The `termcal-de` package

`https://github.com/SFr682k/termcal-de`

Sebastian Friedl
`sfr682k@t-online.de`

2018/03/23 (v2.0)

"To achieve great things, two things are needed;
a plan, and not quite enough time"
— Leonard Bernstein —

Abstract

The `termcal-de` package provides a German localization to the `termcal` package written by Bill Mitchell, which is intended to print a term calendar for use in planning a class.

Contents

Dependencies and other requirements

The `termcal-de` package requires LaTeX 2_ε and the following packages:

`termcal` the main `termcal` package

`pgfkeys`, `pgfopts` for defining key-value sets and processing them as package options

`datetime2`, `datetime2-german` `termcal-de` uses `datetime2` and its German language module, `datetime2-german`, to print the date to the calendar cells. Please ensure that *at least version 2.0* of `datetime2-german` is installed.

Installation

Extract the *package* file first:

1. Run LaTeX over the file `termcal-de.ins`
2. Move the resulting `.sty` file to `TEXMF/tex/latex/termcal-de/`

Then, you can compile the *documentation* yourself by executing

```
lualatex termcal-de-doc.dtx
makeindex -s gind.ist termcal-de-doc.idx
makeindex -s gglo.ist -o termcal-de-doc.gls termcal-de-doc.glo
lualatex termcal-de-doc.dtx
lualatex termcal-de-doc.dtx
```

or just use the precompiled documentation shipped with the source files.
In both cases, copy the files `termcal-de-doc.pdf` and `README.md` to `TEXMF/doc/latex/termcal-de/`

License

This work consists of the following files:

- `termcal-de.dtx`,
- `termcal-de.ins`,
- `termcal-de-doc.dtx`,
- `termcal-de-doc-example1.dtx`,
- `termcal-de-doc-example2.dtx`
- and the derived file `termcal-de.sty`

Part I
The documentation

1 Getting started

`termcal-de` only adds a German localization to the `termcal` package.
If you are already familiar with `termcal`, you should read section 3 about differences to plain `termcal` *in any case*.

However, if you never used `termcal`, you may …

a) read `termcal`'s documentation first and take a look at section 3 afterwards or

b) read the short tutorial on using `termcal` with `termcal-de` in section 2

1.1 Loading the package

Load `termcal-de` with `\usepackage{termcal-de}` *after loading* `babel` *or* `polyglossia`.

Example:
```
\usepackage[german]{babel}
\usepackage{termcal-de}
```

1.2 Package options

How to read this section – an example

The key-value options provided by `termcal-de` are depicted as follows:

- `metasyntacticals` . . . . . `foo`, **`bar`**, `foobar`
  The `metasyntacticals` key specifies the metasyntactical variable printed when using the `\foobar` command.

  How to interpret the first line of each description:

  1. The ***key's name*** is printed on the left hand side of the dotted line using typewriter font. In this case, the key's name is `metasyntacticals` and you can change its value using `\usepackage[metasyntacticals=…]{termcal-de}`.
  2. ***Possible values*** for this key are printed on the right hand side of the dotted line. In this case, valid key-value-specificatios would be `metasyntacticals=foo`, `metasyntacticals=bar` and `metasyntacticals=foobar`.
  3. When using a ***key without a value specified***, the `underlined` value is assumed. Therefore, in this example `\usepackage[metasyntacticals]{termcal-de}` is equal to `\usepackage[metasyntacticals=foo]{termcal-de}`.
  4. `termcal-de`'s ***default configuration set*** is composed out of the **`bold`** printed values of all keys listed here.

Provided key-value options

The following key-value options are provided for configuring `termcal-de`:

- `compat` ..... `true`, **`false`**
  When `compat`'s value is set to `true`, `termcal-de` will retain compatibility to the original `termcal` package and avoid changing the date format required by `termcal`'s commands.

- `drawdateframe` ..... `always`, `atNewMonth`, **`never`**
  This option allows to configure when a frame is drawn around the date.
  Setting `drawdateframe`'s value to `always` will draw a frame around *every* date in the calendar. Specifying `atNewMonth` will draw a frame around the date when the month has changed since the last cell. Using the `never` value will draw no frame around any date.

- `datetime2`
  This key set allows you to change the way `datetime2` is configured for printing dates to the single cells.
  Configuration is possible by changing the subkeys' values:
  `\usepackage[datetime2={local=de-DE, numeric}]{termcal-de}`

  The following subkeys are available:

  - `local` ..... **`useregional`**, `german`, `de-DE`, `de-AT`, `de-CH`
    Determines the language module used by `datetime2`.
    When `useregional` is set, the language module will be loaded according to `babel`'s or `polyglossia`'s configuration.
    Otherwise, the specified language module will be used.
  - `numeric` ..... **`true`**, `false`
    Determines whether `datetime2` uses numeric date styles.
  - `frompreamble` ..... `true`, **`false`**
    When `datetime2` is loaded and configured in your preamble, you should set this key's value to `true`. Otherwise, there will be clashing package options.

    If this key's value is `true`, the keys `local` and `numeric` will be ignored.

2 A short tutorial

This tutorial explains how to use the functionalities provided by `termcal`.
It consists of two parts: How to create a calendar grid and how to customize it.

2.1 Creating a calendar grid

The `calendar` environment

`calendar` `termcal`'s core is the `calendar` environment. It takes two arguments: the starting date and the number of weeks to be printed.
Syntax: `\begin{calendar}{<start date>}{<nr of weeks>}`

> **NOTE:**
> Plain `termcal` requires all dates to be given in the `m/d/y` format, while `termcal-de` expects all dates to be given as `D.M.YYYY` (e. g. `19.3.2018`). However, you are able to switch between both formats using the `compat` option (see section 1.2).

Specifying dates

The (week)days shown in the calendar have to be specified inside the `calendar` environment using the commands `\calday` and `\skipday`.
Both commands specify the days of the week in order, thus there should be seven of them; otherwise, your calendar will shift …

If you never used `\calday` in a `calendar` environment and try to compile your document, you will get some nasty "arithmetic overflow" errors. Anyway, who would print a calendar not containing any days …

`\skipday` `\calday` The macro `\skipday` simply declares that the corresponding day should not be printed in that calendar while the macro `\calday` is used to specify a day which is to be printed. It requires a *mandatory argument* being a (possibly empty) list of (nearly) all LaTeX commands available to be executed before printing the cell content and accepts an *optional argument* being the heading of the date column.

Available options: `\classdays`, `\noclassdays` and `\weeklytext`

`\classday` `\noclassday` The macros `\classday` and `\noclassday` declare that the specified day is, or is not, a class day. Days specified as class days are numbered and can be refered to by their numbers.

Setting `\noclassday` may be omitted as long as you don't have to override a `\classday` specified for the whole column.

`\weeklytext` Also, weekly text can be added by using the `\weeklytext` command inside a column declaration; you may use arbitrary LaTeX code (e. g. `\weeklytext{foo \\ bar}`)

Example: A simple calendar

This example only demonstrates how to use the `calendar` environment and specify some days. See figure 1 for the resulting output.
Further customization of the calendar grid is described in section 2.2.
As the lecturer is a certain "Garfield" the weekday name "Monday" has been censored.

```
% \usepackage{termcal-de}
\begin{calendar}{10.12.2012}{3}
    \calday[*!@\$\#+]{\classday}
    \calday[Tuesday]{\weeklytext{It's Tuesday. \\ *!@\$\#+'s over!}}
    \skipday
    \calday[Thursday]{}
    \calday[Friday]{\classday}
    \skipday
    \skipday
\end{calendar}
```

*!@$#+	TUESDAY	THURSDAY	FRIDAY
10.12.2012 **1**	11.12.2012 It's Tuesday. *!@$#+'s over!	13.12.2012	14.12.2012 **2**
17.12.2012 **3**	18.12.2012 It's Tuesday. *!@$#+'s over!	20.12.2012	21.12.2012 **4**
24.12.2012 **5**	25.12.2012 It's Tuesday. *!@$#+'s over!	27.12.2012	28.12.2012 **6**

Figure 1: Output of the example shown in section 2.1

2.2 Customizing the calendar grid

The output of this example shown above is kind of "primitive": a calender grid is existent, but the text for (nearly all) boxes is missing. Also, one would like to override the general options for some specific dates.

This section of the tutorial describes how to …

- resize the calendar,
- add text to single dates,
- add text to consecutive class days and
- override the column options for specific dates

Resizing the calendar

`termcal` provides two lengths influencing the size of the calendar and its boxes:
`\calwidth` `\calwidth` representing the total width of the calendar and `\calboxdepth` determining
`\calboxdepth` the minimum height of the boxes for each day.

They may be set to other values using the `\setlength` command, e. g.:
`\setlength{\calwidth}{.8\textwidth}` and
`\setlength{\calboxdepth}{1.25cm}`

Adding text to single dates

Changing the size of the grid doesn't do anything to the fact that we still have a grid – without any content but the date and a quite generic weekly text. However, one would certainly like to add specific content for specific dates.

`\caltext` `termcal`'s `\caltext` command requires two arguments: *when* the text should be printed, and – obviously – the actual *text* to be printed.
There are two possibilities to specify the date or class where text should be printed: either by the date or by the class number, for example
`\caltext{24.12.2012}{Christmas Eve \\ No class}` using the date and
`\caltext{C1}{First Class \\ Organisational matters}` using the class number.

> **ATTENTION!!**
> The date format *has* to be `D.M.YYYY` (or `m/d/y` when using the `compat` option).
> This means that the *date specifications must not contain leading zeros.*
>
> **Examples:** Use …
>
> | | | | | | | |
> |---|---|---|---|---|---|---|
> | `5.1.2016` | | `1/5/16` | | `05.01.2016` | | `01/05/16` |
> | `9.11.2019` | or | `11/9/19` | instead of | `09.11.2019` | or | `11/09/19` |
> | `14.3.2018` | | `3/14/18` | | `14.03.2018` | | `03/14/18` |

Adding text to consecutive class days

However, the `\caltext` command described above is not the best way to add text to consecutive class days. As a lecturer, you might want to prepone a certain topic – and it's quite uncomfortable to change every single `C…` specification used in *any* `\caltext` command.

`\caltexton` Therefore, `termcal` provides the commands `\caltexton` and `\caltextnext`.
`\caltextnext` Specify the starting day of the series (as class number) and the text shown there using

the \caltexton command. Then, you are able to add content to the successive class days using \caltextnext. Use \caltextnext with an empty argument for skipping class days.

The following example shows such a simple series:

```
\caltexton{2}{Introduction to metasyntactical variables}
\caltextnext{}% skip next class day
\caltextnext{foo and bar}
```

Override column options for specific dates

Last but not least, we have to override the "global" column options for certain dates.

\options For specifying options applying to a specific day, the \options command is defined, which requires a date specification (like \caltext) and a list of option (like \calday). Options added by \options are executed after options specified for \calday and may therefore be used to override the specification set of a date column.
Weekly text may be suppressed by using \options together with \weeklytext{}.

Some examples:

```
\options{18.12.2012}{\classday\weeklytext{}}
\options{20.12.2012}{\classday}
\options{21.12.2012}{\noclassday}
```

Remember: The date specifications may *not* contain any leading zeros!

Example: A customized calendar

This is an enhanced version of the example shown in section 2.1. Cell text has been added, options were changed for specific days and the cell depth is smaller. See figure 2 for the resulting output.

```
% \usepackage{termcal-de}
\begin{calendar}{10.12.2012}{3}
    \setlength{\calwidth}{.95\textwidth}
    \setlength{\calboxdepth}{1.25cm}

    \calday[*!@\$\#+]{\classday}
    \calday[Tuesday]{\weeklytext{It's Tuesday. \\ *!@\$\#+'s over!}}
    \skipday
    \calday[Thursday]{}
    \calday[Friday]{\classday}
    \skipday
    \skipday

    \options{18.12.2012}{\classday\weeklytext{}}
    \options{20.12.2012}{\classday}

    \options{21.12.2012}{\noclassday}
    \caltext{21.12.2012}{\textbf{Doomsday} \\ No class}

    \options{24.12.2012}{\noclassday}
    \caltext{24.12.2012}{Christmas Eve \\ No class}
```

```
    \caltext{C1}{First Class \\ Organisational matters}

    \caltexton{2}{Introduction to metasyntactical variables}
    \caltextnext{}
    \caltextnext{"bla"/"blub" vs. "foo"/"bar"}
    \caltextnext{"08/15", "42" and the mysterious "237"}
    \caltextnext{Coffee break}
\end{calendar}
```

*!@$#+	Tuesday	Thursday	Friday
10.12.2012 **1** First Class Organisational matters	11.12.2012 It's Tuesday. *!@$#+'s over!	13.12.2012	14.12.2012 **2** Introduction to metasyntactical variables
17.12.2012 **3**	18.12.2012 **4** `bla/blub` vs. `foo/bar`	20.12.2012 **5** `08/15`, `42` and the mysterious `237`	21.12.2012 **Doomsday** No class
24.12.2012 Christmas Eve No class	25.12.2012 It's Tuesday. *!@$#+'s over!	27.12.2012	28.12.2012 **6** Coffee break

Figure 2: Output of the example shown in section 2.2

3 Differences to plain `termcal`

NOTE:
This section only applies until the `compat` *option (see section 1.2) is given.*
As soon as you pass it to `termcal-de`, the date specification required by all commands stays — as in plain `termcal` itself — `m/d/y`.

When using the standard configuration `termcal-de`, does not only change the format of the printed dates, it also changes the date parameter's format expected by `termcal`'s standard commands.
More precisely, these commands are affected:

- `\begin{calendar}{<starting date>}{<nr of weeks>}`
- `\options{<date>}{<option list>}`
- `\caltext{<date>}{<text>}`

Plain `termcal` expects `<starting date>` and `<date>` to be given in the `m/d/y` format (e. g. `1/10/18` for January 10, 2018). Due to redefinition in `termcal-de`, both arguments, `<starting date>` and `<date>` have to be given in the `D.M.YYYY` format (for January 10, 2018: `10.1.2018`).
See table 3 for some examples.

plain `termcal`	**with** `termcal-de` **package**
`\begin{calendar}{1/10/18}{4}`	`\begin{calendar}{10.1.2018}{4}`
`\options{12/21/12}{\noclass}`	`\options{21.12.2012}{\noclass}`
`\caltext{2/7/11}{Exam}`	`\caltext{7.2.2011}{Exam}`

Table 3: Comparison between plain `termcal` and `termcal` extended with `termcal-de`

ATTENTION!!
The date format *has* to be `D.M.YYYY` (or `m/d/y` when using the `compat` option).
This means that the *date specifications must not contain leading zeros.*

Examples: Use …

`5.1.2016`		`1/5/16`		`05.01.2016`		`01/05/16`
`9.11.2019`	or	`11/9/19`	instead of	`09.11.2019`	or	`11/09/19`
`14.3.2018`		`3/14/18`		`14.03.2018`		`03/14/18`

Part II
The package code

Initialize

Identify the package and require LaTeX 2ε

```
\ProvidesPackage{termcal-de}[2018/03/23 v2.0 German locals to the termcal package]
\NeedsTeXFormat{LaTeX2e}
```

Require a basic set of packages

Require the “original” `termcal` package

```
\RequirePackage{termcal}
```

Require packages providing the key-value option stuff

```
\RequirePackage{pgfkeys}
\RequirePackage{pgfopts}
```

Define options

Define variables:

```
\newif\if@termcalde@compat
\newif\if@termcalde@drawbox
\newif\if@termcalde@dtmconf@frompreamble
\newif\if@termcalde@dtmconf@useregional
\newif\if@termcalde@dtmconf@numeric
```

Set variables to default values:

```
\@termcalde@compatfalse
\@termcalde@drawboxfalse
\@termcalde@dtmconf@frompreamblefalse
\@termcalde@dtmconf@useregionaltrue
\@termcalde@dtmconf@numerictrue
```

Define variables, p.r.n. with default values:

```
\def\termcalde@setdrawbox{}
\def\termcalde@dtmdialect{german}
```

Define a `compat` option for switching on compatibility mode:

```
\pgfkeys{%
    /termcal-de/compat/.cd, .is choice, .default=true,
        true/.code={\@termcalde@compattrue},
        false/.code={\@termcalde@compatfalse}}
```

Define a `drawdateframe` option set for configuring whether a frame is drawn around the date:

`always` Always draw a frame around the date

`atNewMonth` Draw a frame around the date at the beginning of a month

`never` Never draw a frame around the date

```
\pgfkeys{%
    /termcal-de/drawdateframe/.cd, .is choice, .default=always,
            always/.code={\def\termcalde@setdrawbox{\@termcalde@drawboxtrue}},
            atNewMonth/.code={\def\termcalde@setdrawbox{%
                \ifnewmonth\@termcalde@drawboxtrue%
                \else\@termcalde@drawboxfalse%
                \fi}},
            never/.code={\def\termcalde@setdrawbox{\@termcalde@drawboxfalse}}}
```

Define a `datetime2` option for configuring `datetime2`:

`local` Defines which language module should be loaded.
Possible values are `german`, `de-DE`, `de-AT` and `de-CH` loading `datetime2-german`'s according sub-module and `useregional`, which determines the used sub-module based on the language settings of `babel` or `polyglossia`

`numeric` Influences whether to use the numeric style when printing dates.
Possible values are `true` and `false`. Is the `numeric` key set without a value, it is assumed to be `true`.

`frompreamble` This option has to be set when `datetime2` is loaded in the preamble. Overrides all other options.

```
\pgfkeys{%
    /termcal-de/datetime2/.code={\pgfkeys{/termcal-de/datetime2/.cd, #1}},
    /termcal-de/datetime2/local/.cd, .is choice, .default=useregional,
        useregional/.code={\@termcalde@dtmconf@useregionaltrue},
        german/.code={%
            \@termcalde@dtmconf@useregionalfalse%
            \def\termcalde@dtmdialect{german}},
        de-DE/.code={%
            \@termcalde@dtmconf@useregionalfalse%
            \def\termcalde@dtmdialect{de-DE}},
        de-AT/.code={%
            \@termcalde@dtmconf@useregionalfalse%
            \def\termcalde@dtmdialect{de-AT}},
        de-CH/.code={%
            \@termcalde@dtmconf@useregionalfalse%
            \def\termcalde@dtmdialect{de-CH}},
    /termcal-de/datetime2/numeric/.cd, .is choice, .default=true,
        true/.code={\@termcalde@dtmconf@numerictrue},
        false/.code={\@termcalde@dtmconf@numericfalse},
    /termcal-de/datetime2/frompreamble/.cd, .is choice, .default=true,
        true/.code={\@termcalde@dtmconf@frompreambletrue},
        false/.code={\@termcalde@dtmconf@frompreamblefalse}}
```

Process the options:

```
\ProcessPgfPackageOptions{/termcal-de}
```

Require and configure `datetime2`

`\termcalde@dtmnumeric` Define an auxiliary command adding `=numeric` to `datetime2`'s `useregional` key and adding `-numeric` to `datetime2`'s module names, depending on the current configuration of `datetime2`:

```
\def\termcalde@dtmnumeric{%
    \if@termcalde@dtmconf@numeric%
        \if@termcalde@dtmconf@useregional=\else-\fi%
        numeric\fi}
```

Require `datetime2` for printing dates inside the calendar boxes and configure it as long as the `datetime2=frompreamble` key is not set.

```
\if@termcalde@dtmconf@frompreamble\RequirePackage{datetime2}%
\else%
    \RequirePackage[%
        \if@termcalde@dtmconf@useregional{useregional}%
        \else\termcalde@dtmdialect\fi%
        %
        \if@termcalde@dtmconf@useregional\termcalde@dtmnumeric\fi]{datetime2}%
\fi
```

When `datetime2`'s language module is loaded by using the module name, a hook executing `\DTMsetstyle` at the begin of the document is required for setting the date style to the numeric format.

```
\if@termcalde@dtmconf@frompreamble\else%
    \if@termcalde@dtmconf@useregional\else%
        \if@termcalde@dtmconf@numeric%
            \AtBeginDocument{\DTMsetstyle{\termcalde@dtmdialect\termcalde@dtmnumeric}}%
\fi\fi\fi
```

Redefinitions

`\setdate` Use `D.M.YYYY` instead of `m/d/y` when entering dates from the code unless the `compat` option is given. Do *not* use leading zeros in date specifications!

```
\if@termcalde@compat\else%
    \def\setdate@#1.#2.#3!{%
        \setcounter{date}{#1}%
        \setcounter{month}{#2}%
        \setcounter{year}{#3}%
        \global\newmonthtrue\setleap}%
\fi
```

`\curdate` This command is used internally by `termcal`.
Redefine `\curdate`'s output format to be the same as `\setdate`'s.
Remember: Do *not* use leading zeros in date specifications!

```
\if@termcalde@compat\else%
    \def\curdate{\arabic{date}.\arabic{month}.\arabic{year}}%
\fi
```

`\currentdate` Provides a facility to print the date inside a cell's content.
The date format can be configured via configuring `\DTMdisplaydate`.

```
\def\currentdate{\DTMdisplaydate{%
    \arabic{year}}{\arabic{month}}{\arabic{date}}{-1}}
```

`\calprintdate` Prints the date displayed in the cell heading.
The date format can be configured via configuring `\DTMDisplaydate`.

```
\def\calprintdate{%
    \termcalde@setdrawbox%
    \if@termcalde@drawbox\framebox{%
        \DTMDisplaydate{\arabic{year}}{\arabic{month}}{\arabic{date}}{-1}}%
    \else\DTMDisplaydate{\arabic{year}}{\arabic{month}}{\arabic{date}}{-1}%
    \fi}
```

Change History

Index