# A CD Cover Class

Sebastiano Vigna

Printed May 4, 2008

## 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 customise. There is also some support for multiple cover printing. Since version 1.1, slim CD boxes are supported.

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. Using `\makeslimcover` instead of `\makecover` and `\makeback`, you will obtain a single cover for a slim CD box.

Equivalently, you can create a file `CD.dat` containing the lines between `\begin{document}` and `\makecover` and compile with LaTeX the file `CD.tex` (or `slimCD.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`, `\lefttracklist`, `\righttracklist`, `\leftinfo` and `\rightinfo` commands (`\insidetext`, `\leftspine`, `\centerspine` and `\rightspine` are ignored for slim covers). Note that by default the material contained in `\covertext`, `\backtext` and `\insidetext` is bottom-aligned, and 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 Graphic Commands

In the case you want to fill the cover or the inside of your CD with a picture, the commands `\covergraphics` and `\insidegraphics` work like `\covertext` and `\insidetext`, but they create no border (as opposed to the standard 1 cm border for text).

# 4 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.

# 5 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 (note that, of course, this does *not* happen with \covergraphics). 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, whereas the \makeslimcover will create a slim cover. Both command have an optional argument that can contain any of the letters lrtb (left, right, top, bottom), which create the respective crop marks (note that the argument must be enclosed in brackets). The default value is lrtb. The possibility of partially eliminating crop marks is particularly useful when stacking several covers in the same sheet.

It is possible to create a single LaTeX document containing a CD cover, but 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 or the \makeslimCD command. With no argument, it checks for the existence of a *jobname*.dat file (where *jobname* is the root of the LaTeXfile under compilation—e.g., CD.dat when compiling CD.tex). If such a file exists, it is input and then the (slim) 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 (which, note again, must be enclosed in brackets) can be used to specify a data file name.

Having a database of data files is particularly useful when using the \makelist or the \makeslimlist commands, which process an entire list of CDs, printing one cover (or two back covers) per page; the crop marks are suitably aligned so to minimise the cutting effort. The CD list must be contained in a list file, one data file name per line. With no argument, \makelist and \makeslimlist check 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.

# 6 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 first three ones), and analogously `aligncovercenter`, etc. that allow to change the default alignment behaviour. The `covergraphics` option lets you use the entire cover area (instead of a centered 10 cm×10 cm square). Finally, the `alignspine` option forces vertical centring 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.

# 7 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.

# 8 The Code

First of all we manage all options. This is done with a `\newif` for `alignspine`, and by defining suitably some macros representing the alignment option for the cover, inside and back text. Default is `b`. All options we do not process are passed to the `article` class.

```
⟨*class⟩
\newif\if@lignspine
\@lignspinefalse

\DeclareOption{alignspine}{\@lignspinetrue}

\def\@ligncover{b}
\def\@ligninside{b}
\def\@lignback{b}

\DeclareOption{aligncovertop}{\def\@ligncover{t}}
\DeclareOption{aligninsidetop}{\def\@ligninside{t}}
\DeclareOption{alignbacktop}{\def\@lignback{t}}

\DeclareOption{covergraphics}{\def\@lignback{t}}

\DeclareOption{aligntop}%
    {\ExecuteOptions{aligncovertop,aligninsidetop,alignbacktop}}

\DeclareOption{aligncovercenter}{\def\@ligncover{c}}
\DeclareOption{aligninsidecenter}{\def\@ligninside{c}}
```

```
\DeclareOption{alignbackcenter}{\def\@lignback{c}}

\DeclareOption{aligncenter}%
    {\ExecuteOptions{aligncovercenter,alignsidecenter,alignbackcenter}}

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}

\ProcessOptions\relax
```

Now we load the `article` class and the `rotating` package, which is fundamental in typesetting the spine text.

```
\LoadClass{article}
\RequirePackage{rotating}
```

The `\onecorrection` command defines a the fraction used for correcting the alignment of 1's. The default is 0.

```
\DeclareRobustCommand*\onecorrection[1]{\def\onec@rrfrac{#1}}
\onecorrection{0}
```

Now we have all the font and text declaration commands. They just define a certain macro to be their argument.

```
\DeclareRobustCommand*\coverfont[1]{\def\coverf@nt{#1}}
\DeclareRobustCommand*\backfont[1]{\def\backf@nt{#1}}
\DeclareRobustCommand*\insidefont[1]{\def\insidef@nt{#1}}
\DeclareRobustCommand*\spinefont[1]{\def\spinef@nt{#1}}
\DeclareRobustCommand*\tracklistfont[1]{\def\tracklistf@nt{#1}}
\DeclareRobustCommand*\infofont[1]{\def\infof@nt{#1}}
\DeclareRobustCommand*\indexfont[1]{\def\indexf@nt{#1}}

\DeclareRobustCommand{\lefttracklist}[1]{\def\lefttr@cklist{#1}}
\DeclareRobustCommand{\righttracklist}[1]{\def\righttr@cklist{#1}}
\DeclareRobustCommand{\leftinfo}[1]{\def\leftinf@{#1}}
\DeclareRobustCommand{\rightinfo}[1]{\def\rightinf@{#1}}
\DeclareRobustCommand{\covertext}[1]{\def\c@vertext{#1}}
\DeclareRobustCommand{\backtext}[1]{\def\b@cktext{#1}}
\DeclareRobustCommand{\insidetext}[1]{\def\insid@text{#1}}

\DeclareRobustCommand{\covergraphics}[1]{\def\c@vertext{#1}\def\c@vergraphics{}}
\DeclareRobustCommand{\insidegraphics}[1]{\def\insid@text{#1}\def\insid@graphics{}}

\DeclareRobustCommand*{\leftspine}[1]{\def\leftspin@{#1}}
\DeclareRobustCommand*{\centerspine}[1]{\def\centerspin@{#1}}
\DeclareRobustCommand*{\rightspine}[1]{\def\rightspin@{#1}}
\DeclareRobustCommand*{\leftspinebis}[1]{\def\leftspin@bis{#1}}
\DeclareRobustCommand*{\centerspinebis}[1]{\def\centerspin@bis{#1}}
\DeclareRobustCommand*{\rightspinebis}[1]{\def\rightspin@bis{#1}}
```

We do not want any `lineskip`, as stacked covers should not be separated by any space. Analogously, we want no margins, no indentation and no hyphens. Offsets will be set command by each command.

```
\evensidemargin=0cm
\oddsidemargin=0cm
\topmargin=0cm
\headheight=0cm
\headsep=0cm
```

```
\footskip=0cm
\textwidth=\paperwidth
%\advance\textwidth by -3cm
\textheight=\paperheight
%\advance\textheight by -3cm

\lineskip=0pt
\lineskiplimit=0pt
\parskip=0pt
\parindent=0pt
\hyphenpenalty=10000
```

We set the unit for the `picture` environment to 1mm, and prepare a number of lengths which will be useful in aligning track numbers and spine text. `\squ@re` holds the side length of the square framing the track numbers. `\h@nging` is its hanging amount. `\@hstrip` and `\@wstrip` are used when aligning the spine. `\winf@` and `\wtr@cklist` are the width of the information and tracklist minipages.

```
\setlength{\unitlength}{1mm}
\newlength{\squ@re}
\newlength{\@temp}
\newlength{\h@nging}
\newlength{\@hstrip}
\newlength{\@wstrip}
\newlength{\winf@}
\newlength{\wtr@cklist}
\newlength{\onec@rrection}
```

The `\track` command typesets a hanging framed box with a small number inside. The number is given by a counter which is reset to 1 at each `\makeback`, and can be changed manually with the `\setindex` command. The alignment inside the small box will be corrected for numbers either starting or ending with a 1 by the fraction of the width of 1 specified with the `\onecorrection` command.

```
\newcounter{tr@ckindex}
\DeclareRobustCommand*{\setindex}[1]{\setcounter{tr@ckindex}{#1}}

\DeclareRobustCommand*{\track}{%
    \par
    \let\@firstdigit=\@empty
    \setlength{\onec@rrection}{0pt}%
    \settowidth{\@temp}{\indexf@nt1}
    \expandafter\@tfor \expandafter\@digit
      \expandafter:\expandafter=\number\value{tr@ckindex}\do {%
        \ifx\@firstdigit\@empty
            \let\@firstdigit=\@digit
        \else
            \if 1\@firstdigit
                \if 1\@digit\else
                    \setlength{\onec@rrection}{-\onec@rrfrac\@temp}%
                \fi
            \else
                \if 1\@digit
                     \setlength{\onec@rrection}{\onec@rrfrac\@temp}%
                \fi
```

```
            \fi
        \fi
    }%
    \settoheight{\@temp}{M}%
    \addtolength{\@temp}{-\squ@re}%
    \raisebox{.5\@temp}{%
        \setlength{\unitlength}{\squ@re}%
        \hspace*{-\h@nging}%
        \begin{picture}(1,1)
            \put(0,0){%
                \framebox(1,1){\hspace*{\onec@rrection}\indexf@nt\thetr@ckindex}%
            }
        \end{picture}%
    }%
    \hspace*{6pt}%
    \addtocounter{tr@ckindex}{1}%
}
```

We declare some utility commands which allow for easy font dimension switch. The `\newcd` command resets to defaults all the fonts and the text defaults.

```
\AtBeginDocument{%
    \pagestyle{empty}%
    \thispagestyle{empty}%
    \newcd
}

\DeclareRobustCommand*{\fhb}[2]{%
    \fontsize{#1pt}{#2pt}\selectfont
    \parskip=.1\baselineskip
}

\DeclareRobustCommand*{\fh}[1]{\fontsize{#1pt}{\baselineskip}\selectfont}

\DeclareRobustCommand*{\newcd}{%
    \lefttracklist{}%
    \righttracklist{}%
    \covertext{}%
    \insidetext{}%
    \leftspine{}%
    \centerspine{}%
    \rightspine{}%
    \leftspinebis{\leftspin@}%
    \centerspinebis{\centerspin@}%
    \rightspinebis{\rightspin@}%
    \leftinfo{}%
    \rightinfo{}%
    \coverfont{\fhb{16}{19}}%
    \backfont{\coverf@nt}%
    \insidefont{\fhb{10}{12}}%
    \spinefont{\fhb{9}{11}\bfseries}%
    \tracklistfont{\fhb{9}{10.5}}%
    \infofont{\fhb{7}{8.3}}%
    \indexfont{\fhb{5}{0}}%
}
```

The following two commands are useful in alignment. The first command decides the height and width of a given strip of text, to be inserted in the spine. The point is that unless the `alignspine` option has been requested, we do not set `\@hstrip`, which has been set previously to the maximum height of a capital letter. The `\alignt@baseline` command is used at the end of boxes which could be bottom aligned: it eliminates the additional height inserted when a box last line has a descendant.

```
\DeclareRobustCommand*{\@sethwstrips}[1]{%
    \settowidth{\@wstrip}{\spinef@nt #1}%
    \if@lignspine
        \settoheight{\@hstrip}{\spinef@nt #1}%
    \fi
}

\DeclareRobustCommand*{\alignt@baseline}{%
    \settodepth{\@temp}{gjpqy}%
    \vphantom{gjpqy}\par
    \vspace*{-\@temp}\par
}
```

It is now easy to write down the `\makecover` command. It is just a matter of laying out the material, and print the requested crop marks.

```
\DeclareRobustCommand*{\makecover}[1][lrtb]{%
\voffset=0in
\begin{picture}(120,240)
\end{picture}%
\begin{rotate}{90}%
\begin{picture}(240,120)
    \@tfor\cr@pmark := #1 \do {
    \if l\cr@pmark
        \put(-1,0){\line(-1,0){5}}
        \put(-1,120){\line(-1,0){5}}
    \else\if r\cr@pmark
        \put(241,0){\line(1,0){5}}
        \put(241,120){\line(1,0){5}}
    \else\if b\cr@pmark
        \put(0,-1){\line(0,-1){5}}
        \put(240,-1){\line(0,-1){5}}
        \put(120,-1){\line(0,-1){1}}
        \put(120,-3){\line(0,-1){1}}
        \put(120,-5){\line(0,-1){1}}
    \else\if t\cr@pmark
        \put(0,121){\line(0,1){5}}
        \put(240,121){\line(0,1){5}}
        \put(120,121){\line(0,1){1}}
        \put(120,123){\line(0,1){1}}
        \put(120,125){\line(0,1){1}}
    \else\if c\cr@pmark
        \put(0,0){\line(1,0){240}}
        \put(0,0){\line(0,1){120}}
        \put(120,0){\line(0,1){120}}
        \put(0,120){\line(1,0){240}}
        \put(240,0){\line(0,1){120}}
    \fi\fi\fi\fi\fi
```

```
    }

    \ifx\insid@graphics\@empty
        \put(0,0){%
            \makebox(120,120)[\@ligninside]{%
                \parbox{12cm}{%
                     \raggedright\insidef@nt\insid@text\alignt@baseline
                }%
            }%
        }
    \else
        \put(10,10){%
            \makebox(100,100)[\@ligninside]{%
                \parbox{10cm}{%
                     \raggedright\insidef@nt\insid@text\alignt@baseline
                }%
            }%
        }
    \fi
    \ifx\c@vergraphics\@empty
        \put(120,0){%
            \makebox(120,120)[\@ligncover]{%
                \parbox{12cm}{%
                     \raggedright\coverf@nt\c@vertext\alignt@baseline
                }%
            }%
        }
    \else
        \put(130,10){%
            \makebox(100,100)[\@ligncover]{%
                \parbox{10cm}{%
                     \raggedright\coverf@nt\c@vertext\alignt@baseline
                }%
            }%
        }
    \fi
\end{picture}%
\end{rotate}%
}
```

The `\makeback` command is slightly more complicated, as it must set up come values for the `\track` command to work. Moreover, it has to check for empty right information or tracklist minipages, as in this case the left ones must be enlarged, and it must try to use the text from the cover page if no back text has been specified.

```
\DeclareRobustCommand*{\makeback}[1][lrtb]{%
\voffset=-.5in
\setindex{1}%
\settowidth{\squ@re}{\indexf@nt00}%
\settoheight{\@temp}{\indexf@nt0}%
\addtolength{\squ@re}{.4\@temp}%
\setlength{\h@nging}{\squ@re}%
\addtolength{\h@nging}{6pt}%
\settoheight{\@hstrip}{\spinef@nt ABCDEFGHIJKLMNOPQRSTUVWXYZ}%
```

```
%
\ifx\righttr@cklist\@empty
    \setlength{\wtr@cklist}{12cm}%
\else
    \setlength{\wtr@cklist}{5.5cm}%
\fi
%
\ifx\rightinf@\@empty
    \setlength{\winf@}{12cm}%
\else
    \setlength{\winf@}{5.5cm}%
\fi
%
\begin{picture}(151,118)
    \@tfor\cr@pmark := #1 \do {
    \if l\cr@pmark
        \put(-1,0){\line(-1,0){5}}
        \put(-1,118){\line(-1,0){5}}
    \else\if r\cr@pmark
        \put(152,0){\line(1,0){5}}
        \put(152,118){\line(1,0){5}}
    \else\if b\cr@pmark
        \put(0,-1){\line(0,-1){5}}
        \put(151,-1){\line(0,-1){5}}
        \put(6.5,-1){\line(0,-1){1}}
        \put(6.5,-3){\line(0,-1){1}}
        \put(6.5,-5){\line(0,-1){1}}
        \put(144.5,-1){\line(0,-1){1}}
        \put(144.5,-3){\line(0,-1){1}}
        \put(144.5,-5){\line(0,-1){1}}
    \else\if t\cr@pmark
        \put(0,119){\line(0,1){5}}
        \put(151,119){\line(0,1){5}}
        \put(6.5,119){\line(0,1){1}}
        \put(6.5,121){\line(0,1){1}}
        \put(6.5,123){\line(0,1){1}}
        \put(144.5,119){\line(0,1){1}}
        \put(144.5,121){\line(0,1){1}}
        \put(144.5,123){\line(0,1){1}}
    \else\if c\cr@pmark
        \put(0,0){\line(1,0){151}}
        \put(0,0){\line(0,1){118}}
        \put(151,0){\line(0,1){118}}
        \put(0,118){\line(1,0){151}}
        \put(6.5,0){\line(0,1){118}}
        \put(144.5,0){\line(0,1){118}}
    \fi\fi\fi\fi\fi
    }

    \@sethwstrips{\leftspin@}

    \put(0,4){%
        \makebox(6.5,110)[b]{%
             \makebox[\@hstrip][r]{%
```

```
                \begin{rotate}{90}\spinef@nt\leftspin@\end{rotate}%
            }%
        }%
    }

    \@sethwstrips{\centerspin@}

    \put(0,4){%
        \makebox(6.5,110){%
            \raisebox{0pt}[\@wstrip]{\makebox[\@hstrip][r]{%
                \begin{rotate}{90}\spinef@nt\centerspin@\end{rotate}%
            }}%
        }%
    }

    \@sethwstrips{\rightspin@}

    \put(0,4){%
        \makebox(6.5,110)[t]{%
            \raisebox{0pt}[\@wstrip]{\makebox[\@hstrip][r]{%
                \begin{rotate}{90}\spinef@nt\rightspin@\end{rotate}%
            }}%
        }%
    }

    \@sethwstrips{\leftspin@bis}

    \put(144.5,4){%
        \makebox(6.5,110)[t]{%
            \makebox[\@hstrip][l]{%
                \begin{rotate}{-90}\spinef@nt\leftspin@bis\end{rotate}%
            }%
        }%
    }

    \@sethwstrips{\centerspin@bis}

    \put(144.5,4){%
        \makebox(6.5,110){%
            \raisebox{\@wstrip}[\@wstrip]{\makebox[\@hstrip][l]{%
                \begin{rotate}{-90}\spinef@nt\centerspin@bis\end{rotate}%
            }}%
        }%
    }

    \@sethwstrips{\rightspin@bis}

    \put(144.5,4){%
        \makebox(6.5,110)[b]{%
            \raisebox{\@wstrip}[\@wstrip]{\makebox[\@hstrip][l]{%
                \begin{rotate}{-90}\spinef@nt\rightspin@bis\end{rotate}%
            }}%
        }%
    }
```

```
356
357     \put(17,0){%
358         \begin{picture}(121,118)
359             \put(0,82){%
360                 \makebox(120,30)[\@lignback]{%
361                     \parbox{12.1cm}{%
362                         \raggedright\backf@nt
363                         \ifx\b@cktext\undefined
364                             \ifx\c@vergraphics\@empty\else\c@vertext\fi
365                         \else
366                             \b@cktext
367                         \fi
368                         \alignt@baseline
369                     }%
370                 }%
371             }
372
373             \put(0,5){%
374                 \makebox(55,70)[tl]{%
375                     \begin{minipage}{\wtr@cklist}%
376                         \lineskip=.5pt\lineskiplimit=1pt\raggedright
377                         \tracklistf@nt\lefttr@cklist
378                     \end{minipage}%
379                 }%
380             }
381
382             \put(65,5){%
383                 \makebox(55,70)[tl]{%
384                     \begin{minipage}{\wtr@cklist}%
385                         \lineskip=.5pt\lineskiplimit=1pt\raggedright
386                         \tracklistf@nt\righttr@cklist
387                     \end{minipage}%
388                 }%
389             }
390
391             \put(0,5){%
392                 \makebox(0,0)[bl]{%
393                     \parbox{\winf@}{%
394                         \raggedright\infof@nt\leftinf@\alignt@baseline
395                     }%
396                 }%
397             }
398
399             \put(65,5){%
400                 \makebox(0,0)[bl]{%
401                     \parbox{\winf@}{%
402                         \raggedright\infof@nt\rightinf@\alignt@baseline
403                     }%
404                 }%
405             }
406
407         \end{picture}%
408     }
409
```

```
\end{picture}%
}
```

The `\makeslimcover` command is essentially a mix of the previous two, as a single slim cover must contain the front matter and the track lists. Note that we have much less space.

```
\DeclareRobustCommand*{\makeslimcover}[1][lrtb]{%
\voffset=0in
\setindex{1}%
\settowidth{\squ@re}{\indexf@nt00}%
\settoheight{\@temp}{\indexf@nt0}%
\addtolength{\squ@re}{.4\@temp}%
\setlength{\h@nging}{\squ@re}%
\addtolength{\h@nging}{6pt}%
\settoheight{\@hstrip}{\spinef@nt ABCDEFGHIJKLMNOPQRSTUVWXYZ}%
%
\ifx\righttr@cklist\@empty
    \setlength{\wtr@cklist}{10cm}%
\else
    \setlength{\wtr@cklist}{4.7cm}%
\fi
%
\ifx\rightinf@\@empty
    \setlength{\winf@}{10cm}%
\else
    \setlength{\winf@}{4.7cm}%
\fi
%
\begin{picture}(120,240)
\end{picture}%
\begin{rotate}{90}%
\begin{picture}(240,120)
    \@tfor\cr@pmark := #1 \do {
    \if l\cr@pmark
        \put(-1,0){\line(-1,0){5}}
        \put(-1,120){\line(-1,0){5}}
    \else\if r\cr@pmark
        \put(241,0){\line(1,0){5}}
        \put(241,120){\line(1,0){5}}
    \else\if b\cr@pmark
        \put(0,-1){\line(0,-1){5}}
        \put(240,-1){\line(0,-1){5}}
        \put(120,-1){\line(0,-1){1}}
        \put(120,-3){\line(0,-1){1}}
        \put(120,-5){\line(0,-1){1}}
    \else\if t\cr@pmark
        \put(0,121){\line(0,1){5}}
        \put(240,121){\line(0,1){5}}
        \put(120,121){\line(0,1){1}}
        \put(120,123){\line(0,1){1}}
        \put(120,125){\line(0,1){1}}
    \else\if c\cr@pmark
        \put(0,0){\line(1,0){240}}
        \put(0,0){\line(0,1){120}}
```

```
        \put(120,0){\line(0,1){120}}
        \put(0,120){\line(1,0){240}}
        \put(240,0){\line(0,1){120}}
    \fi\fi\fi\fi\fi
    }

    \put(12,10){%
        \begin{picture}(100,100)
            \put(0,80){%
                \makebox(100,30)[\@lignback]{%
                    \parbox{10.1cm}{%
                        \raggedright\backf@nt
                        \ifx\b@cktext\undefined
                            \ifx\c@vergraphics\@empty\else\c@vertext\fi
                        \else
                            \b@cktext
                        \fi
                        \alignt@baseline
                    }%
                }%
            }

            \put(0,15){%
                \makebox(47,60)[tl]{%
                    \begin{minipage}{\wtr@cklist}%
                        \lineskip=.5pt\lineskiplimit=1pt\raggedright
                        \tracklistf@nt\lefttr@cklist
                    \end{minipage}%
                }%
            }

            \put(55,15){%
                \makebox(47,60)[tl]{%
                    \begin{minipage}{\wtr@cklist}%
                        \lineskip=.5pt\lineskiplimit=1pt\raggedright
                        \tracklistf@nt\righttr@cklist
                    \end{minipage}%
                }%
            }

            \put(0,0){%
                \makebox(0,0)[bl]{%
                    \parbox{\winf@}{%
                        \raggedright\infof@nt\leftinf@\alignt@baseline
                    }%
                }%
            }

            \put(55,0){%
                \makebox(0,0)[bl]{%
                    \parbox{\winf@}{%
                        \raggedright\infof@nt\rightinf@\alignt@baseline
                    }%
                }%
```

```
            }

        \end{picture}%
    }
    \ifx\c@vergraphics\@empty
        \put(120,0){%
            \makebox(120,120)[\@ligncover]{%
                \parbox{12cm}{%
                    \raggedright\coverf@nt\c@vertext\alignt@baseline
                }%
            }%
        }
    \else
        \put(130,10){%
            \makebox(100,100)[\@ligncover]{%
                \parbox{10cm}{%
                    \raggedright\coverf@nt\c@vertext\alignt@baseline
                }%
            }%
        }
    \fi
\end{picture}%
\end{rotate}%
}
```

Finally, we have the high-level commands that allow to produce one or several CD from data files, `\makeCD`, `\makelist`, `\makeslimCD` and `\makeslimlist`. All have an additional argument for the file name, defaulting to `\jobname.dat` or `\jobname.lst`.

Two separate commands factor out the checks and the user interaction in case the file is not specified or does not exist.

A data file must contain only text declaration commands from the `CD` class. All LaTeX stuff (preamble, etc.) and cover generation commands are handled automatically. A list file must contain a number of lines, each containing a data file name.

```
\DeclareRobustCommand*{\@skCDfile}[1]{%
\def\CDname{#1}%
\ifx\CDname\@empty
    \IfFileExists{\jobname.dat}{%
        \def\CDname{\jobname.dat}%
    }{%
        \typein[\CDname]{Please insert CD data file name:}%
    }%
\fi
\InputIfFileExists{\CDname.dat}{%
}{%
    \InputIfFileExists{\CDname}{%
    }{%
        \ClassError{cd}{CD data file (\CDname.dat or \CDname) not found}{}%
    }
}%
}

\DeclareRobustCommand*{\makeCD}[1][]{%
```

```
\@skCDfile{#1}\makecover\par\makeback\par
}

\DeclareRobustCommand*{\makeslimCD}[1][]{%
\@skCDfile{#1}\makeslimcover\par
}

\newread\CDlist

\newcounter{@cd}
\setcounter{@cd}{0}

\newif\ifne@f

\DeclareRobustCommand*{\@sklistfile}[1]{%
\def\CDlistname{#1}%
\ifx\CDlistname\@empty
    \IfFileExists{\jobname.lst}{%
        \def\CDlistname{\jobname.lst}%
    }{%
        \typein[\CDlistname]{Please insert CD list file name:}
    }%
\fi
\IfFileExists{\CDlistname.lst}{%
    \immediate\openin\CDlist=\CDlistname.lst
    }{%
        \IfFileExists{\CDlistname}{%
            \immediate\openin\CDlist=\CDlistname
        }{%
            \ClassError{cd}{CD list (\CDlistname.lst or \CDlistname) not found}{}%
        }
    }
\ne@ftrue
}

\DeclareRobustCommand*{\makelist}[1][]{%
\@sklistfile{#1}%
\advance\endlinechar\@M
\immediate\read\CDlist to \CDname
\advance\endlinechar-\@M
\ifeof\CDlist\ne@ffalse\fi
%
\@whilesw \ifne@f \fi {%
    \newcd
    \InputIfFileExists{\CDname.dat}{%
    }{%
        \InputIfFileExists{\CDname}{%
        }{%
            \ClassError{cd}{CD data file (\CDname.dat or \CDname) not found}{}%
        }%
    }%
    \advance\endlinechar\@M
    \immediate\read\CDlist to \CDname
    \advance\endlinechar-\@M
```

```
    \ifeof\CDlist\ne@ffalse\fi
    \ifodd\value{@cd}%
        \makeback[lrb]\par\makecover\par
    \else
        \makecover\par\ifne@f\makeback[lrt]\else\makeback\fi\par
    \fi
    \addtocounter{@cd}{1}%
}
}

\DeclareRobustCommand*{\makeslimlist}[1][]{%
\@sklistfile{#1}%
\advance\endlinechar\@M
\immediate\read\CDlist to \CDname
\advance\endlinechar-\@M
\ifeof\CDlist\ne@ffalse\fi
%
\@whilesw \ifne@f \fi {%
    \newcd
    \InputIfFileExists{\CDname.dat}{%
    }{%
        \InputIfFileExists{\CDname}{%
        }{%
            \ClassError{cd}{CD data file (\CDname.dat or \CDname) not found}{}%
        }%
    }%
    \advance\endlinechar\@M
    \immediate\read\CDlist to \CDname
    \advance\endlinechar-\@M
    \ifeof\CDlist\ne@ffalse\fi
    \makeslimcover\par
}
}

⟨/class⟩
```