Slides and Course Notes*

Michael Kohlhase
FAU Erlangen-Nürnberg
http://kwarc.info/kohlhase

2021-12-08

We present a document class from which we can generate both course slides and course notes in a transparent way.

1 Introduction

The `mikoslides` document class is derived from `beamer.cls` [Tana], it adds a "notes version" for course notes derived from the `omdoc` class [**Kohlhase:smomdl**] that is more suited to printing than the one supplied by `beamer.cls`.

2 The User Interface

The `mikoslides` class takes the notion of a slide frame from Till Tantau's excellent `beamer` class and adapts its notion of frames for use in the STEXand OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `mikoslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

2.1 Package Options

EdN:1

The `mikoslides` class takes a variety of class options:[1]

`slides` `notes`
- The options `slides` and `notes` switch between slides mode and notes mode (see Section 2.2).

`sectocframes`
- If the option `sectocframes` is given, then for the `omgroups`, special frames with the `omgroup` title (and number) are generated.

`showmeta`
- `showmeta`. If this is set, then the metadata keys are shown (see [Koh20] for details and customization options).

*Version ? (last revised ?)
[1] EdNote: leaving out noproblems for the moment until we decide what to do with it.

frameimages
fiboxed

- If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames. If also the `fiboxed` option is given, the slides are surrounded by a box.

topsect

- `topsect=`⟨*sect*⟩ can be used to specify the top-level sectioning level; the default for ⟨*sect*⟩ is `section`.

2.2 Notes and Slides

frame
note

Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details. The `mikoslides` class adds the `note` environment for encapsulating the course note fragments.[1]

⚠ Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else LaTeX becomes confused and throws error messages that are difficult to decipher.

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 1: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 1.

\ifnotes

Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notestrue` or `\notesfalse` is strongly discouraged however.

⚠: We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

⚠: The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `mikoslides`

[1] MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive LaTeX trickery. Hints to the author are welcome.

case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add STEXnotes. In this case we can use `\frameimage[`⟨*opt*⟩`]{`⟨*path*⟩`}`, where ⟨*opt*⟩ are the options of `\includegraphics` from the `graphicx` package [CR99] and ⟨*path*⟩ is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.[2]

\frameimage

EdN:2

If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

\inputref*

There are some environments that tend to occur at the top-level of `note` environments. We make convenience versions of these: e.g. the `nomtext` environment is just an `omtext` inside a `note` environment (but looks nicer in the source, since it avoids one level of source indenting). Similarly, we have the `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

nomtext
nomgroup
ndefinition
nexample
nsproof
nassertion

2.3 Header and Footer Lines of the Slides

The default logo provided by the `mikoslides` package is the STEX logo it can be customized using `\setslidelogo{`⟨*logo name*⟩`}`.

\setslidelogo

The default footer line of the `mikoslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `mikoslides` package since he is the main user and designer of this package. `\setsource{`⟨*name*⟩`}` can change the writer's name. For licensing, we use the Creative Commons Attribuition-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

\setsource

\setlicensing

2.4 Colors and Highlighting

The `\textwarning` macro generates a warning sign: ⚠

\textwarning

2.5 Front Matter, Titles, etc.

2.6 Excursion

In course notes, we sometimes want to point to an "excursion" – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```
\excursion{founif}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

The `\excursion{`⟨*ref*⟩`}{`⟨*path*⟩`}{`⟨*text*⟩`}` is syntactic sugar for

\excursion

\activateexcursion

[2] EdNote: MK: the hyperref link does not seem to work yet. I wonder why but do not have the time to fix it.

```
\begin{nomtext}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nomtext}
```

\activateexcursion
\printexcursions

where `\activateexcursion{`⟨*path*⟩`}` augments the `\printexcursions` macro by a call `\inputref{`⟨*path*⟩`}`. In this way, the3 `\printexcursions` macro (usually in the appendix) will collect up all excursions that are specified in the main text.

\excursionref

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{`⟨*label*⟩`}` for that.

\excursiongroup

Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=`⟨*id*⟩`,intro=`⟨*path*⟩`]` is equivalent to

```
\begin{omgroup}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{omgroup}
```

2.7 Miscellaneous

3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeXGitHub repository [sTeX].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying `omdoc` package.

4 MiKoSlides – Implementation

4.1 Class and Package Options

We define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `mikoslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
1  ⟨*cls⟩
2  %\RequirePackage{modules}
3  \RequirePackage{kvoptions}
4  \RequirePackage{stex-metakeys}
5  \RequirePackage{etoolbox}
6  \SetupKeyvalOptions{family=mks@cls,prefix=mks@cls@}
7  \DeclareStringOption[article]{class}
8  \AddToKeyvalOption*{class}{\PassOptionsToClass{class=\mks@cls@class}{omdoc}
9    \ifdefstring{\mks@cls@class}{book}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}}{
10   \ifdefstring{\mks@cls@class}{report}{\PassOptionsToPackage{defaulttopsect=part}{mikoslides}
```

```
\DeclareBoolOption{notes}
\DeclareComplementaryOption{slides}{notes}
\DeclareDefaultOption{%
  \PassOptionsToClass{\CurrentOption}{omdoc}
  \PassOptionsToClass{\CurrentOption}{beamer}
  \PassOptionsToPackage{\CurrentOption}{mikoslides}}
\ProcessKeyvalOptions{mks@cls}
⟨/cls⟩
```

now we do the same for the `mikoslides` package.

```
⟨*package⟩
%\RequirePackage{stex-base}
\RequirePackage{kvoptions}
\RequirePackage{stex-metakeys}
\SetupKeyvalOptions{family=mks@sty,prefix=mks@sty@}
\DeclareStringOption{topsect}
\DeclareStringOption{defaulttopsect}
\newif\ifnotes\notestrue
\DeclareBoolOption{notes}
\AddToKeyvalOption*{notes}{\notestrue}%\PassOptionsToPackage{notes}{statements}}
\DeclareComplementaryOption{slides}{notes}
\AddToKeyvalOption*{slides}{\notesfalse}%\PassOptionsToPackage{nontheorem}{statements}}
\DeclareBoolOption{sectocframes}
\DeclareBoolOption{frameimages}
\DeclareBoolOption{fiboxed}
\DeclareBoolOption{noproblems}
%\DeclareDefaultOption{
  %\PassOptionsToPackage{\CurrentOption}{stex}
  %\PassOptionsToPackage{\CurrentOption}{smglom}
  %\PassOptionsToPackage{\CurrentOption}{tikzinput}}
\ProcessKeyvalOptions{mks@sty}
```

we give ourselves a macro `\@@topsect` that needs only be evaluated once, so that the `\ifdefstring` conditionals work below.

```
\ifx\mks@sty@topsect\@empty\edef\@@topsect{\mks@sty@defaulttopsect}
\else\edef\@@topsect{\mks@sty@topsect}\fi
⟨/package⟩
```

Depending on the options, we either load the `article`-based `omdoc` or the `beamer` class (and set some counters).

```
⟨*cls⟩
\ifmks@cls@notes
  \LoadClass{omdoc}
\else
  \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
  \newcounter{Item}
  \newcounter{paragraph}
  \newcounter{subparagraph}
  \newcounter{Hfootnote}
\fi
```

now it only remains to load the `mikoslides` package that does all the rest.

```
\RequirePackage{mikoslides}
⟨/cls⟩
```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the STEX packages. The first batch of packages we want are loaded on `mikoslides.sty`. These are the general ones, we will load the STEX-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
⟨*package⟩
%\RequirePackage{stex}
\RequirePackage{stex-compatibility}
\ifmks@sty@notes
\RequirePackage{a4wide}
\RequirePackage{marginnote}
\RequirePackage[dvipsnames,svgnames]{xcolor}
\RequirePackage{mdframed}
\RequirePackage[noxcolor,noamsthm]{beamerarticle}
\fi
\RequirePackage{etoolbox}
\RequirePackage{amssymb}
\RequirePackage{amsmath}
\RequirePackage{comment}
\RequirePackage{textcomp}
\RequirePackage{url}
\RequirePackage{graphicx}
\RequirePackage{pgf}
%\RequirePackage{omtext}
\ifmks@sty@notes
\RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
\fi
```

finally, we require the `metakeys` package from STEX, so that we can use the `\addmetakey` mechanism.

```
%\RequirePackage{metakeys}
```

4.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme`⟨*theme*⟩.sty, the notes version loads `beamernotestheme`⟨*theme*⟩`.sty`.[3]

EdN:3

```
\ifmks@sty@notes
\renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
\fi
```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```
\newcounter{slide}
\newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
\newlength{\slideheight}\setlength{\slideheight}{9cm}
```

`note` The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a

[3] EdNote: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```
\ifmks@sty@notes%
  \renewenvironment{note}{\ignorespaces}{}%
\else%
  \excludecomment{note}%
\fi%
```

We first set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
\ifmks@sty@notes
  \newlength{\slideframewidth}
  \setlength{\slideframewidth}{1.5pt}
```

`frame` We first define the keys.

```
  \addmetakey{frame}{label}
  \addmetakey[yes]{frame}{allowframebreaks}
  \addmetakey{frame}{allowdisplaybreaks}
  \addmetakey[yes]{frame}{fragile}
  \addmetakey[yes]{frame}{shrink}
  \addmetakey[yes]{frame}{squeeze}
  \addmetakey[yes]{frame}{t}
```

We define the environment, read them, and construct the slide number and label.

```
  \renewenvironment{frame}[1][]{%
    \metasetkeys{frame}{#1}%
    \sffamily%
    \stepcounter{slide}%
    \def\@currentlabel{\theslide}%
    \ifx\frame@label\@empty\else\label{\frame@label}\fi%
```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```
    \def\itemize@level{outer}%
    \def\itemize@outer{outer}%
    \def\itemize@inner{inner}%
    \renewcommand\newpage{\addtocounter{framenumber}{1}}%
    \renewcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}%
    \renewenvironment{itemize}{%
      \ifx\itemize@level\itemize@outer%
        \def\itemize@label{$\rhd$}%
      \fi%
      \ifx\itemize@level\itemize@inner%
        \def\itemize@label{$\scriptstyle\rhd$}%
      \fi%
      \begin{list}%
      {\itemize@label}%
      {\setlength{\labelsep}{.3em}%
       \setlength{\labelwidth}{.5em}%
       \setlength{\leftmargin}{1.5em}%
      }%
      \edef\itemize@level{\itemize@inner}%
    }{%
      \end{list}%
    }%
```

We create the box with the `mdframed` environment from the equinymous package.

```
    \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=
  }{%
    \medskip\miko@slidelabel\end{mdframed}%
  }%
```

Now, we need to redefine the frametitle (we are still in course notes mode).

`\frametitle`

```
  \renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}\medskip}%
\fi %ifmks@sty@notes
```

(*End definition for* `\frametitle`*. This function is documented on page* ??.)

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package. We also add the `label` key.

```
\define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
\newrobustcmd\frameimage[2][]{%
  \stepcounter{slide}%
  \ifmks@sty@frameimages%
    \def\Gin@ewidth{}\setkeys{Gin}{#1}%
    \ifmks@sty@notes\else\vfill\fi%
    \begin{center}
      \ifmks@sty@fiboxed%
        \fbox{\ifx\Gin@ewidth\@empty\includegraphics[width=\slidewidth,#1]{#2}\else\mygraphic
      \else
        \ifx\Gin@ewidth\@empty\includegraphics[width=\slidewidth,#1]{#2}\else\mygraphics[#1]{
      \fi% ifmks@sty@fiboxed
     \end{center}
    \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
    \ifmks@sty@notes\else\vfill\fi%
  \fi} % ifmks@sty@frameimages
```

(*End definition for* `\frameimage`*. This function is documented on page* ??.)

EdN:4 `\pause` [4]

```
\ifmks@sty@notes\newcommand\pause{}\fi
```

(*End definition for* `\pause`*. This function is documented on page* ??.)

`nomtext`

```
\ifmks@sty@notes\newenvironment{nomtext}[1][]{\begin{omtext}[#1]}{\end{omtext}}%
\else\excludecomment{nomtext}\fi%
```

`nomgroup`

```
\ifmks@sty@notes\newenvironment{nomgroup}[2][]{\begin{omgroup}[#1]{#2}}{\end{omgroup}}%
\else\excludecomment{nomgroup}\fi%
```

`ndefinition`

```
\ifmks@sty@notes\newenvironment{ndefinition}[1][]{\begin{definition}[#1]}{\end{definition}}%
\else\excludecomment{ndefinition}\fi%
```

[4] EdNote: MK: fake it in notes mode for now

nassertion

```
\ifmks@sty@notes\newenvironment{nassertion}[1][]{\begin{assertion}[#1]}{\end{assertion}}%
\else\excludecomment{nassertion}\fi%
```

nsproof

```
\ifmks@sty@notes\newenvironment{nsproof}[2][]{\begin{sproof}[#1]{#2}}{\end{sproof}}%
\else\excludecomment{nsproof}\fi%
```

nexample

```
\ifmks@sty@notes\newenvironment{nexample}[1][]{\begin{example}[#1]}{\end{example}}%
\else\excludecomment{nexample}\fi%
```

\inputref@*skip We customize the hooks for in \inputref.

```
\def\inputref@preskip{\smallskip}
\def\inputref@postskip{\medskip}
```

(*End definition for* \inputref@*skip*. *This function is documented on page* ??.)

\inputref*

```
\let\orig@inputref\inputref
\def\inputref{\@ifstar\ninputref\orig@inputref}
\newcommand\ninputref[2][]{\ifmks@sty@notes\orig@inputref[#1]{#2}\fi}
```

(*End definition for* \inputref*. *This function is documented on page* ??.)

4.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo The default logo is the sTEX logo. Customization can be done by \setslidelogo{⟨*logo name*⟩}.

```
\newlength{\slidelogoheight}
\ifmks@sty@notes%
  \setlength{\slidelogoheight}{.4cm}%
\else%
  \setlength{\slidelogoheight}{1cm}%
\fi%
\newsavebox{\slidelogo}%
\sbox{\slidelogo}{\sTeX}%
\newrobustcmd{\setslidelogo}[1]{%
  \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}%
}%
```

(*End definition for* \setslidelogo. *This function is documented on page* ??.)

\setsource \source stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. \setsource{⟨*name*⟩} can change the writer's name.

```
\def\source{Michael Kohlhase}% customize locally
\newrobustcmd{\setsource}[1]{\def\source{#1}}%
```

(*End definition for* \setsource. *This function is documented on page* ??.)

\setlicensing Now, we set up the copyright and licensing. By default we use the Creative Commons Attribuition-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[`⟨*url*⟩`]{`⟨*logo name*⟩`}` is used for customization, where ⟨*url*⟩ is optional.

```
\def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}%
\newsavebox{\cclogo}%
\sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}%
\newif\ifcchref\cchreffalse%
\AtBeginDocument{%
  \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
}%
\def\licensing{%
  \ifcchref%
    \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}%
  \else%
    {\usebox{\cclogo}}%
  \fi%
}%
\newrobustcmd{\setlicensing}[2][]{%
  \def\@url{#1}%
  \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}%
  \ifx\@url\@empty%
    \def\licensing{{\usebox{\cclogo}}}%
  \else%
    \def\licensing{%
      \ifcchref%
      \href{#1}{\usebox{\cclogo}}%
      \else%
      {\usebox{\cclogo}}%
      \fi%
    }%
  \fi%
}%
```

(*End definition for* `\setlicensing`*. This function is documented on page* **??**.)

EdN:5 \slidelabel Now, we set up the slide label for the `article` mode.[5]

```
\newrobustcmd\miko@slidelabel{%
  \vbox to \slidelogoheight{%
    \vss\hbox to \slidewidth%
    {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}%
  }%
}%
```

(*End definition for* `\slidelabel`*. This function is documented on page* **??**.)

4.4 Colors and Highlighting

We first specify sans serif fonts as the default.

```
\sffamily
```

[5] EdNote: see that we can use the themes for the slides some day. This is all fake.

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to to is to adapt the green so that it is dark enough for most beamers

```
\AtBeginDocument{%
\definecolor{green}{rgb}{0,.5,0}%
\definecolor{purple}{cmyk}{.3,1,0,.17}%
}%
```

We customize the `\defemph`, `\notemph`, and `\stDMemph` macros with colors for the use in the `statements` package. Furthermore we customize the `\@@lec` macro for the appearance of line end comments in `\lec`.

```
% \def\STpresent#1{\textcolor{blue}{#1}}
\def\defemph#1{{\textcolor{magenta}{#1}}}
\def\termemph#1{{\textcolor{cyan}{#1}}}
\def\notemph#1{{\textcolor{magenta}{#1}}}
\def\stDMemph#1{{\textcolor{blue}{#1}}}
\def\@@lec#1{(\textcolor{green}{#1})}
```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```
\pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
\def\smalltextwarning{%
  \pgfuseimage{miko@small@dbend}%
  \xspace%
}%
\pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
\newrobustcmd\textwarning{%
  \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}%
  \xspace%
}%
\pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}%
\newrobustcmd\bigtextwarning{%
  \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}%
  \xspace%
}%
```

(*End definition for* `\textwarning`*. This function is documented on page* **??**.)

```
\newrobustcmd\putgraphicsat[3]{%
  \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}%
}%
\newrobustcmd\putat[2]{%
  \begin{picture}(0,0)\put(#1){#2}\end{picture}%
}%
```

4.5 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```
\ifmks@sty@sectocframes%
\ifdefstring\@@topsect{part}{%
```

```
  \newcounter{chapter}\counterwithin*{section}{chapter}}
{\ifdefstring\@@topsect{chapter}{\newcounter{chapter}\counterwithin*{section}{chapter}}{}}
\fi% ifsectocframes
```

Now that we have defined the counters, we can load the STEX-specific packages (in particular `statements` that needs these counters).

```
\RequirePackage{tikzinput}
```

`\section@level` Finally, we set the `\section@level` counter that governs sectioning according to the class options. We also introduce the sectioning counters accordingly.

`\section@level`

```
\section@level=2
\def\part@prefix{}
\ifdefstring{\@@topsect}{part}
{\section@level=0%
  \def\thesection{\arabic{chapter}.\arabic{section}}%
  \def\part@prefix{\arabic{chapter}.}}{}
\ifdefstring{\@@topsect}{chapter}
{\section@level=1%
  \def\thesection{\arabic{chapter}.\arabic{section}}%
  \def\part@prefix{\arabic{chapter}.}}{}
\ifmks@sty@notes\else% only in slides
```

(*End definition for* `\section@level`. *This function is documented on page* **??**.)

The new counters are used in the `omgroup` environment that choses the LATEX sectioning macros according to `\section@level`.

`omgroup`

```
\renewenvironment{omgroup}[2][]{%
  \metasetkeys{omgroup}{#1}%
  \advance\section@level by 1%
  \advance\omgroup@level by 1%
  \ifmks@sty@sectocframes%
  \stepcounter{slide}
  \begin{frame}[noframenumbering]%
  \vfill\Large\centering%
  \red{%
    \ifcase\section@level\or
    \stepcounter{part}
    \def\@@label{\omdoc@part@kw~\Roman{part}}
    \def\currentsectionlevel{\omdoc@part@kw}
    \or%
    \stepcounter{chapter}
    \def\@@label{\omdoc@chapter@kw~\arabic{chapter}}
    \def\currentsectionlevel{\omdoc@chapter@kw}
    \or
    \stepcounter{section}
    \def\@@label{\part@prefix\arabic{section}}
    \def\currentsectionlevel{\omdoc@section@kw}
    \or
    \stepcounter{subsection}
    \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}}
    \def\currentsectionlevel{\omdoc@subsection@kw}
```

```
    \or
    \stepcounter{subsubsection}
    \def\@@label{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
    \def\currentsectionlevel{\omdoc@subsubsection@kw}
    \or
    \stepcounter{mparagraph}
    \def\@@label{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{subsubsection}.\ar
    \def\currentsectionlevel{\omdoc@paragraph@kw}
    \fi% end ifcase
    \@@label\sref@label@id\@@label
    \quad #2%
  }%
  \vfill%
  \end{frame}%
  \fi %ifmks@sty@sectocframes
  \csname stex_ref_new_doc_target:n\endcsname\omgroup@id%
}
{\advance\section@level by -1}%
\fi% ifmks@sty@notes
```

4.6 Excursions

We set up a `beamer` template for theorems like ams style, but without a block environment.

```
\def\inserttheorembodyfont{\normalfont}
\ifmks@sty@notes\else% only in slides
\defbeamertemplate{theorem begin}{miko}
{\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
  \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
  \inserttheorempunctuation\inserttheorembodyfont\xspace}
\defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```
\setbeamertemplate{theorems}[miko]
```

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```
\expandafter\def\csname Parent2\endcsname{}
\fi% ifmks@sty@notes
\ifmks@sty@notes%
  \renewenvironment{columns}[1][]{%
    \par\noindent%
    \begin{minipage}%
    \slidewidth\centering\leavevmode%
  }{%
    \end{minipage}\par\noindent%
  }%
  \newsavebox\columnbox%
  \renewenvironment<>{column}[2][]{%
    \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
  }{%
    \end{minipage}\end{lrbox}\usebox\columnbox%
  }%
\fi% ifmks@sty@notes
```

```
\ifmks@sty@noproblems%
  \newenvironment{problems}{}{}%
\else%
  \excludecomment{problems}%
\fi%
```

`\excursion*` The excursion macros are very simple, we define a new internal macro in `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```
\gdef\printexcursions{}
\newcommand\activateexcursion[2][]{\gappto\printexcursions{\inputref[#1]{#2}}}
\newcommand\excursionref[2]{% label, text
\ifnotes\begin{omtext}[title=Excursion]#2 \sref[fallback=the appendix]{#1}.\end{omtext}\fi}
\newcommand\excursion[4][]{% opt, label, path, text
  \ifnotes\activateexcursion[#1]{#3}\excursionref{#2}{#4}\fi}
```

(*End definition for* `\excursion*`*. This function is documented on page* ??.)

`\excursiongroup`

```
\srefaddidkey{excursiongroup}%
\addmetakey{excursiongroup}{intro}%
\newcommand\excursiongroup[1][]{%
  \metasetkeys{excursiongroup}{#1}%
  \ifdefempty\printexcursions{}% only if there are excursions
  {\begin{omgroup}[#1]{Excursions}%
      \ifdefempty\excursiongroup@intro{}{\inputref{\excursiongroup@intro}}%
      \printexcursions%
    \end{omgroup}}}
⟨/package⟩
```

(*End definition for* `\excursiongroup`*. This function is documented on page* ??.)

4.7 Miscellaneous