The calcage package*

Robin Schneider
ypid23@aol.de

September 16, 2012

Abstract

The calcage package can calculate the age in years.
Location on CTAN: http://www.ctan.org/pkg/calcage
Fork me on GitHub: https://github.com/ypid/latex-packages/tree/master/calcage

Contents

1 Introduction

The calcage package can calculate the age of someone or something in years. Internally it uses the datenumber package to calculate the age in days. The conversion from days to years is implemented by this package. It is also taken care about leap years and such odd things. So if you enter your birthday you get your exact age in years. You can even get the age as number with up to 18 places after the decimal separator but I heard that this is a bit uncommon for the age of a person ...

*This document corresponds to calcage v0.90, dated 2012/09/09.

2 Usage

Just load the package placing

```
\usepackage{calcage}
```

in the preamble of your LaTeX 2ε source file.

You can also give optional package options to change the default behavior. But you can not give values (see Bugs). So

```
\usepackage[presision]{calcage}
```

works and sets the presision to 3. But "presision=3" is going to fail ...

`\calcage` The macro `\calcage [`⟨*optional parameters*⟩`] {`⟨*year*⟩`} {`⟨*month*⟩`} {`⟨*day*⟩`}` takes a date and typesets the difference to the current date in years.

`year, month, day` There are some optional parameters. For example you can adjust the current date for calcage with the keys "*year*", "*month*" and "*day*". You don't have to specifier all of them. A subset is also possible.

`precision` Another useful parameter is "*precision*". With this parameter you can specify how many places after the decimal separator you would like to get. The default precision is 0 which means that `\calcage` will typeset the age in years as integer. If you omit a value for "precision" then the precision will be 3.

`positive` The macro `\calcage` can also take a date which is in the future. The default behavior in this case is a negative number but you can make the number positive with the "*positive*" boolean switch.

`printyear` The next parameter I would like to show you is the "*printyear*" boolean switch. With this switch you can specify if `\calcage` should add "year" or rather "years" after the age. This is the default setting.

`yearsuffix` For the previous parameter you may need the possibility (at least in the German language) to add a single character to the plural word. For example

```
Albert Einstein wurde vor \calcage[yearsuffix]{1879}{03}{14}
geboren.
```

Albert Einstein wurde vor 133 Jahren geboren.

To do this you can use the boolean switch "*yearsuffix*". By default this switch is false. If this boolean is true and the German language was selected then a "n" will be added after "Jahre". If the English language was selected then this switch will not change the output.

`numberstring` And last but not least there is the boolean switch "numberstring" which is true by default. This means that the typesetting process of an integer is done by the package fnumprint which will typeset the word of a number instead of the Arabic number if the value of the number is between 0 and 12. If you don't like this behavior you can set the boolean to false with "numberstring=false".

3 Bugs

I tried to implement the possibility to change the default behavior as package option in a nice way but with the current implementation there is a problem with the values. If you give a key with value then the `\setkeys` macro which gets these parameters as macro fails. I already tried `\expandafter` so if anyone knows where the problem is or has a solution feel free to contact me.

I think there is a little inaccuracy in the conversion from days to years because my implementation just removes the leap years before dividing. But in my tests this seems to be more theoretically.

The problem only has an effect if leap years are involved because if a leap year is calculated then the additional day is subtracted. This means that one day is not longer $\frac{1}{366}$ year but $\frac{1}{365}$ year ...

4 Examples

```
\TeX{} Live realeases: \\
\calcage{2012}{07}{08} \\
\calcage{2011}{07}{20} \\
\calcage{2010}{09}{10} \\
\calcage{2009}{11}{09}
```

TEX Live realeases:
zero years
one year
two years
two years

```
Donald Knuth is \calcage{1938}{01}{10} old. \\
Donald Knuth will be 100 years old in
  \calcage[positive, precision]{2038}{01}{10}. \\
Albert Einstein died at the age of
  \calcage[year=1955, month=04, day=18]{1879}{03}{14}. \\
Age of Linus Torvalds in years: \calcage[printyear=false]{1969}{12}{28} \\
```

Donald Knuth is 74 years old.
Donald Knuth will be 100 years old in 25.317 years.
Albert Einstein died at the age of 76 years.
Age of Linus Torvalds in years: 42

5 Implementation

This package depends on these packages.

```
\RequirePackage{fnumprint}[2012/08/27]
\RequirePackage{
  datenumber,
  fp,
  calc,
  xkeyval,
  kvoptions,
  xifthen,
}
```

5.1 Declaring the options

```
\DeclareStringOption{year}
\DeclareStringOption{month}
\DeclareStringOption{day}
\DeclareStringOption{precision}[3]
\DeclareBoolOption{positive}
\DeclareBoolOption{printyear}
\DeclareBoolOption{yearsuffix}
\DeclareBoolOption{numberstring}
```

To test if all parameters are valid the macro `\ProcessLocalKeyvalOptions*` is expanded to ensure this before leaving the preamble. This is the only purpose for the `\ProcessLocalKeyvalOptions*` macro in this case.

```
\ProcessLocalKeyvalOptions*
```

The next source code line creates a new macro which captures the parameters to use them as default options for the macro `\calcage`. If this macro is set to "precision=4" for example the `\setkeys` is going to fail.

```
\edef\calcage@options{\@ptionlist{\@currname.\@currext}}
%% ^^A \renewcommand{\calcage@options}{precision=4,printyear=false}
%% ^^A Package xkeyval Error: 'precision=4' undefined in families 'calcage'.
```

5.2 Language selection

Here comes the language selection part. The counter is already set by the fnumprint so I use it's value also in this package because I can rely on this counter.[1] If you prefer another language than English or German you can redefine the following macro definitions.

```
\ifcase\value{fnumprint@language}\or
```

If fnumprint@language is equal 1

```
  \newcommand{\calcage@yearWord}{Jahr}
  \newcommand{\calcage@yearPluralSuffix}{e}
  \newcommand{\calcage@yearSuffix}{n}
\or
```

[1] I am also the maintainer of the fnumprint package ...

If fnumprint@language is equal 2

```
\newcommand{\calcage@yearWord}{year}
\newcommand{\calcage@yearPluralSuffix}{s}
\newcommand{\calcage@yearSuffix}{}
\fi
```

5.3 Macro definition

Some necessary LaTeX counters are declared here before creating the `\calcage` macro.

```
\newcounter{calcage@today}\newcounter{calcage@ageindays}
\newcounter{calcage@myyear}\newcounter{calcage@leapyears}
```

`\calcage` And now comes the important part – the definition of `\calcage`. The first thing that has to be done is expanding the `\setkeys` macro. It sets the default options then the package options get the opportunity to overwrite these settings and finally the macro options get the same possibility.

```
\newcommand{\calcage}[4][]{%
  \setkeys{calcage}{precision=0, positive=true, printyear=true,
    yearsuffix=false, numberstring=true,
    year=\the\year, month=\the\month, day=\the\day, \calcage@options, #1}%
  \setmydatenumber{calcage@today}{\calcage@year}{\calcage@month}{\calcage@day}%
  \setmydatenumber{calcage@ageindays}{#2}{#3}{#4}%
```

Now comes the tricky part which are the leap years ...
My first implementation worked with dividing by 365.2425 but this was not perfect. So the current implementation counts how many leap years are between the birth year and the current year and subtracts (or adds[2]) this number.

```
  \setcounter{calcage@myyear}{#2}%
  \setcounter{calcage@leapyears}{0}%
  \ifthenelse{\equal{#2}{\calcage@year}}{}{%
    \ifthenelse{\value{calcage@myyear}<\calcage@year}{%
```

If the birth year is in the past.

```
      \loop%
        \stepcounter{calcage@myyear}%
        \ifleapyear{\thecalcage@myyear}\stepcounter{calcage@leapyears}\fi%
        \ifnum\value{calcage@myyear}<\calcage@year%
      \repeat%
    }{%
```

If the birth year is in the future.

```
      \loop%
        \ifleapyear{\thecalcage@myyear}\addtocounter{calcage@leapyears}{-1}\fi%
        \addtocounter{calcage@myyear}{-1}%
        \ifnum\value{calcage@myyear}>\calcage@year%
      \repeat%
```

[2] If the birth year is in the future because in this case the calcage@ageindays counter will be negative.

```
    }%
  }%
  \setcounter{calcage@ageindays}{\value{calcage@today}
    - \value{calcage@ageindays} - \value{calcage@leapyears}}%
  \ifthenelse{\boolean{calcage@positive} \AND \value{calcage@ageindays} < 0}{%
    \setcounter{calcage@ageindays}{\value{calcage@ageindays} * -1}%
  }{}%
```

Because we know that every year has now exactly 365 days we can divide by that.

```
  \FPdiv\calcage@age{\thecalcage@ageindays}{365}%
```

The next step is to truncated the age. There is no rounding used for this. I implemented it this way because if the age would be rounded then there would be a wrong result in cases like $x.999$ which would become $x+1$ if "precision=0" and that is probably not what you want ...

```
  \FPtrunc\calcage@age{\calcage@age}{\calcage@precision}%
```

The last thing to do is to typeset the age which is implemented by the next few lines of code.

```
  \ifthenelse{\boolean{calcage@numberstring}
    \AND \equal{\calcage@precision}{0}}%
    {\fnumprint[ein]{\calcage@age}}{\numprint{\calcage@age}}%
  \ifthenelse{\boolean{calcage@printyear}}{%
    ~\calcage@yearWord%
    \ifthenelse{\equal{\calcage@age}{1} \OR \equal{\calcage@age}{-1}}{}{%
      \calcage@yearPluralSuffix%
      \ifthenelse{\boolean{calcage@yearsuffix}}{\calcage@yearSuffix}{}%
    }%
  }{}%
}
```

That's it.

```
\endinput
```

Change History

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.