This task would soon grow tedious and difficult to manage.

Here **make** comes to the rescue. It uses a text file, called **Makefile**, which contains rules for building the document and the PDF files that it includes. This is a simple **Makefile** that can be used to compose (make!) **main.pdf**:

```
# Makefile for main.tex

FIGURES = music1.pdf music2.pdf music3.pdf

# The final document depends on main.tex and the figures
```

```

main.pdf: main.tex $(FIGURES)
    pdflatex main.tex

# music1.pdf depends on music1.mtx
# conversion commands follow
music1.pdf: music1.mtx
    musixtex music1.mtx ; pdfcrop musix1.pdf ; \
    /bin/mv musix1-crop.pdf music1.pdf

# music2.pdf depends on music2.ly
music2.pdf: music2.ly
    lilypond music2.ly; pdfcrop musix2.pdf ; \
    /bin/mv musix2-crop.pdf music2.pdf

# music3.pdf depends on music3.abc
music3.pdf: music3.abc
    abcm2ps -c -O= music3.abc; ps2pdf music3.ps; \
    pdfcrop musix3.pdf ; \
    /bin/mv musix3-crop.pdf music3.pdf

# end of Makefile

```

Let's see what it does. First of all, as you might have guessed the `#` character starts a comment; the rest of the line is ignored.

This line:

```
FIGURES = music1.pdf music2.pdf music3.pdf
```

creates a *variable*, that is a “name” (`FIGURES`) that “contains” the three file names `music1.pdf music2.pdf music3.pdf`. This variable will be referred to later on.

These lines:

```

main.pdf: main.tex $(FIGURES)
    pdflatex main.tex

```

declare that `main.pdf` is a *target* that depends on `main.tex` and on the three files denoted by the `FIGURES` variable. The second line specifies the command that must be run to make `main.pdf`. This line starts with a TAB character, not with spaces: this is important!

Then we have three sections, one for each music file. Each section tells `make` what to do to compose the PDF figure. For instance, the section:

```
music1.pdf: music1.mtx
    musixtex music1.mtx ; pdfcrop musix1.pdf ; \
    /bin/mv musix1-crop.pdf music1.pdf
```

states that **music1.pdf** is a target that depends on **music1.mtx**; the following lines specify the commands to create **music1.pdf**. Commands are separated by the **;** character, while **** indicates that the command continues to the next line.

To make **main.pdf**, we simply run the command:

```
$ make
...
$ _
```

in the same directory where we saved **Makefile**. **make** will make the three figures first, then **main.pdf** that depends on them.

If we modify one of the files, **make** will take care of dependencies and rebuild the final target. For example, if we modify **music2.ly**, **make** will rebuild **music2.pdf** first, then **main.pdf** that depends on **music2.pdf**.

In my opinion, this approach is the most flexible. This guide was compiled using **make** and a pretty long **Makefile**.

♪ ♪ ♪ ♪ ♪ ♪ ♪ ♪

9 The End

That's it, dear fellow musicians: I really hope that this guide will be useful to you. Please help me improve this document: for any suggestions, comments, or contributions, please feel free to contact me by [email](#). I'd like to receive feedback, especially if you use this document in education.

I would like to thank all the geniuses who wrote the packages and programs cited in this guide. Kudos and hats off, folks!

This document is copyleft © Guido Gonzato, PhD, and released under the [GNU Free Documentation Licence](#).



Ciao! =8-)



A List of Packages and Programs

This manual uses many L^AT_EX packages and includes PDF output produced by several programs, all of which are free and open source. Commercial programs were intentionally left out.

The page <https://www.ctan.org/topic/music> is an important starting point for those interested in combining L^AT_EX and music. It's a list of L^AT_EX packages for typesetting music and related stuff. The following is the complete list of packages and programs I used to make this manual, in order of appearance.

- CTAN Music page. “This topic contains packages for typesetting music and related stuff.”
<https://www.ctan.org/topic/music>.
- Package: musicography, v. 2018-05.21.
<https://ctan.org/pkg/musicography>
- Package: leadsheets, v. 0.5b
<https://ctan.org/pkg/leadsheets>
- Package: lilyglyphs, v. 0.2.3
<https://ctan.org/pkg/lilyglyphs>
- Font: Bravura, v.1.272
<https://www.smubl.org/fonts>
- Package: guitar, v. 1.6
<https://ctan.org/pkg/guitar>
- Package: gtrcrd, v. 1.1
<https://ctan.org/pkg/gtrcrd>
- Package: songs, v. 3.0
<https://ctan.org/pkg/songs>
- Package: musixguit, v. 1.2.2
<https://ctan.org/pkg/musixguit>
- Package: songbook, v. 4.5
<https://ctan.org/pkg/songbook>
- Program: Chordii, v. 4.3
<https://www.vromans.org/projects/Chordii>

- Package: gchords, v. 1.20
<https://ctan.org/pkg/gchords>
- Package: guitarchordschemes, v. 0.7
<https://ctan.org/pkg/guitarchordschemes>
- Package: guitartabs, v. 2018-05-01
<https://ctan.org/pkg/guitartabs>
- Page: MusiXTeX and Related Software.
<https://icking-music-archive.org/software/htdocs/htdocs.html>
- Package: MusiXTeX, v. 1.29
<https://ctan.org/pkg/musixtex>
- Package: M-Tx, v. 0.63a
<https://ctan.org/pkg/m-tx>
- Package: Gregoriotex, v. 5.1.1
<https://ctan.org/pkg/gregoriotex>
<http://gregorio-project.github.io/gregoriotex>
<http://gregorio-project.github.io>
- Program: LilyPond, v. 1.18.2
<http://lilypond.org>
- Package: Lyluatex, v. 1.0b
<https://ctan.org/pkg/lyluatex>
- Programs: **abcm2ps** v. 8.14.2, **and2svg** v. 1.19.0
<http://moinejf.free.fr>
- Program: PMW, Philip's Music Writer, v. 4.30
<http://people.ds.cam.ac.uk/ph10/pmw.html>
- Program: MUP v. 6.6
<http://www.arkkra.com>
- Program: MuseScore v. 3.0
<http://musescore.org>
- Package: Abc v. 2.0b
<https://ctan.org/pkg/abc>
- Program: abc2xml.ly v. 218
<https://wim.vree.org/svgParse/abc2xml.html>

B Examples

B.1 A Complete `abc` Example

This source defines environments for M-Tx, PMW, LilyPond, and MUP; it also redefines the `abc` environment as `ABC`. This is necessary for technical reasons. Lilypond sources must begin with a double line `\version "2.18.2"`; this is required to avoid a bug.

```
% typeset with:
% pdflatex -shell-escape sample-abc-all.tex

\documentclass{article}
\thispagestyle{empty}
\usepackage[generate,ps2eps]{abc}

% --- M-Tx support
\newenvironment{mtx}[1] []
{\renewcommand{\normalabcoutputfile}{out-mtx}%
\abc[program=musixtex,options={-g},extension=mtx,#1]}
{\endabc}
\newcommand{\mtxinput}[2] []{%
\abcinput[program=musixtex,options={-g},extension=mtx,#1]{#2}}

% --- PMW support
\newenvironment{pmw}[1] []
{\renewcommand{\normalabcoutputfile}{out-pmw}%
\abc[program=pmw,options={-includefont},extension=pmw,#1]}
{\endabc}
\newcommand{\pmwinput}[2] []{%
\abcinput[program=pmw,options={-includefont},extension=pmw,#1]{#2}}

% --- LilyPond support
% !!! BUG: the LilyPond source must begin with a \null command
\newenvironment{lily}[1] []
{\renewcommand{\normalabcoutputfile}{out-lily}%
\abc[program=lilypond,options={-d backend=eps},extension=ly,#1]}
{\endabc}
\newcommand{\lilyinput}[2] []{%
\abcinput[program=lilypond,options={--ps},extension=ly,#1]{#2}}

% --- MUP support
\newenvironment{mup}[1] []
{\renewcommand{\normalabcoutputfile}{out-mup}%
\abc[program=mup,options={-F},extension=mup,#1]}
{\endabc}
```

```

\newcommand{\mupinput}[2][ ]{%
\abcinput[program=mup,options={-F},extension=mup,#1]{#2}}

% --- ABC must be redefined
\newenvironment{ABC}[1][ ]
{\renewcommand{\normalabcoutputfile}{out-ABC}%
\abc[program=abcm2ps,options={-O=},extension=abc,#1]}
{\endabc}
\newcommand{\ABCinput}[2][ ]{%
\abcinput[program=abcm2ps,options={-O=},extension=abc,#1]{#2}}

\begin{document}

This document includes music excerpts written in several formats. It
uses \texttt{abc.sty} and defines new environments.

This is a short piece, typeset by M-Tx:

\begin{mtx}
Title: \bigtype Music sample in M-Tx
Style: Solo
Meter: C
Width: 160mm

c4 d8 e f g a b | c4 b8 a g f e d | c8 g+ e g c- g+ e g | c4- e c r |
\end{mtx}

The same piece, typeset by LilyPond:

\begin{lily}
% twice - it's required to avoid a bug
\version "2.18.2"
\version "2.18.2"

\header {
  title = "Music sample in LilyPond"
  tagline = "" % no footer
}

\relative c' {
  \time 4/4
  \clef treble
  c4 d8 e f8 g a b | c4 b8 a g8 f e d |
  c8 g' e g c,8 g' e g | c,4 e c r \bar "|."
}
\end{lily}

```


The same piece, typeset by PMW:

```
\begin{pmw}
Heading "|Music sample in PMW"
Key C
Time 4/4

[stave 1 treble 1]
c d- e-; f-g-a-b-; | c' b- a-; g-f-e-d-; |
c-g-e-g-; c-g-e-g-; |c e c r |
[endstave]
\end{pmw}
```

The same piece, typeset by MUP:

```
\begin{mup}
// music sample in MUP notation

header
  title "Music sample in MUP"

score
  time=4/4

music
  1: 4c; 8d bm; e ebm; f bm; g; a; b ebm;
  bar
  1: 4c+; 8b bm; a ebm; g bm; f; e; d ebm;
  bar
  1: 8c bm; g; e; g ebm; c bm; g; e; g ebm;
  bar
  1: 4c; e; c; r;
  endbar
\end{mup}
```

The same piece, typeset by *abcm2ps*:

```
\begin{ABC}
X: 1
T: Music sample in ABC
M: 4/4
L: 1/4
K: C
%
C D/E/ F/G/A/B/|c B/A/ G/F/E/D/|C/G/E/G/ C/G/E/G/|CECz|]
\end{ABC}
```

```
\end{document}
```

This document includes music excerpts written in several formats. It uses `abc.sty` and defines new environments.

This is a short piece, typeset by M-Tx:

Music sample in M-Tx



The same piece, typeset by LilyPond:

Music sample in LilyPond



The same piece, typeset by PMW:

Music sample in PMW



The same piece, typeset by MUP:

Music sample in MUP



The same piece, typeset by `abcm2ps`:

Music sample in ABC



B.2 A Complete Songbook Example

The following source is a minimal template for songbooks. It uses the `gchords` and `guitar` packages (Sections 4.1 and 3.1). The resulting PDF has its own page numbers (don't get confused!) and is included in this document with `\includepdf`.

```

\documentclass[11pt]{article}
\usepackage{graphicx}
\usepackage{gchords}
\usepackage{guitar}

\begin{document}

\title{A Minimal Songbook}
\author{Guido Gonzato}
\date{\today}

\maketitle
\tableofcontents

% -----

\section{For He's a Jolly Good Fellow}

% Typically sung to congratulate somebody.

\smallchords

\def\numfrets{4}
\begin{minipage}[c]{\linewidth} % use less space
  \chords{
    \chord{t}{n,p3,p2,n,p1,n}{C}
    \chord{t}{p3,p2,n,n,n,p3}{G}
    \chord{t}{n,p3,p2,p3,p1,n}{C7}
    \chord{t1}{n,p2,p2,p1,n,n}{F}
  }
\end{minipage}

\medskip

\includegraphics[width=\textwidth]{fellow}

\bigskip

\begin{guitar}

```

```

For [C]he's a jolly good fellow,
For [G]he's a jolly good [C]fellow,
For [C7]he's a jolly good [F]fellow,
Which [C]nobody [G]can [C]deny.

\end{guitar}

% -----

\section{Happy Birthday To You}

% Sung to remind somebody they're growing old.

\def\numfrets{4}
\begin{minipage}[c]{\linewidth}
  \chords{
    \chord{t1}{n,p2,p2,p1,n,n}{F}
    \chord{t}{n,p3,p2,n,p1,n}{C}
    \chord{t}{n,p3,p2,p3,p1,n}{C7}
    \chord{t1}{n,n,p2,p2,p2,n}{Bb}
  }
\end{minipage}

\medskip

\includegraphics[width=\textwidth]{happyb}

\bigskip

\begin{guitar}
Happy [F]birthday to [C]you,
Happy [C7]birthday to [F]you,
Happy birthday dear [Bb]Guido,
Happy [F]birthday [C]to [F]you!
\end{guitar}

% -----

\section{Warm Kitty}

% No need to introduce this lullaby!

\def\numfrets{4}
\begin{minipage}[c]{\linewidth}
\chords{
  \chord{t}{n,n,p2,p2,p2,n}{A}

```

```
\chord{t}{n,p2,p2,p1,n,n}{E}
\chord{t}{n,p2,p2,p1,p3,n}{E7}
}
\end{minipage}

\medskip

\includegraphics[width=\textwidth]{warm}

\bigskip

\begin{guitar}

[A]Soft kitty, [E]warm kitty
[A]Little ball of [E]fur [E7]~
[A]Happy kitty, [E]sleepy kitty
[A]Purr, [E]purr, [A]purr

\end{guitar}

\end{document}
```

A Minimal Songbook

Guido Gonzato

January 19, 2019

Contents

1 For He's a Jolly Good Fellow	1
2 Happy Birthday To You	2
3 Warm Kitty	2

1 For He's a Jolly Good Fellow

The image shows the musical notation for the song 'For He's a Jolly Good Fellow'. At the top, there are four guitar chord diagrams: C (x32010), G (320033), C7 (x32010), and F (133211). Below these is a two-staff musical score in 6/8 time. The first staff has a treble clef and a key signature of one flat (Bb). The melody starts on G4. The lyrics are: 'For he's a jol - ly good fel - - low, for he's a jo - ly good fel - low, for he's a jol - ly good fel - low, which no -bo -dy can de - ny!'. Chord symbols are placed above and below the notes: C, G, C7, F, C, G, C, G, C.

C
 For he's a jolly good fellow,
 G C
 For he's a jolly good fellow,
 C7 F
 For he's a jolly good fellow,
 C G C
 Which nobody can deny.

2 Happy Birthday To You

1 F C C7 Bb

Hap - py birth - day to you, hap - py birth - day to you, hap - py
 birth - day dear Gui - do, hap - py birth - day to you!

F C
 Happy birthday to you,
 C7 F
 Happy birthday to you,
 Bb
 Happy birthday dear Guido,
 F C F
 Happy birthday to you!

3 Warm Kitty

A E E7

A E E7

A E A

A E
 Soft kitty, warm kitty
 A E E7
 Little ball of fur
 A E
 Happy kitty, sleepy kitty
 A E A
 Purr, purr, purr

B.3 A sample Makefile

This is the **Makefile** that was used to typeset the previous example. The **clean** target is used to clean up all temporary files.

```
# Makefile for sample-songbook

FIGURES = fellow.pdf happyb.pdf warm.pdf

sample-songbook: sample-songbook.tex $(FIGURES)
    pdflatex sample-songbook.tex; \
    pdflatex sample-songbook.tex; \
    pdflatex sample-songbook.tex

fellow.pdf: fellow.abc
    abcm2ps -c -0= fellow.abc; \
    ps2pdf fellow.ps; pdfcrop fellow.pdf; \
    /bin/mv -f fellow-crop.pdf fellow.pdf

happyb.pdf: happyb.abc
    abcm2ps -0= happyb.abc; \
    ps2pdf happyb.ps; pdfcrop happyb.pdf; \
    /bin/mv -f happyb-crop.pdf happyb.pdf

warm.pdf: warm.abc
    abcm2ps -0= warm.abc; \
    ps2pdf warm.ps; pdfcrop warm.pdf; \
    /bin/mv -f warm-crop.pdf warm.pdf

clean:
    /bin/rm -f .*~ *~ *aux *bak *lo? *to? *out *tmp *bbl *ps
```

B.4 Verses and Guitar Chords Diagrams

Combining verses and guitar chord diagrams can be done in several ways. I think that the following example shows two of the easiest methods. Please refer to Section 3.1 and Section 4.1 for further details.

```
\documentclass{article}
\usepackage{guitar}
\usepackage{gchords}
\thispagestyle{empty}

\newcommand{\C}{\hspace{-0.8em}\chord{t}{n,p3,p2,n,p1,n}{C}}
```



```

\newcommand{\CmajVII}{\hspace{-0.8em}\chord{t}{n,p3,p2,n,n,n}{Cmaj7}}
\newcommand{\F}{\hspace{-0.8em}\chord{t1}{n,p2,p2,p1,n,n}{F}}

\def\chordsize{1.5mm}
\def\numfrets{3}
\def\namelfont{\it}

\begin{document}

\noindent
We can typeset verses and guitar chord grids in a \texttt{guitar}
environment:

\bigskip

\begin{minipage}[c]{\linewidth} % to avoid indentation
  \begin{guitar}

    \textbf{Imagine (John Lennon)}
    \emph{Intro, $\times$ 2}
    [\C] \hspace{2em} [\CmajVII] \hspace{2em} [\F] \hspace{2em}
    [\C] Imagine there's [\CmajVII|]{no} ~ [\F] heaven
    [\C] It's easy if [\CmajVII|]{you} ~ [\F] try

  \end{guitar}
\end{minipage}

\noindent
{\ldots}and so on. But we could just use the \verb|\upchord| command,
provided by \texttt{gchords}:

\begin{verse}

  \upchord{\C}Imagine there's \upchord{\CmajVII}no \quad
  \upchord{\F}heaven

  \upchord{\C}It's easy if \upchord{\CmajVII}you \quad
  \upchord{\F}try

\end{verse}

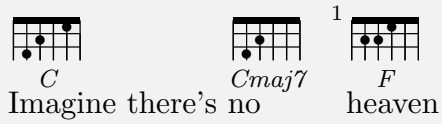
\end{document}

```

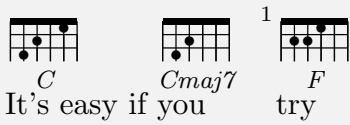
We can typeset verses and guitar chord grids in a `guitar` environment:

Imagine (John Lennon)

Intro, $\times 2$

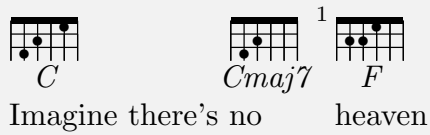


Imagine there's no heaven

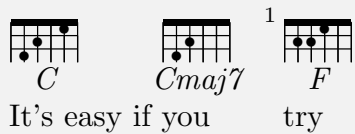


It's easy if you try

...and so on. But we could just use the `\upchord` command, provided by `gchords`:



Imagine there's no heaven



It's easy if you try