The talk Document Class*

Martin Wiebusch

August 12, 2005

Abstract

The talk document class allows you to create slides for screen presentations or printing on transparencies. It also allows you to print personal notes for your talk. You can create overlays and display structure information (current section / subsection, table of contents) on your slides. The main feature that distinguishes talk from other presentation classes like beamer or prosper is that it allows the user to define an arbitrary number of *slide styles* and switch between these styles from slide to slide. This way the slide layout can be adapted to the slide content. For example, the title or contents page of a talk can be given a slightly different layout than the other slides.

The talk class makes no restrictions on the slide design whatsoever. The entire look and feel of the presentation can be defined by the user. The style definitions should be put in a separate sty file. Currently the package comes with only one set of pre-defined slide styles (sidebars.sty). Contributions from people who are artistically more gifted than me are more than welcome!

*This file has version number 1.0, last revised 2005/08/09

Contents

1 Installation and Requirements

The talk class requires the packages amsmath, graphicx, pgf, multido and hyperref. They can all be obtained from

`http://www.ctan.org`.

To install the talk class, you have to copy the files `talk.cls` and `sidebars.sty` to a place where LaTeX can find them.

2 Using talk

The usage of the talk class is almost independent of the chosen style definitions. Therefore style definitions are most conveniently included through separate *style packages*. Currently talk comes with one such file called `sidebars.sty`. (I hope that in the future the number of available style packages will grow due to contributions from people that are artistically more gifted than I am.) The issue of writing your own style definitions is discussed in *section 3*. In this section we will see how to create a presentation using the talk class and some ready-to-use style package like `sidebars.sty`.

The general structure of a presentation `tex` file is shown in *figure 1*. Note that you can structure your talk in the usual way with `\section` and `\subsection` commands. How these commands are handled depends on the loaded style package.

2.1 Class Options

The talk class is loaded in the first line of the listing in *figure 1*:

`\documentclass[`⟨*options*⟩`]{talk}`

The available options are

`screen`, `slides`, `notes`, `rotate` and `norotate`.

The talk class is built upon the `article` class, so it will pass all unknown options to `article`. Thus, in principle, all options of the `article` class can be used with the talk class as well. However, some options like `twocolumn` etc. may lead to undesired results.

The options `screen`, `slides` and `notes` determine the *mode* in which your presentation is compiled. The options `rotate` and `norotate` only take effect in the `slides` mode.

`screen` Use the `screen` mode to create a screen presentation. With this option the paper size is set to the slide size, so that your slides can

```
\documentclass[⟨options⟩]{talk}
\usepackage{⟨style-def⟩}
⋮
(more package inclusions)
⋮
\title{⟨title⟩}
\author{⟨author⟩}
\date{⟨date⟩}
⋮
(global specifications required by ⟨style-def⟩)
⋮
\begin{document}
  \begin{slide}[⟨slide-style⟩]{⟨slide-title⟩}
    (body of first slide)
  \end{slide}
  \begin{notes}
    (notes on first slide)
  \end{notes}
  ⋮
  (more slides and notes)
  ⋮
  \section[⟨short title⟩]{⟨long title⟩}
  ⋮
  (more slides and notes)
  ⋮
  \subsection[⟨short title⟩]{⟨long title⟩}
  ⋮
  (more slides and notes)
  ⋮
  (more sections and subsections)
  ⋮
\end{document}
```

Figure 1: The general structure of a presentation `tex` file.

be displayed without white margins using the fullscreen mode of your favorite document viewer.

`slides` `rotate` `norotate` If you use the `slides` mode your presentation is prepared for printout on transparencies. The slides are centered horizontally and vertically on normal paper. The default paper size is A4, but you can change it by using any of the paper size options of the `article` class. The options `rotate` and `norotate` determine whether or not the slides are rotated by 90 degrees in counterclockwise direction. By default rotation is enabled. In `slides` mode the slides can also be magnified. You can set the magnification factor with

`\slidesmag`

`\slidesmag{`⟨*factor*⟩`}`.

`notes` The `notes` mode allows you to print personal notes for your presentation. In this mode the slides are inserted in the flowing text, between your annotations.

2.2 Style Packages and Slide Styles

Style definitions for the talk class are most conveniently included through a separate *style package*. Style packages can be included with the usual `\usepackage` command:

`\usepackage[`⟨*package-options*⟩`]{`⟨*style-package*⟩`}`.

The available package options depend on the chosen style package. The principal task of a talk style package is to set the width and height of the slides and define a number of *slide styles*. To identify the different slide styles, the style package should give them *style names* like `title`, `framed`, `sidebar`, `plain`, etc. To switch between different slide styles you can use the command

`\slidestyle`

`\slidestyle{`⟨*style-name*⟩`}`.

To change the slide style for only one slide you can pass a style name as an optional argument to the `slide` or `multislide` environment.

Command	Effect
`\backgroundcolor`	sets the colour of the slide background
`\sidebarcolor`	sets the colour of the sidebar
`\titlecolour`	sets the colour of the slide title
`\sidebartitlecolor`	sets the colour of the sidebar title
`\highlightcolor`	sets the colour of highlighted sections and subsections in the sidebar

Table 1: The colour commands of the sidebars package

`sidebars`

The `sidebars` package has one option, `compress`, which affects the way in which the table of contents is displayed in the sidebar. It defines three slide styles called `plain`, `outline` and `normal`. The `plain` style has no decorations at all and can be used for the title page or for showing large pictures. The `outline` style is designed to show the structure of your talk. Use it, for example, for the contents page. The `normal` style should be used for all other slides. It has a sidebar showing the structure of your talk. The current section is highlighted.

`\backgroundcolor` `\sidebarcolor` `\titlecolor` `\sidebartitlecolor` `\highlightcolor`

Colours can be set with the commands listed in *table 1.* their syntax is

\⟨*color-command*⟩{⟨*red-value*⟩,⟨*green-value*⟩,⟨*blue-value*⟩}

where ⟨*color-command*⟩ is one of the commands listed in *table 1* and ⟨*red-value*⟩, ⟨*green-value*⟩ and ⟨*blue-value*⟩ are numbers between 0 and 1.

2.3 Global Specifications

`\title` `\author` `\date`

Like the `article` class, talk allows you to specify the title, author and date of your talk with the commands

\title{⟨*title*⟩}
\author{⟨*author*⟩}
\date{⟨*date*⟩}

However, individual style packages may define addtitional commands, that allow you to specify additional information like institute, logo, place where the talk was given etc.

2.4 Environments

The talk class defines three environments: `slide`, `multislide` and `notes`. All typeset material in your talk should be enclosed in one of these environments.

`slide` The `slide` environment is the most important environment in the talk class. It allows you to typeset the contents of your slides. Its syntax is:

```
\begin{slide}[⟨style-name⟩]{⟨slide-title⟩}
  (slide body)
\end{slide}
```

The ⟨*style-name*⟩ argument is optional. It must be the name of one of the slide styles defined in style package you have loaded. For the `sidebars` package the available slide styles are `plain`, `outline` and `normal`. If no ⟨*style-name*⟩ argument is given, talk uses the slide style specified in the last call of the `\slidestyle` command.

`notes` The `notes` environment allows you to include annotations to your slides in the `tex` file. The contents of the `notes` environment are ignored if you compile your presentation in the `screen` or `slides` mode.

`multislide` The `multislide` environment can be used to create overlays. Its syntax is:

```
\begin{multislide}[⟨style-name⟩]{⟨sub-slides⟩}{⟨slide-title⟩}
  (slide body)
\end{multislide}
```

As for the `slide` environment the optional ⟨*style-name*⟩ argument and the ⟨*slide-title*⟩ argument specify the slide style and the slide title. The ⟨*sub-slides*⟩ argument has to be an integer number greater than zero.

It specifies the number of sub-slides, that the `multislide` environment will generate. In the body of the `multislide` environment, you can use the commands `\fromslide`, `\toslide` and `\onlyslide` to specify which material goes on which sub-slide.

`\fromslide` `\toslide` `\onlyslide`

The syntax of the commands `\fromslide`, `\toslide` and `\onlyslide` is:

```
\fromslide*{n}{⟨material⟩}
\toslide*{n}{⟨material⟩}
\onlyslide*{n}{⟨material⟩}
```

The `\fromslide*` command ignores ⟨*material*⟩ on the first $n-1$ sub-slides. The `\toslide*` command ignores ⟨*material*⟩ on all sub-slides after the n-th. The `\onlyslide*` command ignores ⟨*material*⟩ on all sub-slides exept the n-th. If you use the unstarred commands `\fromslide`, `\toslide` and `\onlyslide` the ⟨*material*⟩ is not ignored but made invisible, so that it still uses up the space (pretty much like the `\phantom` command).

2.5 Title and Contents Pages

Most talks begin with a title page showing the title of the talk, the name of the speaker and possibly additional information like the date and place where the talk is given, the institute of the speaker etc. For long talks you'll also want to show the stucture of your talk at the beginning.

`\maketitle` `\tableofcontents`

Style packages for the talk class should therefore redefine the standard LaTeX commands `\maketitle` and `\tableofcontents` in such a way that they produce suitable title and contents pages. These commands should be used in the body of a slide environment. For example, with the `sidebars` style package you would create title and contents pages by writing

```
\begin{slide}[plain]{}
  \maketitle
\end{slide}
```

and

```
\begin{slide}[outline]{Contents}
  \tableofcontents
\end{slide}
```

For some talks the table of contents may not fit on one slide. The talk class cannot break material into multiple slides automatically, but it allows you to split the table of contents manually. To do this you can pass an optional argument to the `\tableofcontents` command, which has the following form:

`\tableofcontents[`⟨*fromsec*⟩`.`⟨*fromsubsec*⟩`-`⟨*tosec*⟩`.`⟨*tosubsec*⟩`]`

where ⟨*fromsec*⟩, ⟨*fromsubsec*⟩, ⟨*tosec*⟩ and ⟨*tosubsec*⟩ are integer numbers. Their names are self-explaining. Note that the argument of `\tableofcontents` must always have the form given above. If you wan to display the sections 3 to 5 with all their subsections on one slide, you have to write

```
\tableofcontents[3.0-5.99]
```

(assuming that section 5 does not have more than 99 subsections).

3 The talk Class for Package Writers

The entire look-and-feel of a talk presentation is determined by external style packages. The macros provided by the talk class itself only take care of more technical issues like

- magnifying and positioning the slides on the paper,
- creating overlays,
- keeping a table of contents that is accessible on every slide,
- keeping a catalog of slide styles and allowing the user to switch between them.

To exploit these features, a style package writer needs to know some basic facts about the inner workings of the talk class.

3.1 Mode Conditionals

As we have seen in *section 2.1* a talk presentation can be compiled in three different modes: `slides`, `screen` and `notes`. To implement mode-specific behaviour the talk class provides the following `if` commands:

`\@ifslides` `\@ifscreen` `\@ifnotes`

```
\@ifslides{⟨if-code⟩}{⟨else-code⟩}
\@ifscreen{⟨if-code⟩}{⟨else-code⟩}
\@ifnotes{⟨if-code⟩}{⟨else-code⟩}
```

The ⟨*if-code*⟩ is executed if the talk is compiled in `slides`, `screen` or `notes` mode, respectively, and the ⟨*else-code*⟩ is executed otherwise.

3.2 Slide Dimensions

`\slidewidth` `\slideheight` `\@slidesize`

The width and height of the slides can be accessed through the `\slidewidth` and `\slideheight` commands. However, to set the slide dimensions you should always use the command

```
\@slidesize{⟨width⟩}{⟨height⟩}
```

as it also adjusts the papersize and slide positioning.

3.3 Global Specifications

`\@title` `\@author` `\@date`

The title, author and date set by the user with the `\title`, `\author` and `\date` commands are stored in the macros `\@title`, `\@author` and `\@date`. If you intend to display additional information like the speakers institute on your slides you should define an `\institute` command in analogy to the commands above:

```
\gdef\@institute{}
\newcommand{\institute}[1]{\gdef\@institute{#1}}
```

3.4 Counters

slide
subslide

In addition to the standard counters of the `article` class, talk defines the counters `slide` and `subslide`. The number of the current slide is stored in `slide`. The slides of a `multislide` environment have the same `slide` number and different `subslide` numbers.

\theslide
\thesubslide

The commands `\theslide` and `\thesubslide` can be used to print the slide or subslide number, respectively. They are defined as

```
\newcommand{\theslide}{\arabic{slide}}
\newcommand{\thesubslide}{\theslide.\arabic{subslide}}
```

You can redefine them to change the way the slide and subslide numbers are displayed

\theslidelabel

For printing labels on your slides you should use the `\theslidelabel` command. It calls `\theslide` when you are in a `slide` environment and `\thesubslide` when you are in a `multislide` environment.

3.5 Style Definitions

To change the look-and-feel of your slides you have to redefine some or all of the following commands:

```
\@makeslide
\@makeslidebackground
\@makeslidecontent
\@makenotesslide
\@maketocsection
\@maketocsubsection
```

\@newslidestyle

These commands will be called by the `slide`, `multislide` and `notes` environments. Their exact meaning will be explained later on. To create a *named* slide style, you have to wrap a `\@newslidestyle` command around your definitions. A typical style definition takes the form

```
\@newslidestyle{⟨style-name⟩}{
  \renewcommand{\@makeslide}{⟨stuff⟩}
  \renewcommand{\@makeslidebackground}{⟨stuff⟩}
  \renewcommand{\@makeslidecontent}{⟨stuff⟩}
  \renewcommand{\@makenotesslide}{⟨stuff⟩}
  \renewcommand{\@maketocsection}[3]{⟨stuff⟩}
  \renewcommand{\@maketocsubsection}[4]{⟨stuff⟩}
}
```

The `\@newslidestyle` command simply dumps its second argument into a macro called `\pres@sty@`⟨*style-name*⟩. When the style is loaded with `\slidestyle{`⟨*style-name*⟩`}` or with the optional argument of the `slide` environment, all the `\@make`... commands are reset to their default definitions. Then the macro `\pres@sty@`⟨*style-name*⟩ is executed.

Note that the `\renewcommand` calls in the second argument of `\@newslidestyle` appear in the definition of the `\pres@sty@`⟨*style-name*⟩ command. Therefore the arguments of `\@maketocsection` and `\@maketocsubsection` have to be referenced with double hashes, for example

```
\@newslidestyle{⟨style-name⟩}{
  \renewcommand{\@maketocsection}[3]{Section ##1: ##3}
}
```

Single hashes would refer to the arguments of `\pres@sty@`⟨*style-name*⟩.

3.6 Typesetting Slides

`\@slidetitle` `\@slidebody`

The `slide` and `multislide` environments store the slide title in the macro `\@slidetitle` and the slide body in the macro `\@slidebody`. When you redefine the various `\@make`... commands you can therefore use `\@slidetitle` and `\@slidebody` to insert the title and body of the current slide.

`\@makeslide`

In the `screen` or `slides` mode the `slide` and `multislide` envi-

ronments call the \@makeslide command, which should produce an LR box of width \slidewidth and height \slideheight that contains the actual slide. In the slides mode the output of \@makeslide is centered on the page and scaled by the factor specified in the last call of the \slidesmag command. If the rotate option is enabled it is also rotated by 90 degrees in the counterclockwise direction.

\@makenotesslide

If you compile in the notes mode the slide and multislide environments call \@makenotesslide to insert the current slide. By default the \@makenotesslide command simply centers the output of \@makeslide horizontally:

```
\newcommand{\@makenotesslide}{
  \par\hspace*{\fill}\@makeslide\hspace*{\fill}\par
}
```

You can change this behaviour by redefining the \@makenotesslide command. For example, if you only want to print the title of each slide in your notes, you should include something like

```
\renewcommand{\@makenotesslide}{
  \begin{center}\textbf{\@slidetitle}\end{center}
}
```

in your style definition.

To change the way the slides appear in the screen and slides mode you could redefine the \@makeslide command. However, usually you'll want to draw the slide background with some graphics package like pgf and then typeset the contents of the slide in a parbox of width \slidewidth and height \slideheight placed on top of that picture. The talk class faciliates this task by allowing you to customise the commands \@makeslidebackground and \@makeslidecontent. The default definition for \@makeslide is

\@makeslidebackground
\@makeslidecontent

```
\newcommand{\@makeslide}{
  \begin{pgfpicture}{0pt}{0pt}{0pt}{0pt}
    \@makeslidebackground
  \end{pgfpicture}
```

```
    \parbox[b][\slideheight][t]{\slidewidth}{
      \@makeslidecontent
    }
  }
```

Thus the `\@makeslidebackground` macro should expand to a series of pgf commands that draw the background of the slide. Note that the origin of the pgf coordinate system is at the lower left-hand corner of the slide. The `\@makeslidecontent` macro should expand to whatever you want to put in the parbox. Here, you start with the current point in the upper left corner of the slide.

In this way talk gives you complete artistic freedom in the design of your slides: It lets you define the macros that generate the slides while contents like the slide title and body are previously stored in macros like `\@slidetitle` and `\@slidebody`, so that you can insert them where you like. For completeness we now summarise all commands yielding user defined contents:

`\@slidetitle` Title of the current slide.

`\@slidebody` Body of the current slide.

`\theslidelabel` Label of the current slide. Shows the slide number in a `slide` environment and the slide and subslide number in a `multislide` environment.

`\@title` Title of the presentation.

`\@author` Author of the presentation.

`\@date` Date specified with the `\date` command (`\today` by default).

`\@tableofcontents` Prints the table of contents. See the next subsection for more information.

3.7 The Table of Contents

`\@tableofcontents` The table of contents of your talk can be created with the

\@tableofcontents macro. Its name is slightly misleading because, in fact, it allows you to display any kind of structure information on your slides. For example, you can use it to print only the title of the current section.

\@maketocsection
\@maketocsubsection

The \@tableofcontents macro expands to a series of \@maketocsection and \@maketocsubsection commands. By default these commands do nothing. You can control the appearence of the table of contents by redefining them. Their syntax is

\@maketocsection{⟨*section*⟩}{⟨*short-title*⟩}{⟨*long-title*⟩}

\@maketocsubsection{⟨*section*⟩}{⟨*subsection*⟩}%
{⟨*short-title*⟩}{⟨*long-title*⟩}

where ⟨*section*⟩ and ⟨*subsection*⟩ are integer numbers.

\@ifcurrentsection
\@ifcurrentsubsection

Note that the \@tableofcontents macro always expands to the full list of sections and subsections. To implement a special treatment for the *current* section or subsection you can use the \@ifcurrentsection and \@ifcurrentsubsection commands. Their syntax is

\@ifcurrentsection{⟨*number*⟩}{⟨*if-code*⟩}{⟨*else-code*⟩}
\@ifcurrentsubsection{⟨*number*⟩}{⟨*if-code*⟩}{⟨*else-code*⟩}

The ⟨*if-code*⟩ is executed if ⟨*number*⟩ matches the current section or subsection, respectively, and ⟨*else-code*⟩ is executed otherwise. For example, if you want to display the current section in the top left corner of each slide, your style definition should look somewhat like

```
\newslidestyle{⟨style-name⟩}{
  \renewcommand{\@maketocsection}[3]{
    \@ifcurrentsection{##1}{##3}{}
  }
  \renewcommand{\@makeslidecontent}{
    \@tableofcontents
    ⋮
  }
  ⋮
}
```

\tableofcontents

Most talks begin with an outline of the talk's contents. As a package writer you should therefore provide a `\tableofcontents` command that allows the user to print the full table of contents. (talk already defines the `\tableofcontents` command, but it does nothing by default.) You can achieve this, too, by redefining `\@maketocsection` and `\@maketocsubsection` and then calling `\@tableofcontents`.

\@ifinrange

However, if the table of contents does not fit on one slide, the user should be able to split it, using an optional range argument of the form shown in *section 2.5*. It is the package writers task to implement this feature, but the parsing of the range argument is done by the `\@ifinrange` macro. Its syntax is

`\@ifinrange{`⟨*sec*⟩`}{`⟨*subsec*⟩`}{`⟨*range*⟩`}{`⟨*if-code*⟩`}{`⟨*else-code*⟩`}`

⟨*sec*⟩ and ⟨*subsec*⟩ are section and subsection numbers and ⟨*range*⟩ is a string of the form

⟨*fromsec*⟩`.`⟨*fromsubsec*⟩`-`⟨*tosec*⟩`.`⟨*tosubsec*⟩

The ⟨*if-code*⟩ is executed if the subsection specified by ⟨*sec*⟩ and ⟨*subsec*⟩ lies in the range specified by ⟨*range*⟩, the ⟨*else-code*⟩ is executed otherwise.

A typical definition of the `\tableofcontents` command will therefore look as follows:

```
\renewcommand{\tableofcontents}[1][0.0-99.99]{
  \bgroup
  \def\@maketocsection##1##2##3{
    \@ifinrange{##1}{0}{#1}{
      ##1.\space ##3\par
    }{}
  }
  \def\@maketocsubsection##1##2##3##4{
    \@ifinrange{##1}{##2}{#1}{
      ##1.##2.\space ##3\par
    }{}
  }
```

```
    \@tableofcontents
    \egroup
}
```

If you use grouping (`\bgroup` and `\egroup` or curly brackets) and plain TEX definitions (`\def` instead of `\renewcommand`), as shown above, your definitions remain local to the goup, so you don't have to worry about restoring the original definitions of `\@maketocsection` and `\@maketocsubsection`. A more sophisticated definition of the `\tableofcontents` command can be found in `sidebars.sty`.

4 Contact

For comments, bug reports, feature requests or submitting style packages please email to

`martin.wiebusch@gmx.net`

Martin Wiebusch, 31.07.2005

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.