Producing slides with \LaTeX 2ε

Frank Mittelbach

2020/11/23

1 Introduction

With \LaTeX 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 \texttt{slides} as the document class. This class is very similar to the \texttt{slides} style that came with S\LaTeX, in fact it is basically a copy changed to work under \LaTeX 2ε.\(^1\) Thus you have to say something like

\begin{verbatim}
\documentclass[...]{slides}
\end{verbatim}

and process this with \LaTeX 2ε.

3 Fonts

Note, that with NFSS you can easily produce slides with special fonts just by calling an appropriate style file (like \texttt{times}) in a \texttt{\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 \texttt{pandora} won’t work because the standard \texttt{.fd} files shipped with NFSS only contain small sizes. You can, of course, produce additional sizes and change the \texttt{.fd} files accordingly so that they would be useable for slides as well.

4 Invisible text and color separation

In the original S\LaTeX it was possible to produce invisible text using the \texttt{\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.

\(^{1}\)Therefore you should compare the new class with old S\LaTeX styles in case you have local slide classes to see what you have to change in order to use them with \LaTeX 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{}2ε 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{}2ε 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 slides.sty (which then forms slides.cls) followed by a slightly changed copy of 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 \LaTeX{}.

\begin{verbatim}
\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 sfons.cfg. If there isn’t one, we fall back to the default one (sfons.def).

\begin{verbatim}
\InputIfFileExists{sfons.cfg}{\typeout{*****************************************************\%\%
  ** Local config file sfons.cfg used**\%\%
  *****************************************************}}\%
\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 \ 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
\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}{@titlepagetfale}

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.
\ProcessOptions

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

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

Since the number of parameters to set are very large it seems reasonable to set up one command \@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 \size executes the command \@setfontsize\size\langle\text{font-size}\rangle\langle\text{baselineskip}\rangle where:

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

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

\$\begin{array}{ll}
\@vpt & 5 \\
\@vipt & 6 \\
\@viipt & 7 \\
\@viiipt & 8 \\
\@ixpt & 9 \\
\@xpt & 10 \\
\@xipt & 10.95 \\
\@xiipt & 12 \\
\@xivpt & 14.4 \\
\@fortyonept & 41.28 \\
\end{array}\$

\$\begin{array}{ll}
\@ifourteenpt & For \Sl, however, these are not sufficient, and we therefore need to add a few \\
\@iseventeenpt & extra, larger, sizes. \\
\@itwentypt & \def\ifourteenpt{13.82} \\
\@itwentyfourpt & \def\iseventeenpt{16.59} \\
\@itwentyninept & \def\itwentypt{19.907} \\
\@ithirtyfourpt & \def\itwentyfourpt{23.89} \\
\@ifortyonept & \def\itwentyninept{28.66} \\
\end{array}\$

\$\@setfontsize@parms This routine is used in \Sl to interface font size setting it is modeled after the settings I found in 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:

\$\begin{array}{ll}
\@setfontsize@parms \langle\text{lineskip}\rangle \\
\langle\text{parskip}\rangle \\
\langle\text{abovedisplayskip}\rangle \\
\end{array}\$
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 \texttt{size} must be hidden in com-
mands denoting the size by its name prefixed with ‘\texttt{i}’, i.e. 20pt size is called
\texttt{\textit{twentypt}} at this point. The NFSS interface will define those sizes to ex-
pend to the internal size, e.g. 20 but for the old sfonts the command name,
e.g. \texttt{\textit{twentypt}}, will be used to construct the name \texttt{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 \texttt{\@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!

\begin{verbatim}
\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
\%
}
\end{verbatim}

I don’t see a reason why the \texttt{\strutbox} has a dim different from \texttt{\baselineskip}
but we will leave it for the moment

\begin{verbatim}
\setbox\strutbox=\hbox{\vrule \@height#7\p@\@depth#8\p@\@width\z@}\%
\baselineskip\baselinestretch\baselineskip\%
\normalbaselineskip\baselineskip\%
\end{verbatim}

Setting size relations for math scripts:

\begin{verbatim}
\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}
\end{verbatim}
We initially choose the normalsize font.
\normalsize
\small
\def\small{\@setfontsize\small\iseventeenpt{19\p@ plus3\p@ minus\p@}\
\@setfontsize@parms\
{2\p@}\
{15\p@ plus15\p@ minus7\p@}\%
{12\p@ plus3\p@ minus3\p@}\%
{9\p@ plus3\p@ minus3\p@}\%
{6\p@ plus3\p@}\%
\abovedisplayshortskip\
{13.5\{5.6}}}
\footnotesize\scriptsize
\let\footnotesize=\small
\let\scriptsize=\small
\tiny
\def\tiny{\@setfontsize\tiny\ifourteenpt{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@ plus3\p@ minus5\p@}\%
{4\p@ plus3\p@}\%
\abovedisplayskip\
{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 \set@font@size@parms.
\large
\Large
\Huge
\def\Large{\@setfontsize\Large\itwentyonept{40\p@ plus4\p@ minus\p@}\
\@setfontsize@parms\
{2\p@}\
{18\p@ plus15\p@ minus6\p@}\%
{12\p@ plus3\p@ minus3\p@}\%
\abovedisplayshortskip\
{10\{6.5\}}}
\def\Large{\@setfontsize\Large\itwentyonept{40\p@ plus4\p@ minus\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.
\renewcommand{\baselinestretch{}}

\parindent
\parindent is the width of the paragraph indentation.
\setlength{\parindent}{0cm}

\@lowpenalty
\@medpenalty
\@highpenalty
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.
\setlength{\@lowpenalty}{51}
\setlength{\@medpenalty}{151}
\setlength{\@highpenalty}{301}

\clubpenalty
\widowpenalty
These penalties are use to discourage club and widow lines. Because we use their default values we only show them here, commented out.
\setlength{\clubpenalty}{150}
\setlength{\widowpenalty}{150}

\displaywidowpenalty
\predisplaypenalty
\postdisplaypenalty
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.
\setlength{\displaywidowpenalty}{50}
\setlength{\predisplaypenalty}{10000}
Allow the breaking of a page in the middle of a paragraph.

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

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. The topskip is the baselineskip for the first line on a page.

\begin{verbatim}
\setlength\headheight{14\p@}
\setlength\headsep {15\p@}
\setlength\topskip {30\p@}
\end{verbatim}

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:

\begin{verbatim}
\setlength\footskip{25\p@} %
\end{verbatim}

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 = typesize \times 1.5. As it happens, in these classes \maxdepth is equal to the typesize, therefor we set \maxdepth to half the value of \topskip.

\begin{verbatim}
\if@compatibility
\setlength\maxdepth{4\p@}
\else
\setlength\maxdepth{.5\topskip}
\fi
\end{verbatim}

#### 9.3.2 The dimension of text

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.

\begin{verbatim}
\if@compatibility
\setlength\textwidth{460\p@}
\else
\end{verbatim}

When we are not in compatibility mode we can set some of the dimensions differently, taking into account the paper size for instance.

\begin{verbatim}
\else
\end{verbatim}
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 \textwidth and \parindent.

\setlength{\textwidth}{\paperwidth - 2in}
\setbox{\hbox{im}}
\setlength{\parindent}{.5\wd{\hbox{im}}}
\setlength{\textwidth}{65\parindent}

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{\paperwidth}>\textwidth\relax
\setlength{\textwidth}{\textwidth}
\else
\setlength{\textwidth}{\paperwidth - 2in}
\fi

Here we modify the width of the text a little to be a whole number of points.

\settoptextheight{\textwidth}
\setcolumnwidth{\textwidth}
\setcolumnsep{10pt}
\setcolumnseprule{z@}
\settextheight{600\p@}

Now that we have computed the width of the text, we have to take care of the height. The \textheight 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 \textheight minus \topskip, divided by \baselineskip for \normalsize. The old value of \textheight was 528pt.

\if@compatibility
\setlength{\textheight}{600\p@}
\else
\setlength{\tempdima}{\paperheight - 2in}
\addtolength{\tempdima}{-1in}
\setlength{\tempdima}{\tempdima}
\fi

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.
From this we can calculate the height of the text.

The first line on the page has a height of \topskip.

\advance\textheight by \topskip

\section*{Margins}

First we give the values for the compatibility mode.

Values for two-sided printing:

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−2in. 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.

The \evensidemargin can now be computed from the values set above.

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.

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

9.3.4 Footnotes

\footnotepsep \footnotepsep 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{\footnotepsep}{20\p@}

\footins \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).

\begin{verbatim}
\ps@headings
\if@compatibility
\def\ps@headings{\def\ps@slide{\def\@oddfoot{\@mainsize +\hfil\hb@xt@3em{\theslide\hss}}%}
\def\@oddhead{\@mainsize +\hfil+}%
\def\@evenfoot{\@mainsize +\hfil\hb@xt@3em{\theslide\hss}}%
\def\@evenhead{\@mainsize +\hfil+}}%\fi
\end{verbatim}

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.

\texttt{\ps@headings}

\begin{verbatim}
\def\ps@headings{\if@compatibility
\def\ps@headings{\def\ps@slide{\def\@oddfoot{\@mainsize +\hfil\hb@xt@3em{\theslide\hss}}%}
\def\@oddhead{\@mainsize +\hfil+}%
\def\@evenfoot{\@mainsize +\hfil\hb@xt@3em{\theslide\hss}}%
\def\@evenhead{\@mainsize +\hfil+}}%\fi
\end{verbatim}
\def\@evenhead{\@mainsize +\hfil +}}
\def\@oddfoot{\@mainsize \hbox{\hfil\thenote}}
\def\@oddhead{}}
\def\@evenfoot{\@mainsize \hbox{\hfil\thenote}}
\def\@evenhead{}}
\else \%\if@compatibility
\% \%
\def\ps@headings{\%\def\ps@slide{\%\def\@oddfoot{\@mainsize \hbox{\hfil\@arabic\c@slide\hss}}}\%\def\@oddhead{}}\%\def\@evenfoot{\@mainsize \hbox{\hfil\@arabic\c@slide\hss}}}\%\def\@evenhead{}}\%
\def\ps@overlay{\%\def\@oddfoot{\@mainsize \hbox{\hfil\@arabic\c@overlay\hss}}}\%\def\@oddhead{}}\%\def\@evenfoot{\@mainsize \hbox{\hfil\@arabic\c@overlay\hss}}}\%\def\@evenhead{}}\%
\def\ps@note{\%\def\@oddfoot{\@mainsize \hbox{}\hfil\thenote}}\%\def\@oddhead{}}\%\def\@evenfoot{\@mainsize \hbox{}\hfil\thenote}}}\%\def\@evenhead{}}}
\ps@plain
\ps@plain{\def\ps@slide{\%\def\@oddfoot{\@mainsize \hbox{\hfil\@arabic\c@slide\hss}}}\%\def\@oddhead{}}\%\def\@evenfoot{\@mainsize \hbox{\hfil\@arabic\c@slide\hss}}}\%\def\@evenhead{}}\%
\def\ps@overlay{\def\@oddfoot{\@mainsize \hbox{\hfil\@arabic\c@overlay\hss}}}\%\def\@oddhead{}}\%\def\@evenfoot{\@mainsize \hbox{\hfil\@arabic\c@overlay\hss}}}\%\def\@evenhead{}}\%
\def\ps@note{\def\@oddfoot{\@mainsize \hbox{}\hfil\thenote}}
\% \% if@compatibility
\ps@plain
\def\@oddhead{}% 
\def\@evenfoot{\@mainsize \hbox{}\hfil\thenote}% 
\def\@evenhead{}}%

\ps@empty

\def\ps@empty{% 
\def\ps@slide{\def\@oddhead{}\def\@oddfoot{}% 
\def\@evenhead{}\def\@evenfoot{}}% 
\def\ps@overlay{\def\@oddhead{}\def\@oddfoot{}% 
\def\@evenhead{}\def\@evenfoot{}}% 
\def\ps@note{\def\@oddhead{}\def\@oddfoot{}% 
\def\@evenhead{}\def\@evenfoot{}}}%

Default definition the 'slide', 'overlay', and 'note' page styles.

\ps@headings

Set ordinary page style to 'empty'

\let\@oddhead\@empty\let\@oddfoot\@empty
\let\@evenhead\@empty\let\@evenfoot\@empty

9.5 Providing math versions

\LaTeX provides two versions. We call them normal and bold, respectively. \Silent\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}{lcmsy}{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}{lcmsy}{m}{In}

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

\texttt{titlepage}\ This environment starts a new page, with pagestyle \texttt{empty}\ and sets the page counter to 0.
\begin{verbatim}
\newenvironment{titlepage}{\newpage\	\thispagestyle{empty}\	\setcounter{page}{\z@}}{\newpage}
\end{verbatim}

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 \setlength{\leftmargini}{38\p@}
\leftmarginii \setlength{\leftmarginii}{30\p@}
\leftmarginiii \setlength{\leftmarginiii}{20\p@}
\leftmarginiv \setlength{\leftmarginiv}{15\p@}
\leftmarginv \setlength{\leftmarginv}{15\p@}
\leftmarginvi \setlength{\leftmarginvi}{10\p@}
\@listi \def\@listi{
\leftmargin\leftmargini
\parsep .5\parskip
\topsep \parsep
\itemsep \parskip}
\@listii \def\@listii{
\leftmargin\leftmarginii
\labelwidth\leftmarginii
\advance\labelwidth-\labelsep
\parsep .5\parskip
\topsep \parsep
\itemsep \parskip}
\@listiii \def\@listiii{
\leftmargin\leftmarginiii
\labelwidth\leftmarginiii
\advance\labelwidth-\labelsep}
\@listiv \def\@listiv{
\leftmargin\leftmarginiv
\labelwidth\leftmarginiv
\advance\labelwidth-\labelsep}
\@listv \def\@listv{
\leftmargin\leftmarginv
\labelwidth\leftmarginv
\advance\labelwidth-\labelsep}
\@listvi \def\@listvi{
\leftmargin\leftmarginvi
\labelwidth\leftmarginvi
\advance\labelwidth-\labelsep}
\end{verbatim}
Here we initialize `\leftmargin` and `\labelwidth`.

9.6.2 Paragraph-formatting environments

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

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

**quotation** The `quotation` environment fills lines, indents paragraphs.

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

**quote** The `quote` environment is the same as the `quotation` environment, except that there is no paragraph indentation.

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

9.6.3 List-making environments

**description** The description environment is defined here – while the itemize and enumerate environments are defined in `latex.dtx`.

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

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

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

9.6.4 Enumerate

The enumerate environment uses four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enumN` controls the numbering of the Nth level enumeration.

```latex
\theenumi \theenumii \theenumiii \theenumiv
```

The counters are already defined in `latex.dtx`, but their representation is changed here.

```latex
\renewcommand{\theenumi}{\@arabic{\c@enumi}}
\renewcommand{\theenumii}{\@alph{\c@enumii}}
```
The label for each item is generated by the four commands \labelenumi ... \labelenumiv.

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

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

Array and tabular

The columns in an array environment are separated by 2\arraycolsep. Array and tabular environment parameters

The columns in an tabular environment are separated by 2\tabcolsep.

The width of rules in the array and tabular environments is given by the length parameter\arrayrulewidth.

The space between adjacent rules in the array and tabular environments is given by \doublerulesep.

Tabbing

This controls the space that the \! command puts in. (See \LaTeX\ manual for an explanation.)

\labelsep 10pt
9.7.3 Minipage

\@minipagerestore 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; \textbackslash{skip}\@mpfootins plays same rôle for footnotes in a minipage as \textbackslash{skip}\textbackslash{footins} does for ordinary footnotes.

\textbackslash{skip}\@mpfootins = \textbackslash{skip}\textbackslash{footins}

9.7.4 Framed boxes

\fboxsep The space left by \textbackslash{fbox} and \textbackslash{framebox} between the box and the text in it.

\fboxrule The width of the rules in the box made by \textbackslash{fbox} and \textbackslash{framebox}.

\setlength{\fboxsep}{5\textbackslash{p@}}
\setlength{\fboxrule}{.6\textbackslash{p@}}

\theequation The equation number will be typeset as arabic numerals.

\def\theequation{\@arabic\c@equation}

\jot \textbackslash{jot} is the extra space added between lines of an eqnarray environment. The default value is used.

\setlength{\jot}{3pt}

\@eqnnum The macro \@eqnnum defines how equation numbers are to appear in equations. Again the default is used.

\def\@eqnnum{\textbackslash{\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 \textbackslash{text}..., and \textbackslash{math}... commands instead. These commands are redefined using \textbackslash{DeclareOldFontCommand}, a command with three arguments: the user command to be defined, \textbackslash{LATEX} commands to execute in text mode and \textbackslash{LATEX} commands to execute in math mode.

\textbackslash{rm} The commands to change the family. When in compatibility mode we select the ‘default’ font first, to get \textbackslash{LATEX}2.09 behaviour.

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

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

The commands \texttt{\cal} and \texttt{\mit} should only be used in math mode, outside math 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’.

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

\section*{9.9 Footnotes}

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

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

\texttt{\c@footnote} Footnotes are numbered within slides, overlays, and notes and numbered with *, †, etc.

\texttt{442} \texttt{\% \newcounter{footnote}}
\texttt{443} \texttt{\def{\thefootnote}{\fnsymbol{footnote}}}
\texttt{444} \texttt{\@addtoreset{footnote}{slide}}
\texttt{445} \texttt{\@addtoreset{footnote}{overlay}}
\texttt{446} \texttt{\@addtoreset{footnote}{note}}

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

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

\begin{verbatim}
\long\def\@makefntext#1{\%\setpar{0\par
      \@tempdim = \hsize
      \advance\@tempdim-10pt
      \parshape \one 10pt \@tempdim\%
      \parindent 1em\noindent
      \hbox to \z{\@makefnmark#1}}
\end{verbatim}

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.

%\def\@makefnmark{
%  \hbox{$^\@thefnmark\m@th$}}

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 \LaTeX-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
⟨/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

(*cmd)
\message{hacks,}
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

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

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

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

\newif\@makingslides

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 ‘ckt list’ so that the \include mechanism works.
\g@addto@macro\cl@cktpt{\@elt{slide}\@elt{overlay}\@elt{note}}
\@addtoreset{overlay}{slide}
\@addtoreset{note}{slide}
\@definecounter{page}
\@addtoreset{page}{slide}
\@addtoreset{page}{note}

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.
\@setlimits \LIST \LOW \HIGH

Assumes that \LIST = RANGE1,RANGE2,...,RANGE_n \text{ (n>0)}
Where RANGE_i = j or j-k.

Then \@setlimits globally sets
(i) \LIST := RANGE_2, ..., RANGE_n
(ii) \LOW := p
(iii) \HIGH := q
where either RANGE_i = p-q or RANGE_i = 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
  @onlyslidew := 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 := G LIST,999999,999999
  fi
message: Only Notes LIST
END
\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
  @onlyslidew := 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 := G LIST,999999,999999
  fi
message: Only Notes LIST
END
\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
  @onlyslidew := 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 := G LIST,999999,999999
  fi
message: Only Notes LIST
END
\def\onlynotes#1{\@onlynoteswtrue
\gdef\@doglnotelist{#1,999999,999999}\
\if\@onlynotesw \else
  \@onlynoteswtrue\gdef\@doglnotelist{999999,999999}\fi
\typeout{Only Notes #1}}
\def\onlyslides#1{\@onlyslideswtrue
\gdef\@doglslidelist{#1,999999,999999}\
\if\@onlynotesw \else
  \@onlyslideswtrue\gdef\@doglslidelist{999999,999999}\fi
\typeout{Only Slides #1}}
\setupcounters ::= (similar to old \blackandwhite #1 ::= )
newpage
page counter := 0
@bw := T
@visible := T
if @onlyslidesw = true
then \doslidelist := \dodslidalist
  \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
\if@compatibility
% In compatibility mode, need to define \verb+\blackandwhite+, \verb+colors+, \verb+colorslides+, etc.
\def\blackandwhite#1{\newpage\setcounter{page}{0}\@bwtrue\@visibletrue
\if@onlyslidesw \xdef\doslidelist{\dodslidalist}\% \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}\@makingslidestrue\input #1\@makingslidesfalse}
\colors{COLORS} ::= for \@colortemp := COLORS
  do \csname \@colortemp \endcsname == \@color{\@colortemp} od
if \@colorlist = empty
then \@colorlist := COLORS
else \@colorlist := \@colorlist, COLORS
fi\def\colors#1{\@for\@colortemp:=#1\do{\expandafter\xdef\csname\@colortemp\endcsname{\noexpand\@color{\@colortemp}}}\ifx\@colorlist\@empty \gdef\@colorlist{#1}\else \xdef\@colorlist{\@colorlist,#1}\fi}
532 \def\@colorlist{}
\colorslides{FILE} ::= \newpage
page counter := 0
@bw := F
for \@currcolor := \@colorlist
  do \@visible := T
  if @onlyslidesw = true
    \doslidelist := \dodslidalist
    \setlimits\doslidelist\doslidelow\doslidehigh
  elif @onlynotesw = true
    \donotelist := \doglnotelist
    \setlimits\donotelist\donotelow\donotehigh
  fi
\normalsize % Note, this sets font to \rmfamily , which sets % \@currfont to \rmfamily
then \doslidelist := \doglslidelist
\setlimits\doslidelist\doslidelow\doslidehigh
fi
if @onlynotesw = true
then \donotelist := \doglnotelist
\setlimits\donotelist\donotelow\donotehigh
fi
\normalsize
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}\fi
\setlimits\doslidelist\doslidelow\doslidehigh\fi
\if@onlynotesw \xdef\donotelist{\doglnotelist}\fi
\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}\@bwtrue\@visibletrue
\if@onlyslidesw \xdef\doslidelist{\doglslidelist}\fi
\setlimits\doslidelist\doslidelow\doslidehigh\fi
\if@onlynotesw \xdef\donotelist{\doglnotelist}\fi
\setlimits\donotelist\donotelow\donotehigh\fi
\normalsize\setcounter{slide}{0}\setcounter{overlay}{0}%
\setcounter{note}{0}\@makingslidesfalse}
% \else % if@compatibility
\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\@doslidelow\@doslidehigh od
if \c@slide < \@doslidelow
  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 @makingslides := T \thispagestyle{slide}
else \end{slide} \@gobbletoend{slide} fi
END
\endslide ::= BEGIN \par\break END
567 \if@compatibility 568 \def\slide#1{\stepcounter{slide}\G@slideswtrue\if@onlyslidesw 569 \@whilenum \c@slide >\@doslidehigh\relax
570 \do{\setlimits\@doslidelist\@doslidelow\@doslidehigh}\ifnum
571 \c@slide <\@doslidelow\relax\G@slideswfalse\fi\fi
572 \if@slidesw
573 \G@slideswfalse
574 \% FM this is only a hack at the moment to get things running.
575 \% \begin{group}
576 \if@bw\G@slideswtrue\else
577 \color{#1}\if@visible \G@slideswtrue \fi
578 \fi
579 \% \end{group
580 \fi
581 \if@slidesw \newpage\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).

\begin{slide} \gobbletoend{slide} \end{slide}

% if@compatibility
\def\slide\{
\stepcounter{slide}\setcounter{page}{0}\G@slideswtrue
\if@onlyslidesw
\@whilenum \c@slide > \@doslidehigh \relax
\do{\@setlimits \@doslidelist \@doslidelow \@doslidehigh} \ifnum \c@slide < \@doslidelow \relax \G@slideswfalse \fi \fi
\ifG@slidesw
\G@slideswfalse % FMi this is only a hack at the moment to get things running.
% \begingroup
\if@bw \G@slideswtrue \else \fi
% \endgroup
\fi
\ifG@slidesw \@makingslidestrue \thispagestyle{slide} %
\begin{overlay COLORS :=
BEGIN \par\break
\setcounter{overlay}{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 \beginninggroup
if @bw = true
then \@slidesw :=G T \else \@color\{COLORS\} \if@visible then \@slidesw :=G T \fi
\fi
\end{overlay}
end of overlay COLORS
end of slide
\end{slide}
\if\@slidesw = T
  \begin{overlay}
  \if@onlyslidesw \ifnum \c@slide < \@doslide low \relax
    \ifG@slidesw \G@slideswfalse \fi\fi
  \else\if\@visible \G@slideswtrue \fi\fi\endgroup
  \ifG@slidesw \newpage \thispagestyle{overlay} \else \end{overlay} \@gobbletoend{overlay} \fi
\fi
\end{overlay} ::=
\begin{par}
\stepcounter{overlay}
\setcounter{page}{0} % in case of non-slide pages
\if@bw = T
  \then \@slidesw := \G T
  \if\@onlynotesw = \true % set \@notesw = T iff
    \then \% page to be output
      \while \c@slide > \@donote high
        \@setlimits \@donotelist \@donotelow \@donote high \od
      \fi
    \fi
  \fi
\end{par}
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}\fi\num
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}\fi\num
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
then \texttt{\visible}
\fi
\od
\fi
\ignorespaces
\END

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

\begin{verbatim}
660 \def\@color#1{\@mmodetest
661 {\if@bw \@tempswatrue \else \@tempswafalse
662 \@for \reserved@a :=#1\do{\ifx\reserved@a\@currcolor\@tempswatrue\fi
663 \let\last@color\reserved@a}\fi
664 \if@tempswa \visible \else \invisible \fi
665 \ignorespaces}}
666 \def\@mmodetest#1{\ifmmode\ClassWarning{slides}{Color-changing command
667 in math mode has been ignored}\else #1\fi}
668 \def\invisible{\@mmodetest
669 {\if@visible
670 \@visiblefalse
671 \fontshape{\f@shape}\selectfont
672 \mathversion{invisible}\%
673 \fi
674 \ignorespaces}}
675 \def\visible{\@mmodetest
676 {\if@visible
677 \else
678 \@visibletrue
679 \fontshape{\f@shape}\selectfont
680 \mathversion{normal}\%
681 \fi
682 \ignorespaces}}
683 \def\fontshape#1{\edef\f@shape{\if@visible \else I\fi #1}}
\end{verbatim}

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 \texttt{I} in front of the shape name for invisible shapes and remove it again if we want to become visible.

\begin{verbatim}
10.3 Macros for font handling
We let \texttt{\familydefault} point at \texttt{\sfdefault}, to make it easier to use the document class slides with packages that set up other fonts.
\renewcommand{\familydefault}{\sfdefault}

The \texttt{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 \texttt{lasy10} are used. For slides we want to use magnifications of \texttt{lasy8}, so we set up the lazy family here to prevent \LaTeX{} from loading \texttt{Ulasy.fd}.
\end{verbatim}
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.

\begin{verbatim}
\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@ \divide\@dashcnt \tw@ \advance\@dashcnt \m@ne \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth \@width \@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 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth \@width \@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}
\end{verbatim}

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

\rule

\message{mods,}

% _ (Added 10 Nov 86)
\overline, \underline, \frac and \sqrt

\@mathbox{STYLE}{BOX}{MTEXT} : Called in math mode, typesets MTEXT and stores result in BOX, using STYLE.

\@bphantom{BOX} : Creates a phantom with dimensions BOX.
\@vbphantom{BOX} : Creates a phantom with ht of BOX and zero width.
\@hbphantom{BOX} : Creates a phantom with width of BOX and zero ht & dp.
\@hvsmash{STYLE}{MTEXT} : Creates a copy of MTEXT with zero height and width in STYLE.

\def\@mathbox#1#2#3{\setbox#2\hbox{$\m@th#1{#3}$}}
\def\@vbphantom#1{\setbox\tw@\null \ht\tw@\ht #1\dp\tw@\dp #1\box\tw@}
\def\@bphantom#1{\setbox\tw@\null \wd\tw@\wd #1\ht\tw@\ht #1\dp\tw@\dp #1\box\tw@}
\def\@hbphantom#1{\setbox\tw@\null \wd\tw@\wd #1\ht\tw@\z@ \dp\tw@\z@ \box\tw@}
\def\@hvsmash#1#2{\@mathbox#1\z@{#2}\ht\z@\dp\z@\wd\z@\z@ \box\z@}
\def\underline#1{\relax\ifmmode\@xunderline{#1}\else$\m@th\@xunderline{\hbox{#1}}\relax\fi}
\def\@xunderline#1{\mathchoice{\@xyunderline\displaystyle{#1}}{\@xyunderline\textstyle{#1}}{\@xyunderline\scriptstyle{#1}}{\@xyunderline\scriptscriptstyle{#1}}}
\def\@xyunderline#1#2{\@mathbox#1\@smashboxa{#2}\@hvsmash#1{\copy\@smashboxa}\if@visible\@hvsmash#1{\@@underline{\@bphantom\@smashboxa}}\fi\@mathbox#1\@smashboxb{\@@underline{\box\@smashboxa}}\@bphantom\@smashboxb}

\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\def\@xoverline#1#2{\@mathbox#1\@smashboxa{#2}\@hvsmash#1{\copy\@smashboxa}\if@visible\@hvsmash#1{\@@overline{\@bphantom\@smashboxa}}\fi\@mathbox#1\@smashboxb{\@@overline{\box\@smashboxa}}\@bphantom\@smashboxb}

\let@\@overline=\overline

\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\def\@xoverline#1#2{\@mathbox#1\@smashboxa{#2}\@hvsmash#1{\copy\@smashboxa}\if@visible\@hvsmash#1{\@@overline{\@bphantom\@smashboxa}}\fi\@mathbox#1\@smashboxb{\@@overline{\box\@smashboxa}}\@bphantom\@smashboxb}
\frac{\text{STYLE}}{\text{DENOMSTYLE}}{\text{NUM}}{\text{DEN}}{\text{FONTSIZE}}:

Creates $\frac{\text{NUM}}{\text{DENOM}}$

in style \text{STYLE} with \text{NUM} and \text{DENOM} in style \text{DENOMSTYLE}

FONTSIZE should be \text{\textfont} \text{\scriptfont} or \text{\scriptscriptfont}

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

\def\frac#1#2{\mathchoice{
\text{\frac}{\text{\displaystyle}}{\text{\textstyle}}{#1}{#2}\text{\textfont}{\text{\scriptfont}}}{\text{\frac}{\text{\textstyle}}{\text{\scriptstyle}}{#1}{#2}\text{\textfont}{\text{\scriptfont}}}{\text{\frac}{\text{\scriptstyle}}{\text{\scriptscriptstyle}}{#1}{#2}\scriptfont}{\text{\frac}{\text{\scriptscriptstyle}}{\text{\scriptscriptstyle}}{#1}{#2}\scriptscriptfont}}
10.3.3 Changes to \TeX 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
}
fi
\@freelist := @\@freelist * \@midlist
\@midlist := @empty
@combinefloats
\@outputbox := \vbox to \@colht{\boxmaxdepth := \maxdepth
  \vfil \%\vfil added for slides
  \unvbox@outputbox
  \vfil } \%\vfil added for slides
\maxdepth := @\@maxdepth
END

FMi simple hack to allow none centered slides Should be revised of course.
\let\@topfil\vfil
917 \let\@topfil\vfil
918 \def\@makecol{\if@makingslides{\ifnum\c@page>0 \@extraslide\fi\fi
919 \ifvoid\footins \setbox\@outputbox\box\cclv \let\@botfil\vfil
920 \else\let\@botfil\relax\setbox\@outputbox
921 \vbox{\unvbox\cclv\vfil
922 \vskip\skip\footins\footnoterule\unvbox\footins\vskip
923 \z@ plus.1fil\relax}\fi
924 \xdef\@freelist{\@freelist\@midlist}\gdef\@midlist{\@combinefloats
925 \setbox\@outputbox\vbox to\@colht{\boxmaxdepth\maxdepth
926 \vfil\@topfil\unvbox\@outputbox\@botfil\global\maxdepth\@maxdepth}
927 \@topfil\unvbox\@outputbox\@botfil\global\maxdepth\@maxdepth}
928 \def\@extraslide{\ifnum\c@note>0 \z@\fi
929 \ClassWarning{slides}{Note \thenote space too long}\else
930 \ifnum\c@overlay>0 \z@\fi
931 \ClassWarning{slides}{Overlay \theoverlay space too long}\else
932 \ClassWarning{slides}{Slide \theslide space too long}\fi
933 \message{init}

10.3.4 Special \LaTeX initializations

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