A CD Cover Class

Sebastiano Vigna

Printed January 13, 2002

1 Introduction

The purpose of this class is to print CD covers. The main design line is allowing the creation of labels with minimum effort, without restraining the freedom to customize. There is also some support for multiple cover printing.

Each CD cover is created by a number of commands which set the content of the front cover, back cover, track lists etc. After everything is ready, additional commands actually generate the covers. This is a very simple example:

```
\documentclass{cd}
\begin{document}

\covertext{
The Artist\\
\bfseries The Title
}

\leftspine{THE ARTIST}
\centerspine{THE TITLE}

\lefttracklist{
\track Song 1
\track Song 2
\track Song 3
}

\leftinfo{Words and Music by The Artist}

\makecover\par
\makeback\par
\end{document}
```

By compiling the file above, you will obtain your first CD cover. Equivalently, you can create a file `CD.dat` containing the lines between `\begin{document}` and `\makecover` and compile with LaTeX the file `CD.tex`. This is a better mechanism—each CD should have its own data (`.dat`) file, which is run through the driver file `CD.tex` or the more powerful list mechanism described below. This also allows to set some parameters one for all (for instance, the font family) in the driver file. My driver file, for instance, is as follows (see below for the non-standard commands):

```
\documentclass[a4paper]{cd}
```

```
\usepackage[latin1]{inputenc}
\usepackage{avant}
\renewcommand\rmdefault{\sfdefault}
\onecorrection{.2}
\begin{document}
\makeCD
\end{document}
```

The `CD` class loads the `article` class, so commands like `\Large` or `\smallskip` are available. However, the `CD` class provides its own precise size-switching commands, and for greater accuracy it is advisable to use LaTeX's `\\[`⟨*vspace*⟩`]` mechanism in order to generate vertical spacing.

Note that the class uses heavily the `rotating` package, so you must convert the resulting dvi file into PostScript®, or use directly `pdflatex`.

2 The Text Commands

The content of a CD cover are set using the self-explaining `\covertext`, `\backtext`, `\insidetext`, `\leftspine`, `\centerspine`, `\rightspine`, `\leftracklist`, `\leftracklist`, `\leftinfo` and `\rightinfo` commands. Note that by default the material contained in `\covertext`, `\backtext` and `\insidetext` is bottom-aligned, and the the arguments of the spine commands must not contain line breaks. The left and right track lists should use the `\track` command, which inserts a `\par` and an automatically numbered box with the track number. Should you need to set manually the track number, use `\setindex{`⟨*n*⟩`}`. The text contained in `\leftinfo` and `\rightinfo` is bottom-aligned just under the respective track lists. Note that if the right information or track list box is empty, the left one will span across the whole cover. By default everything is typeset with no justification, and no paragraph indentation. One tenth of the current baseline skip is inserted between paragraphs.

In extreme cases you may want to create different spines (e.g., for R.E.M.'s *Fables Of The Reconstruction*); the `\leftspinebis`, `\centerspinebis` and `\rightspinebis` commands allows you to insert different content into the "back" spine.

3 The Font Commands

The `CD` class provides some simple commands for switching the font dimension and line spacing. The command `\fh{`⟨*height*⟩`}` sets the font height to the given number of points (line spacing is not affected), while `\fhb{`⟨*height*⟩`}{`⟨*baselineskip*⟩`}` sets both the font height and the baseline skip (usually 6/5 of the font height will work). Note that you can just write `\fh7` in order to switch to a 7 point font, and that the `\fhb` command always sets `\parskip` to 1/10 of the current baseline skip, so `\par` will always space a little more than `\\`.

When you issue a `\newcd` command, all fonts are reset to their default values. But there are a number of self-explaining commands, i.e., `\coverfont`, `\backfont`, `\insidefont`, `\spinefont`, `\tracklistfont`, `\infofont` and `\indexfont`, that allow to change the font assigned to a part of the cover. In fact, they are just one-

argument macros whose arguments are expanded just before the corresponding text commands, and can contain other formatting parameters.

4 The Cover Creation Commands

Before setting the content of the cover, the `\newcd` command takes care of resetting everything to default values. In particular, `\backtext` is the same as `\covertext` (unless you change it explicitly), so usually you do not need to set the former. Analogously, `\backfont` is the same as `\coverfont`.

Once everything is set up, the `\makecover` and `\makeback` commands will create a cover and a back cover using the data set so far. Both command have an optional argument that can contain any of the letters `lrtbc` (left, right, top, bottom, contour). The first four letter create the respective crop marks. The last letter is mainly for "debugging" purposes, and makes evident the contour of the various parts of the cover. The default value is `lrtb`. The possibility of partially eliminating crop marks is particularly useful when stacking several covers in the same sheet.

While it is possible to create a single LaTeX document containing a CD cover, it is usually more useful to create a data file containing all `CD`-specific command, and include it from a "driver" file, containing the `\makeCD` command. With no argument, it checks for the existence of a `CD.dat` file. If it exists, it is input and then the CD cover is generated. Otherwise, the user is asked for a data file name (the `CD` class will try automatically to append the `.dat` extension to the name), which is read and processed. Of course, the optional argument can be used to specify a data file name.

Having a database of data files is particularly useful when using the `\makelist` command, which processes an entire list of CDs, printing one cover or two back covers per page; the crop marks are suitably aligned so to minimize the cutting effort. The CD list must be contained in a list file, one data file name per line. With no argument, `\makelist` checks for the existence of a `CD.lst` file. If it exists, it is input; otherwise, the user is asked for a list file name (the `CD` class will try automatically to append the `.lst` extension to the name), which is read and processed. Again, the optional argument can be used to specify a list file name.

5 The options

You can pass to the `CD` class all the options of the `article` class (e.g., paper size). Moreover, there are options `aligncovertop`, `aligninsidetop`, `alignbacktop` and `aligntop` (the last one resumes the fisrt three ones), and analogously `aligncovercenter`, etc. that allow to change the default alignment behaviour. Finally, the `alignspine` option forces vertical centering of the spine text on the "real" height of the box involved, rather than on the height of a generic upper case character. This is not usually what you want, since, e.g., accents can lead to ugly results. Experiment.

6 Getting Obsessed

PostScript fonts usually are set up in such a way that the metric of all digits is the same, regardless of the actual appearance. This (in particular with sans-serif fonts) can lead to a very ugly alignment of two-digit track numbers in which either the first or the last digit is a 1. The solution is to put in the preamble a `\onecorrection{`⟨*fraction*⟩`}` command: the positioning of two-digit numbers either starting or ending with 1 will be corrected by the given fraction of the width of a 1. For instance, `\onecorrection{.2}` works great for AvantGarde. The values for other fonts must be set by trial-and-error.