Producing slides with $\LaTeX\,2\varepsilon$

Frank Mittelbach

2020/02/20

1 Introduction

With $\LaTeX\,2\varepsilon$ 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 \TeX, in fact it is basically a copy changed to work under $\LaTeX\,2\varepsilon$.\footnote{Therefore you should compare the new class with old \TeX \ styles in case you have local slide classes to see what you have to change in order to use them with $\LaTeX\,2\varepsilon$.} Thus you have to say something like

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

and process this with $\LaTeX\,2\varepsilon$.

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 \TeX \ 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.
The availability of color printers and the color package make color separation obsolete, so it has been removed. Although the color has also made \texttt{invisible} obsolete, the command is retained in the \LaTeX{} 2\epsilon implementation, but there are a few restrictions. Invisible fonts are implemented as special shapes where the shape names are built by prefixing the normal shape name with an uppercase I. For example, the ‘normal invisible shape’ would be \texttt{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{} 2\epsilon 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 \texttt{\DeclareFontShape} settings may not contain invisible font shape groups and thus you may be unable to use these features without adding additional \texttt{\DeclareFontShape} commands to your .fd files or the preamble of your document.

\section{The Implementation}

\textbf{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 lowlevel 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.

\subsection{The class code}

At this point we input the redefinitions that are necessary for \史料\TEX{}.

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

\begin{verbatim}
\InputIfFileExists{sfonts.cfg}
\{\typeout{******************************************************************************%\input{sfonts.def}}%
\end{verbatim}

\subsection{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}}
\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{\@tempdima{\the\minutes minutes}}
\newcounter{minutes}
\newcounter{seconds}
\newcommand*{\settime}{\addtocounter{seconds}{0}
\addtime{#1}}
\newcommand*{\addtime}{\addtocounter{seconds}{#1}
\setcounter{minutes}{\value{seconds}}
\global\divide\value{minutes}\by60}
\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}{\@titlepagefalse}

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 \Setfontsizes@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 \Setfontsizes\size(font-size)\baselineskip where:

\baselineskip The normal value of \baselineskip for the size of the font selected. (The actual value will be \baselinestretch * \baselineskip.)

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

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

\ifourteenpt For \Slshape{}, however, these are not sufficient, and we therefore need to add a few extra, larger, sizes.

\iseventeenpt \itwentypt \itwentyfourpt \itwentyninept \ithirtyfourpt \ifortyonept

\Setfontsizes@parms This routine is used in \Slshape{} 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{verbatim}
\Setfontsizes@parms \lineskip \parskip \abovedisplayskip
\end{verbatim}
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 \texttt{\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. \texttt{\itwentypt}, 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!

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

Setting size relations for math scripts:

```latex
\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
\setbox\strutbox=\hbox{\vrule \@height7\p@\@depth8\p@\@width\z@}%
\baselineskip\baselinestretch\baselineskip
\normalbaselineskip\baselineskip
}\
```

```
\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}
```
We initially choose the normalsize font.

\small
\footnotesize
\scriptsize
\tiny
\large
\Large
\Huge
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.
9.2 Paragraphing

\baselineskip 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 discourrage 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 \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.
\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:
\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 \LaTeX2e\ 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 \topskip\ is equal to the typesize, therefor we set \maxdepth\ to half the value of \topskip.
\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 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

Here we modify the width of the text a little to be a whole number of points.
\@settopoint{\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 \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}\addtolength{\@tempdima}{-2in}\setlength{\@tempdimb}{\@tempdima}\addtolength{\@tempdimb}{-1in}\fi

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

\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 \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}\addtolength{\@tempdima}{-2in}\setlength{\@tempdimb}{\@tempdima}\addtolength{\@tempdimb}{-1in}\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.

\else\setlength{\@tempdima}{\paperheight}\addtolength{\@tempdima}{-2in}\setlength{\@tempdimb}{\@tempdima}\addtolength{\@tempdimb}{-1in}\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.
\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

\section{Margins}

First we give the values for the compatibility mode.

Values for two-sided printing:
\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 – 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.
\setlength\@tempdima{\paperwidth}
\addtolength\@tempdima{-2in}
\addtolength\@tempdima{-\textwidth}
\setlength\oddsidemargin{.5\@tempdima}
\setlength\marginparwidth{.8in}
\addtolength\marginparwidth{.5\@tempdima}
\fi

The \evensidemargin can now be computed from the values set above.
\setlength\evensidemargin{\@tempdima}\setlength\evensidemargin{-2in}
\addtolength\evensidemargin{-\textwidth}
\addtolength\evensidemargin{\oddsidemargin}
\fi

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

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

9.3.4 Footnotes

\footnotesep \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@}

\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\p@ \minus\p@}

9.4 Page Styles

The page style foo is defined by defining the command \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).

The \ps@... command defines the macros \@oddhead, \@oddfoot, \@evenhead, and \@evenfoot to define the running heads and feet—e.g., \@oddhead is the macro to produce the contents of the heading box for odd-numbered pages. It is called inside an \hbox of width \textwidth.

The page styles of slides is determined by the ‘slide’ page style, the slide environment executing a \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 re-defining the ‘slide’, ‘overlay’, and ‘note’ styles.

\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 +}%
\def\ps@overlay{\def\@oddfoot{\@mainsize +\hfil\hb@xt@3em{\theoverlay \hss}}%}
\def\@oddhead{\@mainsize +\hfil +}%
\def\@evenfoot{\@mainsize +\hfil\hb@xt@3em{\theoverlay\hss}}%}
\fi
\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. \LaTeX{} 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}
\SetSymbolFont{operators}{invisible}

\SetSymbolFont{letters}{normal}
\SetSymbolFont{letters}{invisible}

\SetSymbolFont{symbols}{normal}
\SetSymbolFont{symbols}{invisible}

\SetSymbolFont{largesymbols}{normal}
\SetSymbolFont{largesymbols}{invisible}

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

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.

\leftmargini
\leftmarginii
\leftmarginiii
\leftmarginiv
\leftmarginv
\leftmarginvi

These commands set the values of \leftmargin, \parsep, \topsep, and \itemsep for the various levels of lists. It is even necessary to initialize \leftmargin in \@listi, i.e., for a level one list, as a list environment may appear inside a \texttt{trivlist}, for example inside a \texttt{theorem} environment.

\@listi
\@listii
\@listiii
\@listiv
\@listv
\@listvi

\@listi
\@listii
\@listiii
\@listiv
\@listv
\@listvi

15
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.

quotation The quotation environment fills lines, indents paragraphs.

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

9.6.3 List-making environments

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

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

9.6.4 Enumerate

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

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

\renewcommand*{\theenumi}[1]{\@arabic{\c@enumi}}
\renewcommand*{\theenumii}[1]{\@arabic{\c@enumii}}
The label for each item is generated by the four commands \labelenumi ... \labelenumiv.

\renewcommand\p@enumii{\theenumi}
\renewcommand\p@enumiii{\theenumi(\theenumii)}
\renewcommand\p@enumiv{\p@enumiii\theenumiii}

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.

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

\section{Itemize}

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

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 \labelsep command puts in. (See \LaTeX\ manual for an explanation.)
\setlength\tabbingsep{10pt}

\labelsep 10pt
\setlength\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; \skip\@mpfootins plays same rôle for footnotes in a minipage as \skip\footins does for ordinary footnotes.

\@mpfootins = \@skip

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.

\fboxsep = 5\pct\, \fboxrule = .6\pct

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

\theequation = \@arabic\c@equation

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

\jot = 3\pt

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

\@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 default font first, to get \LaTeX\2.09 behaviour.

\sf

\bf The command to change to the bold series. One should use \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.

$\texttt{\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}}$
$\texttt{\DeclareOldFontCommand{\sl}{\normalfont\slshape}{\relax}}$
$\texttt{\DeclareOldFontCommand{\sc}{\normalfont\scshape}{\relax}}$

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’.

$\texttt{\DeclareRobustCommand*{\cal}{\@fontswitch{\relax}{\mathcal}}}$
$\texttt{\DeclareRobustCommand*{\mit}{\@fontswitch{\relax}{\mathnormal}}}$

### 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.

$\texttt{\renewcommand{\footnoterule}{}}$
$\texttt{% \let \footnoterule = \relax}$

Footnotes are numbered within slides, overlays, and notes and numbered with $^*$, $^\dagger$, etc.

$\texttt{\def{\thefootnote}{\fnsymbol{footnote}}}$
$\texttt{\@addtoreset{footnote}{slide}}$
$\texttt{\@addtoreset{footnote}{overlay}}$
$\texttt{\@addtoreset{footnote}{note}}$

The footnote mechanism of \TeX 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
$\texttt{\hsize = \texttt{columnwidth}}$).

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

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

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
The mark is flushed right 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.

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

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.

```latex
%\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`.

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

### 10 Initialisation

#### 10.1 Date

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

```latex
\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

```latex
\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

```latex
\(*\cmd\)
```
FMi: I don’t see any reason for this command since \fi is hidden anyway in the replacement text. \def\@xfi{\fi}

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

Counters
- slide slide number
- overlay overlay number for a slide
- note note number for a slide

Add these counters explicitly to the ‘ckpt list’ so that the \include mechanism works.

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\textit{n} (n>0)
Where RANGE\textit{i} = j or j-k.

Then \setlimits globally sets
(i) \LIST := RANGE2, ... , RANGE\textit{n}
(ii) \LOW := p
(iii) \HIGH := q
where either RANGE1 = p-q or RANGE1 = p and q=p.

\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
\setupcounters ::=  (similar to old \blackandwhite #1 ::= )
\newpage
page counter := 0
@bw := T
@visible := T
if @onlyslidesw = true
  then \doslidelist := \dogslidelist
    \setlimits\doslidelist\doslidelow\doslidehigh
fi
if @onlynotesw = true
  then \donotelist := \dognotelist
    \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{\dogslidelist}\fi
  \if@onlynotesw \xdef\donotelist{\dognotelist}\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\colorslides{FILE} ::= 
\newpage
page counter := 0
@bw := F
for \@currcolor := \@colorlist
  do \@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
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}
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
\ifnum \c@slide > \@doslidehigh \relax
\if\$@setlimits \@doslidelist \@doslideelow \@doslidehigh \$\fi\num
\c@slide \!< \@doslideelow \relax \G@slideswfalse \fi\fi
\ifG@slidesw
\G@slideswfalse
\%
\begingroup
\if@bw \G@slideswtrue \else
\if@visible \G@slideswtrue \fi
\fi
\%
\endgroup
\fi
\%
\ifG@slidesw \\makingslidestrue \thispagestyle{slide} \%
\fi
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
\fi
\def\endslide{\par\break}

\overlay COLORS ::= BEGIN
\par\break
\stepcounter{overlay}
\setcounter{page}{0} \%
\ifnum @slidesw = G T \%
\if@onlyslidesw = T \%
\if \c@slide < \@doslideelow \%
\if\rels\doslidelow \relax \G@slideswfalse \fi\fi
\fi
\if \@slidesw = G F
\fi
\fi
\if \@slidesw = G T
\endgroup
\begin{group}
\if@bw = true
\then \G@slidesw = G T
\else \@color\{COLORS\}
\fi\visible then \G@slidesw = G T \fi
\endgroup
\fi
\if@slidesw = T
  \then\@makingslides := T
  \thispagestyle{overlay}
  \else\end{overlay}\@gobbletoend{overlay}\fi
\fi
\END

\endoverlay ::=\BEGIN\par\break\END

609 \if@compatibility
610 \def\overlay#1{\stepcounter{overlay}\G@slideswtrue%
611 \if@onlyslidesw\ifnum \c@slide <\@doslide\relax
612 \G@slideswfalse\fi\fi
613 \ifG@slidesw \G@slideswfalse\begingroup\if@bw\G@slideswtrue%
614 \else\@color{#1}\if@visible \G@slideswtrue\fi\fi\endgroup\fi
615 \ifG@slidesw \newpage\thispagestyle{overlay}\%
616 \else\end{overlay}\@gobbletoend{overlay}\fi}
617 \else \%
618 \else \%
619 \def\overlay{\par\break
620 \setcounter{page}{0} % in case of non-slide pages
621 \if@bw = T
622 then\slidestrue
623 \setcounter{page}{0} % case of non-slide pages
624 \if@onlynotesw = true % set \@notesw = T iff
625 \then % page to be output
626 while \c@slide > \@donotehigh
627 do \setlimits\@notelist\@notelow\@donotehigh od
628 \fi \%
629 \end{overlay}\fi
630 \ifG@slidesw \@makingslidestrue\thispagestyle{overlay}\%
631 \else\end{overlay}\@gobbletoend{overlay}\fi}
632 \fi \%
633 \def\endoverlay{\par\break}

\note ::=\BEGIN\par\break\stepcounter{note}\setcounter{page}{0}
\if@slidesw = T
then
\slidestrue
\setcounter{page}{0} % case of non-slide pages
\if@onlynotesw = true % set \@notesw = T iff
\then % page to be output
while \c@slide > \@donotehigh
\do \setlimits\@notelist\@notelow\@donotehigh od

27
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

\def\note{\stepcounter{note}\
  if@compatibility
  \def\note{%
  if\@bw
    \G@slideswtrue
    \if@onlynotesw\@whilenum \c@slide >\@donotehigh\relax
    \do{\@setlimits\@donotelist\@donotelow\@donotehigh}\ifnum
    \c@slide <\@donotelow\relax \G@slideswfalse\fi\fi
    \else\G@slideswfalse\fi
    \ifG@slidesw \newpage\thispagestyle{note}\else
    \end{note}\@gobbletoend{note}\fi%
  \else %if@compatibility
  %
  \def\note{\par\break\stepcounter{note}\setcounter{page}{0}\
  if\@bw
    \G@slideswtrue
    \if@onlynotesw\@whilenum \c@slide >\@donotehigh\relax
    \do{\@setlimits\@donotelist\@donotelow\@donotehigh}\ifnum
    \c@slide <\@donotelow\relax \G@slideswfalse\fi\fi
    \else\G@slideswfalse\fi
    \ifG@slidesw \@makingslidestrue\thispagestyle{note}\else
    \end{note}\@gobbletoend{note}\fi%
  \fi %if@compatibility
  \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
FMi: \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.

\def\fontshape{\expandafter\@gobble\f@shape}\selectfont
\mathversion{normal}%
\fi
\ignorespaces}

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

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}

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.
10.3.1 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@
\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

\def\dashbox#1(#2,#3){%
\advance\@dashcnt \@ne \divide\@dashcnt \tw@ \\
else
\divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \@ne
\setbox\@dashbox\hbox{
\hskip -\@halfwidth \vrule \@width \@wholewidth 
\@height \@dashdim}
\put(0,0){\copy\@dashbox}\
\put(#2,0){\copy\@dashbox}\
\put(0,#3){\lower\@dashdim\copy\@dashbox}\
\put(#2,#3){\lower\@dashdim\copy\@dashbox}\
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox\hbox{\vrule \@width \@wholewidth 
\@height #1\unitlength}
\@tempcnta\z@
\put(0,0){\hskip -\@halfwidth \vbox{
\@whilenum \@tempcnta < \@dashcnt \do{
\vskip #1\unitlength \copy\@dashbox \advance\@tempcnta \@ne }\
\vskip\@dashdim}}\@tempcnta\z@
\put(#2,0){\hskip -\@halfwidth \vbox{
\@whilenum \@tempcnta < \@dashcnt \relax \do{
\vskip #1\unitlength \copy\@dashbox \advance\@tempcnta \@ne }\
\vskip\@dashdim}}
\fi \@makepicbox(#2,#3)

(re)declare these booleans as they not defined in old format (or with latexrelease package)
\newif\if@ovvline \@ovvlinetrue
\newif\if@ovhline \@ovhlinetrue
\def\@oval(#1,#2)[#3][]{
\if@visible \begingroup 
\boxmaxdepth \maxdimen
\@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
\@ovvlinefalse \@ovhlinefalse
\@tfor\reserved@a :=#3\do
{\csname @ov\reserved@a false\endcsname}\n\@ovxx#1\unitlength \@ovyy#2\unitlength
\@tempdimb \ifdim \@ovyy > \@ovxx \@ovxx \@ovyy \fi 
\ifdim \@ovxx > \@ovyy \@ovyy \@ovxx \else \@ovvlinetrue \fi \fi
\@circlem\@docommand \@ovx \@ovy\@ovxx \@ovyy \@ovvlinetrue
\else \@ovx \@docommand \@ovy \@ovxx \@ovvlinetrue \fi \fi
\advance \@tempdimb - \@p\fi
\@getcirc \@tempdimb
\@circlem \@ht \@tempboxa \@ovth \@dp \@tempboxa
\@ovdx \@ovxx \@advance \@ovdx - \@tempdimb \@divide \@ovdx \tw@
\@ovdy \@ovyy \@advance \@ovdy - \@tempdimb \@divide \@ovdy \tw@
\ifdim \@ovdx > \@z\@ovvlinetrue \fi
\ifdim \@ovdy > \@z \@ovvlinetrue \fi
10.3.2 Other modifications to \LaTeX \ and \TeX \ commands

% \_ (Added 10 Nov 86)
\def\_{\leavevmode \kern.06em \if@visible\vbox{\hrule \@width.3em}\else
\vbox{\hrule \@height \z@ \@width.3em} \vbox{\hrule \@width \z@}\fi}

\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@\z@ \dp\z@\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
\let\@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\let\@@overline=\overline
\def\overline#1{\mathchoice{\@xoverline\displaystyle{#1}}{\@xoverline\textstyle{#1}}{\@xoverline\scriptstyle{#1}}{\@xoverline\scriptscriptstyle{#1}}}
\def\frac#1#2{% 
\@mathbox#1\@smashboxc{{\begingroup#2\endgroup\over\null}}% 
\setbox\tw@\null 
\ht\tw@ \ht\@smashboxc 
\dp\tw@ \dp\@smashboxc 
\wd\tw@ \wd\@smashboxc 
\box\if@visible\@smashboxcelse\tw@fi}
10.3.3 Changes to \texttt{\LaTeX} output routine

\begin{verbatim}
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
ifvoid \insert\footins
  then \@outputbox := \box255
  else \@outputbox := \vbox {\unvbox255
                          \vskip \skip\footins
                          \footnoterule
                          \unvbox\@footinsert
                      }
fi
\@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.
\10.3.4 Special \LaTeX\ initializations
FMi why not allow for ref’s ?
\end{verbatim}
\% \nofiles
\@visibletrue
(/cmd)