Producing slides with ΛATEX 2ε

Frank Mittelbach

2022/05/18

1 Introduction

With ΛATEX 2ε it is now no longer necessary to maintain a special format for producing overhead slides. Instead the standard format may be used and internally only different font definition files come into play.

2 Usage

For producing slides you have to use slides as the document class. This class is very similar to the slides style that came with SIΛTEX, in fact it is basically a copy changed to work under ΛATEX 2ε. Thus you have to say something like

\documentclass[...]{slides}

and process this with ΛATEX 2ε.

3 Fonts

Note, that with NFSS you can easily produce slides with special fonts just by calling an appropriate style file (like times) in a \usepackage command. This works, for example, with all fonts that are defined to be scaleable (e.g., PostScript fonts) since they can be used at any size by NFSS.

However, packages like pandora won’t work because the standard .fd files shipped with NFSS only contain small sizes. You can, of course, produce additional sizes and change the .fd files accordingly so that they would be useable for slides as well.

4 Invisible text and color separation

In the original SIΛTEX it was possible to produce invisible text using the \invisible command, so that one was able to put several slides on top of each other (with each slides showing additional details, etc.). It was also possible to produce ‘color’ slides. This was done by producing individual slides one for each color and placing them on top of each other.

\footnote{Therefore you should compare the new class with old SIΛTEX styles in case you have local slide classes to see what you have to change in order to use them with ΛATEX 2ε.}
The availability of color printers and the color package make color separation obsolete, so it has been removed. Although the color has also made \invisible obsolete, the command is retained in the \LaTeX{} 2e implementation, but there are a few restrictions. Invisible fonts are implemented as special shapes where the shape names are build by prefixing the normal shape name with an uppercase I. For example, the ‘normal invisible shape’ would be In. When \LaTeX{} is requested to typeset invisible it will thus change the current shape attribute in this manner. To make this work it is necessary that the resulting font shape group is defined. If not, the normal font substitution mechanism of \LaTeX{} 2e will change the attribute until it finds a usable font shape group with the result that the text may become visible.

As long as you use the standard fonts for slides this is not a problem because all the visible font shape groups have invisible counterparts. However, if you decide on using special fonts, e.g., PostScript fonts, your \DeclareFontShape settings may not contain invisible font shape groups and thus you may be unable to use these features without adding additional \DeclareFontShape commands to your .fd files or the preamble of your document.

5 The Implementation

Warning: The implementation is still very experimental and may change internally very much. It currently basically consists of a slightly modified copy of \texttt{slides.sty} (which then forms \texttt{slides.cls}) followed by a slightly changed copy of \texttt{slitex.tex}. Documentation is practically non-existing. Everybody is invited to help changing this!

The code is divided into two parts, we first implement the class related functions and declarations and then define low level stuff that is necessary within every class. By placing such commands into a separate file it will be possible to share it with other slide classes.

5.1 The class code

At this point we input the redefinitions that are necessary for \Str\LaTeX{}.

\begin{verbatim}
\{class
\input{slides.def}
\end{verbatim}

Now we are ready for setting up the font tables. As usual, we first look for a local configuration file \texttt{sfonts.cfg}. If there isn’t one, we fall back to the default one (\texttt{sfonts.def}).

\begin{verbatim}
\{\InputIfFileExists{sfonts.cfg}{\typeout{******************************************^
\* Local config file sfonts.cfg used---J}\{\input{sfonts.def}}\}
\end{verbatim}

6 Declaration of Options

We declare a few options as illegal.
6.1 Setting Paper Sizes

The variables \texttt{\paperwidth} and \texttt{\paperheight} should reflect the physical paper size after trimming. For desk printer output this is usually the real paper size since there is no post-processing. Classes for real book production will probably add other paper sizes and additionally the production of crop marks for trimming.

\begin{verbatim}
\DeclareOption{a4paper}
{\setlength\paperheight {297mm} \setlength\paperwidth {210mm}}
\DeclareOption{a5paper}
{\setlength\paperheight {210mm} \setlength\paperwidth {148mm}}
\DeclareOption{b5paper}
{\setlength\paperheight {250mm} \setlength\paperwidth {176mm}}
\DeclareOption{letterpaper}
{\setlength\paperheight {11in} \setlength\paperwidth {8.5in}}
\DeclareOption{legalpaper}
{\setlength\paperheight {14in} \setlength\paperwidth {8.5in}}
\DeclareOption{executivepaper}
{\setlength\paperheight {10.5in} \setlength\paperwidth {7.25in}}
\end{verbatim}

The option \texttt{landscape} switches the values of \texttt{\paperheight} and \texttt{\paperwidth}, assuming the dimensions were given for portrait paper.

\begin{verbatim}
\DeclareOption{landscape}
{\setlength@tempdima {\paperheight} \setlength\paperheight {\paperwidth} \setlength\paperwidth {\@tempdima}}
\end{verbatim}

6.2 The clock option

The option \texttt{clock} prints the time at the bottom of each note. We also define here the commands and counters used to keep track of time.

\begin{verbatim}
\newif\if@clock \@clockfalse
\DeclareOption{clock}{\@clocktrue
\AtEndDocument{\typeout{@arabic\c@minutes\space minutes}}
}%
\newcounter{minutes}%
\newcounter{seconds}%
\newcommand*{\settime}[1]{\setcounter{seconds}{0}\addtime{#1}}%
\newcommand*{\addtime}[1]{\addtocounter{seconds}{#1}\setcounter{minutes}{\value{seconds}}
\global \divide \value{minutes} by 60\relax}
\end{verbatim}

6.3 Two-side or one-side printing

Two-sided printing is not allowed, so don’t declare an option. But it is necessary to initialize the switch.

\begin{verbatim}
\@twosidefalse
\end{verbatim}
6.4 Draft option

If the user requests draft we show any overfull boxes. We could probably add some more interesting stuff to this option.

\DeclareOption{draft}{\setlength\overfullrule{5pt}}
\DeclareOption{final}{\setlength\overfullrule{0pt}}

6.5 Titlepage option

The default is for a \maketitle command to make a new page.

\newif\if@titlepage
\@titlepagetrue
\DeclareOption{titlepage}{\@titlepagetrue}
\DeclareOption{notitlepage}{\@titlepagetruefalse}

6.6 Twocolumn printing

Two-column printing is again forbidden.

\DeclareOption{onecolumn}{}
\DeclareOption{twocolumn}{% \ClassWarning{slides}{No ‘twocolumn’ layout for slides}}

6.7 Equation numbering on the left

The option leqno can be used to get the equation numbers on the left side of the equation.

\DeclareOption{leqno}{\input{leqno.clo}}

6.8 Flush left displays

The option fleqn redefines the displayed math environments in such a way that they come out flush left, with an indentation of \mathindent from the prevailing left margin.

\DeclareOption{fleqn}{\input{fleqn.clo}}

7 Executing Options

Here we execute the default options to initialize certain variables.

\ExecuteOptions{letterpaper,final}

The \ProcessOptions command causes the execution of the code for every option FOO which is declared and for which the user typed the FOO option in his \documentclass command. For every option BAR he typed, which is not declared, the option is assumed to be a global option. All options will be passed as document options to any \usepackage command in the document preamble.

8 Loading Packages

The standard class files do not load additional packages.
9 Document Layout

In this section we are finally dealing with the nasty typographical details.

9.1 Fonts

\%
\def\rmdefault{lcmss} % no roman
\def\sfdefault{lcmss}
\def\ttdefault{lcmtt}
\def\itdefault{sl}
\def\sldefault{sl}
\def\bfdefault{bx}

As \texttt{fontshape} gets redefined we need to make sure that the default for \texttt{upshape} is no longer up but again n.
\def\updefault{n}

Since the number of parameters to set are very large it seems reasonable to set up one command \texttt{\@setfontsize\@parms} which will do the work for us.

\LaTeX{} offers the user commands to change the size of the font, relative to the ‘main’ size. Each relative size changing command \texttt{\size} executes the command \texttt{\@setfontsize\size\langle font-size\rangle\langle baselineskip\rangle} where:

\langle \texttt{font-size} \rangle The absolute size of the font to use from now on.
\langle \texttt{baselineskip} \rangle The normal value of \texttt{\baselineskip} for the size of the font selected. (The actual value will be \texttt{\baselinestretch \* \langle \texttt{baselineskip}\rangle}.)

A number of commands, defined in the \LaTeX{} kernel, shorten the following definitions and are used throughout. They are:

\cg{\@vpt}\ 5 \ cg{\@vipt}\ 6 \ cg{\@viipt}\ 7
\cg{\@viipt}\ 8 \ cg{\@ixpt}\ 9 \ cg{\@xpt}\ 10
\cg{\@xipt}\ 10.95 \ cg{\@xiipt}\ 12 \ cg{\@xivpt}\ 14.4

... 

\texttt{\ifourteenpt} For Sl\TeX{}, however, these are not sufficient, and we therefore need to add a few extra, larger, sizes.
\def\ifourteenpt{13.82}
\def\iseventeenpt{16.59}
\def\itwentypt{19.907}
\def\itwentyfourpt{23.89}
\def\itwentyninept{28.66}
\def\ithirtyfourpt{34.4}
\def\ifortyonept{41.28}

\texttt{\@setfontsize\@parms} This routine is used in Sl\TeX{} to interface font size setting it is modeled after the settings I found in \texttt{slides.sty}, so it probably needs an update. But any class is free to redefine it, as it is used only as an abbreviation. It’s syntax is:

\cg{\@setfontsize\@parms}\langle \texttt{lineskip}\rangle
\langle \texttt{parskip}\rangle
\langle \texttt{abovedisplayskip}\rangle

5
For NFSS1 a similar style existed which did run both with a SL\TeX with old font selection or with NFSS1. But when no separate format is made this doesn't make much sense. So the following note is history and would only be true if all NFSS stuff would be removed from the file and placed into the format.

Note: To interface the old sfonts.tex the \langle size\rangle must be hidden in commands denoting the size by its name prefixed with 'i', i.e. 20pt size is called \itwentypt at this point. The NFSS interface will define those sizes to expand to the internal size, e.g. 20 but for the old sfonts the command name, e.g. \itwentypt, will be used to construct the name \twentypt etc.

This is a crude interface to the old sfonts.tex. It will be a bit slower than the old one because it must define \@tiny etc. every time a size changes.

If classes are set up that are only for use with NFSS then the second argument may be an ordinary font size!

```
\def\@setfontsize@parms#1#2#3#4#5#6#7#8{% 
\lineskip #1\relax% 
\parskip #2\relax 
\abovedisplayskip #3\relax 
\belowdisplayskip #4\relax 
\abovedisplayshortskip #5\relax 
\belowdisplayshortskip #6\relax 
%
I don't see a reason why the \strutbox has a dim different from \baselineskip but we will leave it for the moment
\setbox\strutbox=\hbox{\vrule \@height#7\p@\@depth#8\p@\@width\z@}% 
\baselineskip\baselinestretch\baselineskip 
\normalbaselineskip\baselineskip 
}
```

Setting size relations for math scripts:

```
\DeclareMathSizes{13.82}{13.82}{10}{7}
\DeclareMathSizes{16.59}{16.59}{12}{7}
\DeclareMathSizes{19.907}{19.907}{16.59}{13.82}
\DeclareMathSizes{23.89}{23.89}{19.907}{16.59}
\DeclareMathSizes{28.66}{28.66}{23.89}{19.907}
\DeclareMathSizes{34.4}{34.4}{28.66}{23.89}
\DeclareMathSizes{41.28}{41.28}{34.4}{28.66}
```

```
\normalsize
\def\normalsize{\@setfontsize\normalsize\itwentypt{28\p@ plus3\p@ minus4\p@}%
\@setfontsize@parms}
```

```
% \setfontsize\normalsize{\itwentypt{28\p@ plus3\p@ minus4\p@} % made a bit shorter
\@setfontsize@parms{2pt} %
\{30\p@ plus18\p@ minus9\p@} %
\{15\p@ plus3\p@ minus3\p@} %
\{10\p@ plus3\p@ minus3\p@} %
```
We initially choose the normalsize font.

\small

\def\small{$@\setfontsize@small\iseteenpt{19\p@ plus3\p@ minus\p@}$
\@setfontsize@parms
{2\p@}
{15\p@ plus15\p@ minus7\p@}
{12\p@ plus3\p@ minus3\p@}
{9\p@ plus\p@ minus3\p@}
{6\p@ plus\p@}
\abovedisplayshortskip
{13.5}{5.6}$

\footnotesize
\scriptsize
\let\footnotesize=\small
\let\scriptsize=\small

\tiny
\def\tiny{$@\setfontsize@tiny\isefourteenpt{16\p@ plus2\p@ minus\p@}$
\@setfontsize@parms
{2\p@}
{14\p@ plus3\p@ minus10\p@}
{11\p@ plus3\p@ minus10\p@}
\abovedisplayskip
{8\p@ plus\p@ minus5\p@}
\{\p@ plus3\p@}
\{10\{4\}$

Actually copying the code above would be better because this would correct the error message. Maybe one should remove the first argument of \setfontsize@parms.

\large
\Large
\LARGE
\huge
\Huge
\def\large{$@\setfontsize@large\itwentyfourpt{42\p@ plus8\p@ minus5\p@}$
\@setfontsize@parms
{2\p@}
{40\p@ plus20\p@ minus4\p@}
\abovedisplayskip
\{10\p@ plus5\p@}
\abovedisplayshortskip
\{20\{8.5\}$

\def\Large{$@\setfontsize@Large\itwentyninept{48\p@ plus10\p@ minus6\p@}$
\@setfontsize@parms
{2\p@}
{48\p@ plus30\p@ minus6\p@}
\{24\p@ plus10\p@ minus6\p@}$
9.2 Paragraphing

\baselinestretch This is used as a multiplier for \baselineskip. The default is to not stretch the baselines.

\parindent \parindent is the width of the paragraph indentation.

\@lowpenalty The commands \nopagebreak and \nolinebreak put in penalties to discourage these breaks at the point they are put in. They use \@lowpenalty, \@medpenalty or \@highpenalty, dependent on their argument.

\clubpenalty These penalties are use to discourage club and widow lines. Because we use their default values we only show them here, commented out.

\displaywidowpenalty Discourage (but not so much) widows in front of a math display and forbid breaking directly in front of a display. Allow break after a display without a penalty. Again the default values are used, therefore we only show them here.
\interlinepenalty Allow the breaking of a page in the middle of a paragraph.
\brokenpenalty We allow the breaking of a page after a hyphenated line.

9.3 Page Layout

All margin dimensions are measured from a point one inch from the top and lefthand side of the page.

9.3.1 Vertical spacing

\headheight The \headheight is the height of the box that will contain the running head. The \headsep is the distance between the bottom of the running head and the top of the text. \topskip is the \baselineskip for the first line on a page.

\headheight{14\p@} \headsep{15\p@} \topskip{30\p@}

\footskip The distance from the baseline of the box which contains the running footer to the baseline of last line of text is controlled by the \footskip. Bottom of page:
\footskip{25\p@}

\maxdepth The \TeX{} primitive register \maxdepth has a function that is similar to that of \topskip. The register \maxdepth should always contain a copy of \maxdepth. In both plain \TeX{} and \LaTeX{} 2.09 \maxdepth had a fixed value of 4pt; in native \LaTeX{}2e mode we let the value depend on the typesize. We set it so that \maxdepth + \topskip = \textsize \times 1.5. As it happens, in these classes \topskip is equal to the typesize, therfore we set \maxdepth to half the value of \topskip.
\maxdepth{25\p@}

9.3.2 The dimension of text

\textwidth When we are in compatibility mode we have to make sure that the dimensions of the printed area are not different from what the user was used to see.
\textwidth{640\p@}

When we are not in compatibility mode we can set some of the dimensions differently, taking into account the paper size for instance.
First, we calculate the maximum textwidth, which depends on the papersize. Then we calculate the approximate length of 65 characters, which should be the maximum length of a line of text. The calculated values are stored in \@tempdima and \@tempdimb.

\setlength\@tempdima{\paperwidth}
\addtolength\@tempdima{-2in}
\setbox\@tempboxa\hbox{\rmfamily im}
\setlength\@tempdimb{.5\wd\@tempboxa}
\setlength\@tempdimb{65\@tempdimb}

Now we can set the \textwidth, depending on whether we will be setting one or two columns.

The text should not be wider than the minimum of the paperwidth (minus 2 inches for the margins) and the maximum length of a line as defined by the number of characters.

\ifdim\@tempdima>\@tempdimb\relax
\setlength\textwidth{\@tempdimb}
\else
\setlength\textwidth{\@tempdima}
\fi
\fi

Here we modify the width of the text a little to be a whole number of points.
\@settopoint\textwidth
\columnwidth
\columnsep
\columnseprule
\textwidth
\columnwidth \textwidth
\columnsep 10pt
\columnseprule \z@\textheight

Now that we have computed the width of the text, we have to take care of the height. The \texttheight is the height of text (including footnotes and figures, excluding running head and foot).

First make sure that the compatibility mode gets the same dimensions as we had with \LaTEX2.09. The number of lines was calculated as the floor of the old \texttheight minus \topskip, divided by \baselineskip for \normalsize. The old value of \texttheight was 528pt.

\if@compatibility
\setlength\texttheight{600\p@}
\else
\setlength\texttheight{\textwidth\textheight{600\p@}}
\fi

Again we compute this, depending on the papersize and depending on the baselineskip that is used, in order to have a whole number of lines on the page.
\columnwidth
\columnsep
\texttheight
\columnwidth \texttheight
\columnsep 10pt
\columnseprule \z@\textheight
\texttheight

\setlength\@tempdima{\paperheight}
We leave at least a 1 inch margin on the top and the bottom of the page.
\addtolength\@tempdima{-2in}

We also have to leave room for the running headers and footers.
\addtolength\@tempdima{-1in}

Then we divide the result by the current \baselineskip and store this in the count register \@tempcnta, which then contains the number of lines that fit on this page.
\texttt{divide}@tempdima\baselineskip
\@tempcnta=@tempdima
From this we can calculate the height of the text.
\setlength{textheight}{@tempcnta\baselineskip}
\fi
The first line on the page has a height of \topskip.
\advance{textheight} by \topskip

\subsection{Margins}
\oddsidemargin
\evensidemargin
\marginparwidth
First we give the values for the compatibility mode.

\if@compatibility
\setlength\oddsidemargin {17\p@}
\setlength\evensidemargin {17\p@}
\setlength\marginparwidth {20\p@}
\else
When we are not in compatibility mode we can take the dimensions of the
selected paper into account.

We center the text on the page, by calculating the difference between
\textwidth\ and \paperwidth$-2\text{in}$. Half of that difference is then used for the
margin. The amount of space that can be used for marginal notes is at least
0.8 inch, to which we add any ‘leftover’ space.
\setlength{@tempdima}{\paperwidth}
\addtolength{@tempdima}{-2\textwidth}
\addtolength{@tempdima}{-\textwidth}
\setlength\oddsidemargin{.5{@tempdima}}
\setlength\marginparwidth{.8in}
\addtolength\marginparwidth{.5{@tempdima}}
The \evensidemargin\ can now be computed from the values set above.
\setlength\evensidemargin{\paperwidth}
\addtolength\evensidemargin{-2\textwidth}
\addtolength\evensidemargin{-\textwidth}
\addtolength\evensidemargin{-\oddsidemargin}
\fi

\marginparsep
\marginparpush
The horizontal space between the main text and marginal notes is determined by
\marginparsep, the minimum vertical separation between two marginal notes is
controlled by \marginparpush.
\setlength\marginparsep{5\p@}
\setlength\marginparpush{5\p@}

\topmargin
The \topmargin\ is the distance between the top of ‘the printable area’ –which
is 1 inch below the top of the paper– and the top of the box which contains the
running head.

It can now be computed from the values set above.
\if@compatibility
\setlength\topmargin{-10pt}
\else
\setlength\topmargin{\paperheight

11
By changing the factor in the next line the complete page can be shifted vertically.

9.3.4 Footnotes

\footnotesep is the height of the strut placed at the beginning of every footnote. It equals the height of a normal \footnotesize strut in this class, thus no extra space occurs between footnotes.

\setlength{\footnotesep}{20\p@}

\skip\footins is the space between the last line of the main text and the top of the first footnote.

\setlength{\skip\footins}{10\p@ \@plus 2\p@ \@minus 4\p@}

9.4 Page Styles

The page style \texttt{foo} is defined by defining the command \texttt{\ps@foo}. This command should make only local definitions. There should be no stray spaces in the definition, since they could lead to mysterious extra spaces in the output (well, that’s something that should be always avoided).

\ps@headings The \texttt{\ps@...} command defines the macros \texttt{\@oddhead}, \texttt{\@oddfoot}, \texttt{\@evenhead}, \texttt{\@oddhead} and \texttt{\@evenfoot} to define the running heads and feet—e.g., \texttt{\@oddhead} is the \texttt{\@evenfoot} called inside an \texttt{\hbox} of width \texttt{textwidth}.

The page styles of slides is determined by the ‘slide’ page style, the slide environment executing a \texttt{\thispagestyle{slide}} command. The page styles of overlays and notes are similarly determined by ‘overlay’ and ‘note’ page styles. The command standard ‘headings’, ‘plain’ and ‘empty’ page styles work by redefining the ‘slide’, ‘overlay’, and ‘note’ styles.

\ps@headings
9.5 Providing math versions

\LaTeX{} provides two versions. We call them normal and bold, respectively. \SL\TeX{} does not have a bold version. But we treat the invisible characters as a version. The only thing we have to take care of is to ensure that we have exactly the same fonts in both versions available.

\DeclareMathVersion{invisible}

Now we define the basic math groups used by \LaTeX{}. Later on, in packages some other math groups, e.g., the AMS symbol fonts, will be defined.

As a default I used serif fonts for mathgroup 0 to get things like $\log$ look right.

\SetSymbolFont{operators}{normal}\{OT1\}{lcmsa}\{m\}{n}
\SetSymbolFont{letters}{normal}\{OML\}{lcmm}\{m\}{it}
\SetSymbolFont{symbols}{normal}\{OMS\}{lcmsy}\{m\}{n}
\SetSymbolFont{largesymbols}{normal}\{OMX\}{lcmea}\{m\}{n}

\SetSymbolFont{operators}{invisible}\{OT1\}{lcmsa}\{m\}{In}
\SetSymbolFont{letters}{invisible}\{OML\}{lcmm}\{m\}{Iit}
\SetSymbolFont{symbols}{invisible}\{OMS\}{lcmsy}\{m\}{In}
\SetSymbolFont{largesymbols}{invisible}\{OMX\}{lcmea}\{m\}{In}

\def\@mainsize{\visible\tiny}
9.6 Environments

**titlepage (env.)** This environment starts a new page, with pagestyle *empty* and sets the page counter to 0.

```latex
\newenvironment{titlepage}
{\newpage
 \thispagestyle{empty}\
 \setcounter{page}{\z@}}
{\newpage}
```

9.6.1 General List Parameters

The following commands are used to set the default values for the list environment’s parameters. See the \LaTeX{} manual for an explanation of the meaning of the parameters.

\begin{verbatim}
 \leftmargini
 \leftmarginii
 \leftmarginiii
 \leftmarginiv
 \leftmarginv
 \leftmarginvi

 \setlength\leftmargini {38\p@}
 \setlength\leftmarginii {30\p@}
 \setlength\leftmarginiii {20\p@}
 \setlength\leftmarginiv {15\p@}
 \setlength\leftmarginv {15\p@}
 \setlength\leftmarginvi {10\p@}

 \@listi
 These commands set the values of \leftmargin, \parskip, \topsep, and \itemsep
 \@listii, \@listiii, \@listi, i.e. for a level one list, as a list environment may appear inside a\n \@listiv trivlist, for example inside a theorem environment.
 \@listv
 \def\@listi{\leftmargini}\leftmargini
 \def\@listii{\leftmarginii}\leftmarginii
 \def\@listiii{\leftmarginiii}\leftmarginiii
 \def\@listiv{\leftmarginiv}\leftmarginiv
 \def\@listv{\leftmarginv}\leftmarginv
 \def\@listvi{\leftmarginvi}\leftmarginvi

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par

 \parsep .5\parskip
 \topsep \parsep
 \itemsep \parskip
 \partopsep \z@}
 \par
\end{verbatim}
Here we initialize \leftmargin and \labelwidth.

\leftmargin\leftmargini
\labelwidth\leftmargini\advance\labelwidth-\labelsep

9.6.2 Paragraph-formatting environments

verse (env.) Inside a verse environment, \ ends a line, and line continuations are indented further. A blank line makes new paragraph with \parskip space.

\newenvironment{verse}{\let\=\@centercr
\list{}{\itemsep \z@
\itemindent -15\p@\n\rightmargin \leftmargin
\advance\leftmargin 15\p@}\
\item[]\}{\endlist}

quotation (env.) The quotation environment fills lines, indents paragraphs.

\newenvironment{quotation}{\list{}{\listparindent 20\p@
\itemindent\listparindent
\rightmargin\leftmargin\item[]}\}{\endlist}

quote (env.) The quote environment is the same as the quotation environment, except that there is no paragraph indentation.

\newenvironment{quote}{\list{}{\rightmargin\leftmargin}\item[]}\}{\endlist}

9.6.3 List-making environments

description (env.) The description environment is defined here – while the itemize and enumerate environments are defined in the LaTeX format.

\newenvironment{description}{\list{}{\labelwidth\z@\n\itemindent-\leftmargin\n\let\makelabel\descriptionlabel}}\}{\endlist}

\descriptionlabel To change the formatting of the label, you must redefine \descriptionlabel.

\newcommand*{\descriptionlabel}[1]{\hspace\labelsep\normalfont\bfseries #1}

9.6.4 Enumerate

The enumerate environment uses four counters: \theenumi, \theenumii, \theenumiii and \theenumiv, where \theenumN controls the numbering of the Nth level enumeration.

\theenumi The counters are already defined in the \LaTeX format, but their representation is changed here.

\newcommand{\theenumi}{\@arabic{\c@enumi}}
\newcommand{\theenumii}{\@alph{\c@enumii}}
\newcommand{\theenumiii}{\@roman{\c@enumiii}}
\newcommand{\theenumiv}{\@alph{\c@enumiv}}
The label for each item is generated by the four commands \labelenumi ... \labelenumiv.

\labelenumi The expansion of \p@enumi \theenumi defines the output of a \ref command when referencing an item of the Nth level of an enumerated list.

\labelenumii
\p@enumii
\newcommand\p@enumii{\theenumi}
\newcommand\labelenumii{\theenumii}
\newcommand\p@enumiii{\theenumi(\theenumii)}
\newcommand\labelenumiii{\theenumiii}
\newcommand\p@enumiv{\p@enumiii\theenumiii}
\newcommand\labelenumiv{\theenumiv}

9.6.5 Itemize
\labelitemi Itemization is controlled by four commands: \labelitemi, \labelitemii, \labelitemiii, and \labelitemiv, which define the labels of the various itemization levels.

\labelitemi
\newcommand\labelitemi{$\m@th\bullet$}
\labelitemii
\newcommand\labelitemii{\normalfont\bfseries \textendash}
\labelitemiii
\newcommand\labelitemiii{$\m@th\ast$}
\labelitemiv
\newcommand\labelitemiv{$\m@th\cdot$}

9.7 Setting parameters for existing environments
9.7.1 Array and tabular
\arraycolsep The columns in an array environment are separated by 2\arraycolsep. Array and tabular environment parameters
\setlength\arraycolsep{8\p@}

\tabcolsep The columns in a tabular environment are separated by 2\tabcolsep.
\setlength\tabcolsep{10\p@}

\arrayrulewidth The width of rules in the array and tabular environments is given by the length parameter\arrayrulewidth.
\setlength\arrayrulewidth{.6\p@}

\doublerulesep The space between adjacent rules in the array and tabular environments is given by \doublerulesep.
\setlength\doublerulesep{3\p@}

9.7.2 Tabbing
\tabbingsep This controls the space that the \' command puts in. (See \LaTeX manual for an explanation.)
\setlength\tabbingsep{10pt}
\setlength\tabbingsep{\labelsep}
9.7.3 Minipage

The macro \@minipagerestore is called upon entry to a minipage environment to set up things that are to be handled differently inside a minipage environment. In the current styles, it does nothing.

\@mpfootins Minipages have their own footnotes; \skip\@mpfootins plays same rôle for footnotes in a minipage as \skip\footins does for ordinary footnotes.
425 \skip\@mpfootins = \skip\footins

9.7.4 Framed boxes

\fboxsep The space left by \fbox and \framebox between the box and the text in it.
\fboxrule The width of the rules in the box made by \fbox and \framebox.
426 \setlength{\fboxsep}{5pt}
427 \setlength{\fboxrule}{.6pt}
\theequation The equation number will be typeset as arabic numerals.
428 \def\theequation{\arabic{equation}}

\jot \jot is the extra space added between lines of an eqnarray environment. The default value is used.
429 \setlength{\jot}{3pt}
\@eqnnum The macro \@eqnnum defines how equation numbers are to appear in equations. Again the default is used.
430 \def\@eqnnum{\theequation}

9.8 Font changing

Here we supply the declarative font changing commands that were common in \LaTeX version 2.09 and earlier. These commands work in text mode and in math mode. They are provided for compatibility, but one should start using the \text... and \math... commands instead. These commands are redefined using \DeclareOldFontCommand, a command with three arguments; the user command to be defined, \LaTeX commands to execute in text mode and \LaTeX commands to execute in math mode.
\rm The commands to change the family. When in compatibility mode we select the \tt ‘default’ font first, to get \LaTeX2.09 behaviour.
431 \DeclareOldFontCommand{\rm}{\normalfont}{\mathrm}
432 \DeclareOldFontCommand{\sf}{\normalfont}{\mathsf}
433 \DeclareOldFontCommand{\tt}{\normalfont}{\mathtt}

\bf The command to change to the bold series. One should use \mdseries to explicitly switch back to medium series.
434 \DeclareOldFontCommand{\bf}{\normalfont}{\bfseries}{\mathbf}
\sl And the commands to change the shape of the font. The slanted and small caps
\it shapes are not available by default as math alphabets, so those changes do nothing
\sc in math mode. One should use \upshape to explicitly change back to the upright
shape.

435 \DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
436 \DeclareOldFontCommand{\sl}{\normalfont\slshape}{\relax}
437 \DeclareOldFontCommand{\sc}{\normalfont\scshape}{\relax}

\cal The commands \cal and \mit should only be used in math mode, outside math
\mit mode they have no effect. Currently the New Font Selection Scheme defines these
commands to generate warning messages. Therefore we have to define them ‘by
hand’.

438 \DeclareRobustCommand{\cal}{\@fontswitch\relax\mathcal}
439 \DeclareRobustCommand{\mit}{\@fontswitch\relax\mathnormal}

9.9 Footnotes

\footnoterule Usually, footnotes are separated from the main body of the text by a small rule.
This rule is drawn by the macro \footnoterule. We have to make sure that the
rule takes no vertical space (see plain.tex). The resulting rule will appear on all
color layers, so it’s best not to draw a rule.

440 \renewcommand\footnoterule{}
441 % \let \footnoterule = \relax

% \newcounter{footnote}
442 \def\thefootnote{\fnsymbol{footnote}}
443 \@addtoreset{footnote}{slide}
444 \@addtoreset{footnote}{overlay}
445 \@addtoreset{footnote}{note}

% \makefntext The footnote mechanism of \LaTeX \ calls the macro \makefntext to produce the
actual footnote. The macro gets the text of the footnote as its argument and should use
\makefntext \hspace to produce the mark of the footnote. The macro \makefntext is called when effectively inside a \parbox of width \columnwidth (i.e., with
\hsize = \columnwidth).

An example of what can be achieved is given by the following piece of \TeX code.

\long\def\makefntext#1{\%\setpar{\@par \@tempdima = \hsize \@tempdima-10pt \parshape \one 10pt \@tempdima}\% \parindent 1em\noindent \hbox to \z@{\hss@\makefntext#1}}

The effect of this definition is that all lines of the footnote are indented by 10pt,
while the first line of a new paragraph is indented by 1em. To change these
dimensions, just substitute the desired value for ‘10pt’ (in both places) or ‘1em’.
The mark is flushright against the footnote.

In these document classes we use a simpler macro, in which the footnote text
is set like an ordinary text paragraph, with no indentation except on the first line
of a paragraph, and the first line of the footnote. Thus, all the macro must do is
set \parindent to the appropriate value for succeeding paragraphs and put the
proper indentation before the mark.

\long\def\@makefntext#1{
\noindent
\hangindent 10\p@
\hb@xt@10\p@{\hss\@makefnmark}#1}

\@makefnmark The footnote markers that are printed in the text to point to the footnotes should
be produced by the macro \@makefnmark. We use the default definition for it.

9.10 The title

The commands \title, \author, and \date are already defined, so here we just
define \maketitle.

\newcommand\maketitle{{\centering \Large \@title \par}\
\@author \par \@date\par}\
\if@titlepage \break \fi}

10 Initialization

10.1 Date

\today This macro uses the \TeX primitives \month, \day and \year to provide the date
of the \TeX-run.

\newcommand\today{\ifcase\month\or
January\or February\or March\or April\or May\or June\or
July\or August\or September\or October\or November\or December\fi
\space\number\day, \number\year}

Default initializations

\pagenumbering{arabic}
\onecolumn
\(/\text{class}\)

10.2 Basic code

The code below is basically a copy of \slitex.tex with some changes.

Global changes so far:

10.2.1 Hacks for slide macros

\message{hacks,}
\outer\def\newifG#1{\count\escapechar \escapechar\m@ne
\expandafter\expandafter\expandafter\edef\@ifG#1{true}{\global\let\noexpand#1
\noexpand\iftrue}%
\expandafter\expandafter\expandafter\edef\@ifG#1{false}{\global\let\noexpand#1
\noexpand\iffalse}%
\@ifG#1{false}\escapechar\count0} % the condition starts out false
\def\@ifG#1#2{\csname\expandafter\ifG@\string#1#2\endcsname}
\uccode'1='i \uccode'2='f \uccode'3='G \uppercase{\gdef\ifG@123{G}}
% 'ifG' is required
\def\@gobbletoend#1{\def\@argend{#1}\@ggobtoend}
\long\def\@ggobtoend#1\end#2{\fi\def\reserved@a{#2}\
\ifx\reserved@a\@argend\else\@ggobtoend\fi}

FMi: I don’t see any reason for this command since \fi is hidden anyway in the
replacement text \def\@xfi{\fi}
\message{slides,}

10.2.2 Slide macros

Switches:
@bw true if making black and white slides
@visible true if visible output to be produced.
@makingslides true if making a slide/overlay/note

\newif\if@bw
\newif\if@visible
\newif\if@onlyslidesw \onlyslideswfalse
\newif\if@onlynotesw \onlynoteswfalse
\newif\if@makingslides

FMi: \newifG replaces \gdef\@slidesw{T} stuff
\newifG\if@slidesw

Counters
slide slide number
overlay overlay number for a slide
note note number for a slide
\countdef\c@slide=0 \c@slide=0
\def\cl@slide{}
\countdef\c@overlay=1 \c@overlay=0
\def\cl@overlay{}
\countdef\c@note=2 \c@note=0
\def\cl@note{}

Add these counters explicitly to the ‘ckpt list’ so that the \include mechanism
works.
\g@addto@macro\cl@ckpt{\@elt{slide}\@elt{overlay}\@elt{note}}
\@addtoreset{overlay}{slide}
\@addtoreset{note}{slide}

Redefine page counter to some other number. The page counter will always be
zero except when putting out an extra page for a slide, note or overlay.
\@definecounter{page}
\@addtoreset{page}{slide}
\@addtoreset{page}{note}
\@setlimits \LIST \LOW \HIGH

Assumes that \LIST = RANGE1,RANGE2,...,RANGEn (n>0)
Where RANGEi = j or j-k.

Then \@setlimits globally sets
(i) \LIST := RANGE2, ..., RANGEn
(ii) \LOW := p
(iii) \HIGH := q
where either RANGE1 = p-q or RANGE1 = p and q=p.

\def\@sl@getargs#1-#2-#3\relax#4#5\xdef#4{#1}\xdef#5{#2}
\def\@sl@ccdr#1,#2\relax#3#4\xdef#3{#1-#1-}\xdef#4{#2}
\def\@setlimits #1#2#3\expandafter\@sl@ccdr#1\relax\@sl@gtmp #1%
\expandafter\@sl@getargs\@sl@gtmp\relax#2#3

\onlyslides{LIST} ::= BEGIN
  @onlyslidesw := true
  \@doglslidelist :=G LIST,999999,999999
  if @onlynotesw = true
    else @onlynotesw := true
        \@doglnotelist :=G LIST,999999,999999
  fi
  message: Only Slides LIST
END

\onlynotes{LIST} ::= BEGIN
  @onlynotesw := true
  \@doglnotelist :=G LIST,999999,999999
  if @onlyslidesw = true
    else @onlyslidesw := true
        \@doglslidelist{999999,999999}
  fi
  message: Only Notes LIST
END

22
\setupcounters ::= (similar to old \blackandwhite #1 ::= )
\newpage
page counter := 0
@bw := T
@visible := T
if @onlyslidesw = true
  then \doslidelist := \doglslidelist
      \setlimits\doslidelist\doslidelow\doslidehigh
fi
if @onlynotesw = true
  then \donotelist := \doglnotelist
      \setlimits\donotelist\donotelow\donotehigh
fi
\normalsize % Note, this sets font to \rmfamily , which sets % \@currfont to \rmfamily
counter slidenumber := 0
counter note := 0
counter overlay := 0
@makingslides := F % \blackandwhite: @makingslides := T
%% input #1
%% \makingslides := F
518 \if@compatibility
519 % In compatibility mode, need to define \verb+\blackandwhite+, \verb+\colors+, \verb+\colorslides+, etc.
520 \def\blackandwhite#1{\newpage\setcounter{page}{0}\@bwtrue\@visibletrue
521 \if@onlyslidesw \xdef\doslidelist{\doglslidelist}\fi
522 \if@onlynotesw \xdef\donotelist{\doglnotelist}\fi
523 \normalsize\setcounter{slide}{0}\setcounter{overlay}{0}\@makingslidestrue\input #1\@makingslidesfalse}
524 \colors{COLORS} ::= for \@colortemp := COLORS
doi \csname \@colortemp \endcsname == \@color{\@colortemp} od
525 if \@colorlist = empty
526 \xdef\@colorlist{COLORS}
527 \else \xdef\@colorlist{\@colorlist,COLORS}\fi
528 \def\colors#1{\for \@colortemp=#1\do{\expandafter
529 \xdef\csname\@colortemp\endcsname{\noexpand\color{\@colortemp}}}}\ifx
530 \@colorlist\empty \def\@colorlist{1}\fi
531 \else \def\@colorlist{\@colorlist,1}\fi
532 \def\@colorlist{ }
533 \colorslides{FILE} ::= 
\newpage
page counter := 0
@bw := F
for \@currcolor := \@colorlist
do @visible := T
  if @onlyslidesw = true
23
counter slide := 0
counter overlay := 0
counter note := 0
type message
 generate color layer output page
\makingslides := T
input #1
\makingslides := F
od
\def\colorslides#1\{newpage\setcounter{page}{0}\@bwfalse
\@for\@currcolor:=\@colorlist\do
\@visibletrue
\if@onlyslidesw \xdef\@doslidelist{\@doglslidelist}\
@setlimits\@doslidelist\@doslidelow\@doslidehigh\fi
\if@onlynotesw \xdef\@donotelist{\@doglnotelist}\
@setlimits\@donotelist\@donotelow\@donotehigh\fi
\normalsize\setcounter{slide}{0}\setcounter{overlay}{0}\
\setcounter{note}{0}\typeout{color \@currcolor}\
\newpage
\begin{huge}\
\begin{center}\
COLOR LAYER\[.75in]\%
\@currcolor
\end{center}\
\end{huge}\
\newpage
\@makingslidestrue
\input #1
\@makingslidesfalse}\
\else %% if@compatibility
\def\setupcounters\{newpage\setcounter{page}{0}\@btrue\@visibletrue
\if@onlyslidesw \xdef\@doslidelist{\@doglslidelist}\
@setlimits\@doslidelist\@doslidelow\@doslidehigh\fi
\if@onlynotesw \xdef\@donotelist{\@doglnotelist}\
@setlimits\@donotelist\@donotelow\@donotehigh\fi
\normalsize\setcounter{slide}{0}\setcounter{overlay}{0}\
\setcounter{note}{0}\@makingslidesfalse}
\AtBeginDocument\{setupcounters\}
\fi %% if@compatibility
\slide COLORS ::=
BEGIN\changes{v2.3}{1994/03/16}{Moved \cs{newpage} up front, here and in
\cs{note} and \cs{overlay}}\par\break\stepcounter{slide}\setcounter{page}{0} % in case of non-slide pages
\@slidesw := G T
if @onlyslidesw = true % set \@slidesw = T iff
then % page to be output
while \c@slide > \@doslidehigh
  do \@setlimits\@doslidelist\@doslide\@doslidehigh od
if \c@slide < \@doslide\@doslide\@doslidehigh
  then \@slidesw := F
  fi
fi
if \@slidesw = T
then \@slidesw := G F
\begingroup
  if @bw = true
    then \@slidesw := G T
  else \@color{COLORS}
    \if@visible then \@slidesw := G T \fi
  fi
\endgroup
fi
if \@slidesw = T
then \@making\@slides := T
else \end{slide}
\@gobbletoend{slide}
fi
\endslide ::= 
BEGIN
  \par\break
END
\if@compatibility
567 \def\slide#1{\stepcounter{slide}\@slidestrue\if@onlyslidesw
568 \@whilenum \c@slide > \@doslidehigh\relax
569 \do{\@setlimits\@doslidelist\@doslide\@doslidehigh\@doslidehigh}
570 \ifnum \c@slide < \@doslide\@doslide\@doslidehigh\relax\@slidesfalse\fi\fi
572 \if@slidesw
573 \@slidesfalse
574 \if@compatibility
575 \percent FMi this is only a hack at the moment to get things running.
576 \percent \begingroup
577 \if@bw\@slidestrue\else
577 \@color{#1}\if@visible \@slidestrue \fi
578 \fi
579 \endgroup
580 \fi
581 \if@slidesw \newpage\thispagestyle{slide}\percent
This will set up the last color specified in the argument to \slide as the current color. If only back and white slides are prepared \last@color will be empty and effectively \relax will be generated (hopefully).
We need to reset to a default font at the beginning of a slide. (not done yet).

\csname \last@color \endcsname
\else\end{slide}\gobbletoend{slide}\fi}
% \else
% if@compatibility
% \def\slide{\par\break
\stepcounter{slide}\setcounter{page}{0}\G@slideswtrue\if@onlyslidesw
\@whilenum \c@slide >\@doslidehigh\relax
\do{\@setlimits\@dosolidelist\@doslidelow\@doslidehigh}\ifnum
\c@slide <\@doslidelow\relax\G@slideswfalse\fi\fi
\if\@slidesw
\G@slideswfalse
% FMi this is only a hack at the moment to get things running.
% \begingroup
\if@bw\G@slideswtrue\else
\if@visible \G@slideswtrue \fi
\fi
% \endgroup
\fi
\if\@slidesw \makingslidestrue\thispagestyle{slide}\%

This will set up the last color specified in the argument to \slide as the current color. If only back and white slides are prepared \last@color will be empty and effectively \relax will be generated (hopefully).

We need to reset to a default font at the beginning of a slide. (not done yet).

\csname \last@color \endcsname
\else\end{slide}\gobbletoend{slide}\fi}
% \fi
% if@compatibility
% \let\last@color\@empty
% \def\endslide{\par\break}
\overlay COLORS ::= BEGIN
\par\break
\stepcounter{overlay}
\setcounter{page}{0} % in case of non-slide pages
\@slidesw :=G T
if @onlyslidesw = T % set \@slidesw = T iff
then
% page to be output
if \c@slide < \@doslidelow
then \@slidesw :=G F
fi
fi
if \@slidesw = T
\@slidesw :=G F
\begin{group}
if @bw = true
then \@slidesw :=G T
else \@color{COLORS}
\if@visible then \@slidesw :=G T \fi
\end{group}
\if@slidesw = T
  \makingslides := T
  \thispagestyle{overlay}
else \end{overlay}
\gobbletoend{overlay}
\fi
END
\endoverlay ::= 
\begin{par}
\break
\end{par}
\end{overlay}
\def\overlay#1{
\stepcounter{overlay}
\setcounter{page}{0} % in case of non-slide pages
\if@bw = T
  \slidesw := G T
  \if @onlynotesw = true
    \setlimits\donotelow\donotehigh od
  then
\fi
\if@compatibility
  \def\overlay{
    \par
    \stepcounter{overlay}
    \setcounter{page}{0} % in case of non-slide pages
    \if@bw = T
      \slidesw := G T
      \if @onlynotesw = true
        \setlimits\donotelow\donotehigh od
      then
\fi
\end{par}
\end{overlay}
\else
  \end{overlay}
\fi
\end{par}
\end{overlay}
if \c@slide < \@donotelow
  then \@slidesw :=G F
  fi
else \@slidesw :=G F
fi
if \@slidesw = T
  then \@makingslides := T
\thispagestyle{note}
else \end{note}
\@gobbletoend{note}
fi
END
\endnote ::= BEGIN \par \break END
635 \if@compatibility
636 \def\note{\stepcounter{note}\%}
637 \if@bw
638 \G@slideswtrue
639 \if@onlynotesw\@whilenum \c@slide >\@donotehigh\relax
640 \do{\@setlimits\@donotelist\@donotelow\@donotehigh}\ifnum
641 \c@slide <\@donotelow\relax \G@slideswfalse\fi\fi
642 \else\G@slideswfalse\fi
643 \if@G@slidesw \newpage\thispagestyle{note}\else
644 \end{note}\@gobbletoend{note}\fi
645 \%
646 \else \%\if@compatibility
647 \%
648 \def\note{\par \break\stepcounter{note}\setcounter{page}{0}\%}
649 \if@bw
650 \G@slideswtrue
651 \if@onlynotesw\@whilenum \c@slide >\@donotehigh\relax
652 \do{\@setlimits\@donotelist\@donotelow\@donotehigh}\ifnum
653 \c@slide <\@donotelow\relax \G@slideswfalse\fi\fi
654 \else\G@slideswfalse\fi
655 \if@G@slidesw \@makingslidestrue\thispagestyle{note}\else
656 \end{note}\@gobbletoend{note}\fi
657 \fi \%\if@compatibility
658 \%
659 \def\endnote{\par \break}
\%\color{COLORS} ::=
BEGIN
  if math mode
    then type warning
  fi
  if \@bw
    then \visible
  else \invisible
    for \last@color := COLORS
      do if \last@color = \@currcolor

  28
FM: \last@color will be used in \slide to set up first color if no color is given. I suppose that this is much too complicated. \else \@tempswafalse would produce the same effect I imagine.

\def\@color#1{\@mmodetest{\if@bw \@tempswatrue \else \@tempswafalse \@for \reserved@a :=#1\do{\ifx\reserved@a\@currcolor\@tempswatrue\fi} \let\last@color\reserved@a\fi \if@tempswa \visible \else \invisible \fi \ignorespaces}}

\def\@mmodetest#1{\ifmmode\ClassWarning{slides}{Color-changing command in math mode has been ignored}\else #1\fi}

\def\invisible{\@mmodetest{\if@visible \@visiblefalse \fontshape{\f@shape}\selectfont\mathversion{invisible}\% \fi \ignorespaces}}

\def\visible{\@mmodetest{\if@visible \else \@visibletrue \fontshape{\expandafter\@gobble\f@shape}\selectfont\mathversion{normal}\% \fi \ignorespaces}}

\def\fontshape#1{\edef\f@shape{\if@visible \else I\fi#1}}

Here is the \LaTeX{} interface hidden. We use a trick to provide ourselves with a sort of additional attribute without making the current mechanism even larger. The trick is that we denote invisible by putting an uppercase I in front of the shape name for invisible shapes and remove it again if we want to become visible.

\renewcommand{\familydefault}{\sfdefault}

The \latexsym{} package, which is needed to be able to access the \LaTeX{} symbol fonts (lasy), sets things up so that for sizes larger than 10 point magnifications of lasy10 are used. For slides we want to use magnifications of lasy8, so we set up the lasy family here to prevent \LaTeX{} from loading Ulasy.fd.

10.3 Macros for font handling

We let \familydefault point at \sfdefault, to make it easier to use the document class slides with packages that set up other fonts.

\renewcommand{\familydefault}{\sfdefault}
10.3.1 Modifications to the picture environment

Below are the new definitions of the picture-drawing macros required for SLiTeX. Only those commands that actually draw something must be changed so that they do not produce any output when the @visible switch is false.

\def\line(#1,#2)#3\{
  \if@visible\@xarg #1\relax \@yarg #2\relax
  \@linelen #3\unitlength
  \ifnum\@xarg =\z@ \@vline
    \else \@ifnum\@yarg =\z@ \@hline \else \@sline\fi
  \fi\fi\}

\def\vector(#1,#2)#3\{
  \if@visible\@xarg #1\relax \@yarg #2\relax
  \@linelen #3\unitlength
  \ifnum\@xarg =\z@ \@vvector
    \else \@ifnum\@yarg =\z@ \@hvector \else \@svector\fi
  \fi\fi\}

\def\dashbox#1(#2,#3)\{
  \leavevmode
  \if@visible
    \hb@xt\z@\baselineskip \z@
    \lineskip \z@
    \@dashdim #2\unitlength
    \@dashcnt \@dashdim \advance\@dashcnt 200
    \@dashdim #1\unitlength \divide\@dashcnt \@dashdim
    \ifodd\@dashcnt \@dashdim=\z@
      \advance\@dashcnt \m@ne
    \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
    \fi
    \setbox\@dashbox \hbox{
      \vrule @height \@halfwidth @depth \@halfwidth
      \@width \@dashdim\hskip \@dashdim
    }
    \put(0,0){\copy\@dashbox}
    \put(0,#3){\copy\@dashbox}
    \put(#2,0){\hskip-\@dashdim\copy\@dashbox}
    \put(#2,#3){\hskip-\@dashdim\box\@dashbox}
    \multiply\@dashdim \thr@@
  \fi
  \leavevmode
  \hb@xt\z@\baselineskip \z@
  \lineskip \z@
  \@dashdim #2\unitlength
  \@dashcnt \@dashdim \advance\@dashcnt 200
  \@dashdim #1\unitlength \divide\@dashcnt \@dashdim
  \ifodd\@dashcnt \@dashdim=\z@
    \advance\@dashcnt \m@ne
  \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
  \fi
  \setbox\@dashbox \hbox{
    \vrule @height \@halfwidth @depth \@halfwidth
    \@width \@dashdim\hskip \@dashdim
  }
  \put(0,0){\copy\@dashbox}
  \put(0,#3){\copy\@dashbox}
  \put(#2,0){\hskip-\@dashdim\copy\@dashbox}
  \put(#2,#3){\hskip-\@dashdim\box\@dashbox}
  \multiply\@dashdim \thr@@
\fi
\leavevmode
\if@visible
  \hb@xt\z@\baselineskip \z@
  \lineskip \z@
  \@dashdim #2\unitlength
  \@dashcnt \@dashdim \advance\@dashcnt 200
  \@dashdim #1\unitlength \divide\@dashcnt \@dashdim
  \ifodd\@dashcnt \@dashdim=\z@
    \advance\@dashcnt \m@ne
  \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
  \fi
  \setbox\@dashbox \hbox{
    \vrule @height \@halfwidth @depth \@halfwidth
    \@width \@dashdim\hskip \@dashdim
  }
  \put(0,0){\copy\@dashbox}
  \put(0,#3){\copy\@dashbox}
  \put(#2,0){\hskip-\@dashdim\copy\@dashbox}
  \put(#2,#3){\hskip-\@dashdim\box\@dashbox}
  \multiply\@dashdim \thr@@
\fi

10.3.2 Other modifications to \TeX{} and \LaTeX{} commands

\rule
\frac {STYLE}{DENOMSTYLE}{NUM}{DEN}{FONTSIZE}:
Creates \frac{NUM}{DENOM} in style STYLE with NUM and DENOM in style DENOMSTYLE.
FONTSIZE should be \textfont \scriptfont or \scriptscriptfont.

Added a group around the first argument of \frac to prevent changes (for example, font changes) to modify the contents of the second argument.

\def\frac#1#2{
\mathchoice
\fracdisplaystyle\textstyle{#1}{#2}\textfont
\fracscriptstyle\scriptstyle{#1}{#2}\textfont
\fracscriptscriptstyle\scriptscriptstyle{#1}{#2}\scriptfont
\fracscriptscriptstyle\scriptscriptstyle{#1}{#2}\scriptscriptfont}

\def\@frac#1#2#3#4#5{\@mathbox#1\@smashboxc{{\begingroup#3\endgroup\over#4}}
\setbox\tw@\null
\ht\tw@ \ht\@smashboxc
\dp\tw@ \dp\@smashboxc
\wd\tw@ \wd\@smashboxc
\if@visible\@smashboxc\else\tw@\fi}

\def\r@@t#1#2{\setbox\z@hbox{$\m@th#1\@xysqrt#1{#2}$}
\dimen@\ht\z@ \advance\dimen@-\dp\z@
\mskip5mu\raise.6\dimen@\copy\rootbox\mskip-10mu\box\z@}

\def\sqrt{\@ifnextchar[\@sqrt}{\@xsqrt}
\def\@sqrt[#1]{\root #1\of}
\def\@xsqrt#1{\mathchoice{\@xysqrt\displaystyle{#1}}{\@xysqrt\textstyle{#1}}{\@xysqrt\scriptstyle{#1}}{\@xysqrt\scriptscriptstyle{#1}}}

\def\@xysqrt#1#2{\@mathbox#1\@smashboxa{#2}\if@visible\@hvsmash#1{\sqrtsign{\@bphantom\@smashboxa}}\fi}

\newbox\@smashboxa
\newbox\@smashboxb
\newbox\@smashboxc

array and tabular environments: changes to ‘|’, \hline, \cline, and \vline, added 8 Jun 88

\def\arrayrule{\if@visible\@addtopreamble{\hskip -.5\arrayrulewidth
\vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}\fi}

\def\cline{\if@visible\@cline\@nil\fi}

\def\vline{\if@visible \vrule \@width \arrayrulewidth\else \vrule \@width \arrayrulewidth \@height \z@ \@depth \z@ \fi}

\message{output,}
10.3.3 Changes to \LaTeX output routine

\@makecol ==
BEGIN
% Following test added for slides to check if extra page
if \@makingslides = T
  then if \c@page > 0
    then if \c@note > 0
      then type 'Note \thenote too long.'
    else if \c@overlay > 0
      then type 'Overlay \theoverlay too long,'
    else type 'Slide \theslide too long'
  fi fi fi fi
fi fi fi fi
ifvoid \insert\footins
  then \@outputbox := \box255
else \@outputbox := \vbox {\unvbox255
  \vskip \skip\footins
  \footnoterule
  \unvbox\@footinsert}
\@freelist := G \@freelist * \@midlist
\@midlist := G empty
\@combinefloats
\@outputbox := \vbox to \@colht{\boxmaxdepth := \maxdepth
  \vfil %\vfil added for slides
  \unvbox@outputbox
  \vfil } %\vfil added for slides
\maxdepth := G \@maxdepth
END

FMi simple hack to allow none centered slides Should be revised of course.
\let\@topfil\vfil
\def\@makecol{\if@makingslides\ifnum\c@page > \z@ \@extraslide\fi\fi
  \ifvoid\footins \setbox\@outputbox\box\@cclv \let\@botfil\vfil
  \else\let\@botfil\relax\setbox\@outputbox
    \vbox{\unvbox@outputbox\vfil
      \vskip\skip\footins\footnoterule\unvbox\footins\vskip
      \z@ plus.1fil\relax}\fi
  \xdef\@freelist{\@freelist\@midlist}\gdef\@midlist{G empty}\@combinefloats
  \setbox\@outputbox\vbox to\@colht{\boxmaxdepth=\maxdepth\@maxdepth
  \vfil \unvbox@outputbox \vfil}
\def\@extraslide{\ifnum\c@note > \z@
  \ClassWarning{slides}{Note \thenote space too long}\else
  \ifnum\c@overlay > \z@
  \ClassWarning{slides}{Overlay \theoverlay space too long}\else
    \ClassWarning{slides}{Slide \theslide space too long}\fi\fi}
\message{init}

10.3.4 Special \SLiTeX initializations

FMi why not allow for ref’s?
% \nofiles
\@visibletrue
\langle cmd\rangle