The latexdemo package

Matthias Pospiech
matthias@pospiech.eu

v0.2 from 2023/03/26

1 Introduction

In order to demonstrate LaTeX code it is very useful to have the code and the resulting output together in the same document. `latexdemo` is a package that provides configurable tools to print out LaTeX code and the resulting output in the same document.

The difference to other similar packages is the support of verbatim material inside a conditional sequence and the consequence that each code must be written to an external file.

The commands provided by this package are based on the packages `listings`, `mdframed` and the enviroment `filecontents`.

2 Basic example

Below is a principle example, which demonstrates the usage of the package by showing some commands of a package (`soul`) with the resulting code side by side:

Code:

```
\so{letterspacing}, \\
\ul{underlining},   \\
\st{overstriking}   \\
and \hl{highlighting}.
```

Result:

l e t t e r s p a c i n g,
underlining,
~~overstriking~~
and highlighting.

which is created with the code of this package using `\ifcsdef` from etoolbox to test if the code will run through or fail because of unkown commands.

```
\begin{DefineCode}
\so{letterspacing}, \\
\ul{underlining},   \\
\st{overstriking}   \\
and \hl{highlighting}.
\end{DefineCode}

\ifcsdef{so}{%
%
\PrintDemo{style=parallel}
%
}{%
```

```
  \DemoError{Command \cs{so} of package%
  \package{soul} not available.
  Probably the package was not loaded.
  }
}%
```

3 Usage

3.1 Define code

This package requires the example code to be written to an external file. The filename is saved in the command sequence \democodefile. The output is done with the DefineCode environment:

```
\begin{DefineCode}
... code ...
\end{DefineCode}
```

The requirement for an external file originates from the problem that verbatim content cannot be saved in normal tex macros since the line breaks and white spaces get lost. Furthermore, any solution which would solve this problem would fails finally because saving such contend cannot be included in conditional code statements[1].

The content in the file defined by \democodefile is further read for the printing of the code and the corresponding output.

3.2 Print code and result

\PrintDemo {style=⟨*option*⟩}

This is the macro for the output of code and result. The layout of both is defined with the style option. The following options are possible

parallel	code and result side by side.
stacked	(default) code and result with 100 % text width stacked.
lines	like stacked, but with lines on top and bottom of the result instead of a surrounding box.
none	like stacked, but with nothing around the result.
page	result on a single page.

You should use parallel for small examples and stacked otherwise. The options lines and none is primarily for those cases where a surrounding box is disturbing or impossible. The latter occurs for example in cases where content is written across the text width boundaries. The option page is obviously for those cases where the output is very large or written to another page anyway.

3.2.1 Examples

The following code is used in the examples

[1] See discussion on: http://tex.stackexchange.com/questions/29256/

```
\begin{DefineCode}
This code shows some basic math: $a^2 + b^2 = c^2$.
\end{DefineCode}
```

- \PrintDemo{style=parallel}

  Code:

  ```
  This code shows some basic math:
  $a^2 + b^2 = c^2$.
  ```

  Result:

  This code shows some basic math: $a^2 + b^2 = c^2$.

- \PrintDemo{style=stacked}

  Code:

  ```
  This code shows some basic math: $a^2 + b^2 = c^2$.
  ```

  Result:

  This code shows some basic math: $a^2 + b^2 = c^2$.

- \PrintDemo{style=lines}

  Code:

  ```
  This code shows some basic math: $a^2 + b^2 = c^2$.
  ```

  Result:

  This code shows some basic math: $a^2 + b^2 = c^2$.

- \PrintDemo{style=none}

  Code:

  ```
  This code shows some basic math: $a^2 + b^2 = c^2$.
  ```

  Result:

  This code shows some basic math: $a^2 + b^2 = c^2$.

The prefix text for code and result and the filename can be changed. The commands are introduced in section 3.3. Commands for the output of content such as the name of of a package, a command, an environment or a generalized error message are introduced in section 3.4.

3.3 Setup

The following commands define the name for the temporary file or the strings used for the printing of code or results. Use `\renewcommand` to change the definitions.

`\democodefile` Filename for the temporary file required for code and results printing.
Default: `democode`

`\democodeprefix` Prefix text for output of code.
Default: `Code:`

`\demoresultprefix` Prefix text for output of the result.
Default: `Result:`

3.4 Output commands

The following commands are provided for the user to print out and format commands, environments, packages and errors. Some are provided by the `doctools` package.

`\command` `{`⟨*cmd*⟩`}`
Prints out the argument `\command{foo}` as `\foo`.

`\cs` `{`⟨*cmd*⟩`}`
Shortcut for `\command`.

`\arg` `{`⟨*cmd*⟩`}`
Prints out an argument in curled brackets without the use of angle brackets as in `\marg` or `\oarg`. Thus prints `\arg{foo}` as {`foo`}.

`\environment` `{`⟨*environment*⟩`}`
Prints out an environment name as `environment`.

`\env` `{`⟨*environment*⟩`}`
Shortcut for `\environment`

`\package` `{`⟨*package*⟩`}`
Prints out a package name as `package`.

`\DemoError` prints out the given error message
Example: `Error: foo`

4 Implementation

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexdemo}[2012/12/01 v0.1 typeset code and resulting
output]
\@ifpackageloaded{hypdoc}
  {\RequirePackage[loadHyperref=true,%
                   createIndexEntries=false,%
                   applyLayout=false]{doctools}}
  {\@ifpackageloaded{doc}
    {\RequirePackage[loadHyperref=false,%
                     createIndexEntries=false,%
                     applyLayout=false]{doctools}}
    {}}
%%% listings (must be loaded before \AtBeginDocument)
\RequirePackage{listings}
\PassOptionsToPackage{table}{xcolor}
% This code needs to be executed at the beginning
% of the document because some packages (eg. xcolor)
% could lead to option clashes otherwise
%
%% Programming
\RequirePackage{xspace}
\RequirePackage{etoolbox}
%% Packages for frames
\RequirePackage{mdframed}
\RequirePackage{framed}
%
\AtBeginDocument{%
%
```

4.1 Preamble

4.1.1 Packages

```
%% Package for colors
\RequirePackage{xcolor}
%
%% load doctools without hyperref if not loaded and no documentation
%% package was loaded.
\@ifpackageloaded{doctools}{}
  {\RequirePackage[loadHyperref=false,%
                   createIndexEntries=true,%
                   applyLayout=false]{doctools}}
%
```

4.1.2 Colors

```
%% Colors
\colorlet{demo@stringcolor}{green!40!black!100}
\colorlet{demo@commentcolor}{green!50!black!100}
\colorlet{demo@numbercolor}{white!50!black!100}
\colorlet{demo@codebackcolor}{white!95!black!100}
\colorlet{demo@resultbackcolor}{white}
```

```
\definecolor{demo@keywordcolor}{rgb}{0,0.47,0.80}
\definecolor{demo@rulecolor}{rgb}{0,0.4,0.5}
\definecolor{demo@code@rulecolor}{rgb}{0.5,0.5,0.5}
%
```

4.2 Commands

```
%% === Simple Commands ================================
%
```

`\democodefile` Saves the filename for temporary file output.

```
\newcommand{\democodefile}{democode}
%
```

`\democodeprefix` Prefix text for code output.

```
\newcommand{\democodeprefix}{Code: }
%
```

`\demoresultprefix` Prefix text for result output.

```
\newcommand{\demoresultprefix}{\noindent Result:}
%
```

`\DemoError` Output and formatting of error messages.

```
%% Print Error
\newcommand{\DemoError}[1]{
  \ifcsdef{textcolor}
    {\textcolor{red}{Error:~}}
    {Error:~}
  #1 \par\noindent
}
%
```

4.3 Define keys

```
%% === Define Keys ====================================
\RequirePackage{kvoptions-patch}
\RequirePackage{kvoptions}  % options
\RequirePackage{pdftexcmds} % string comparison
\SetupKeyvalOptions{family=demo,prefix=demo@}
%
```

Define default option for style key: *stacked*

```
\DeclareStringOption[stacked]{style}
\ProcessKeyvalOptions{demo}
%
```

`\PrintDemoUsingKeys` Evaluate key and execute corresponding commands

```
\newcommand{\PrintDemoUsingKeys}{%
  \ifnum\pdf@strcmp{\demo@style}{parallel}=0%
    \PrintCodeAndResultsParallel%
  \else\ifnum\pdf@strcmp{\demo@style}{stacked}=0%
    \PrintCodeAndResultsStacked%
  \else\ifnum\pdf@strcmp{\demo@style}{lines}=0%
    \PrintCodeAndResultsStackedLines%
  \else\ifnum\pdf@strcmp{\demo@style}{page}=0%
    \PrintCodeAndResultsPage%
  \else\ifnum\pdf@strcmp{\demo@style}{none}=0%
    \PrintCodeAndResultsNone%
  \else%
     \PackageError{latexdemo}{%
       \MessageBreak%
       value >\tplbugs@style< unkown \MessageBreak%
     }{}%
  \fi\fi\fi\fi\fi%
}%
%
```

`DefineCode` Define code for demonstration using the enviroment `DefineCode` which is based on the enviroment `filecontents*`.

```
  \newenvironment{DefineCode}{%
  \csname filecontents*\endcsname[overwrite]{\democodefile}%
}{%
  \csname endfilecontents*\endcsname%
}
%
```

`\PrintDemo` Print code and result using the key-value syntax

```
\newcommand{\PrintDemo}[1]{%
\begingroup
  \setkeys{demo}{#1}%
  \PrintDemoUsingKeys
\endgroup
}
%
```

4.4 listings package style

```
%% === Listings style ================================
%% reuses style from doctools
\lstdefinestyle{demostyle}{
  ,style=lstDemoStyleLaTeXCode%
  ,numbers=none%
}
\lstloadlanguages{[LaTeX]TeX}
%
```

4.5 mdframed package style

```
%% === Mdframed style ================================
\mdfdefinestyle{DemoStyleFrames}{
  linecolor=demo@rulecolor,%
  linewidth=0.8pt,
  skipabove=0.5\baselineskip,
  skipbelow=0.5\baselineskip,
  leftmargin =-3.5pt,
  rightmargin=-3.5pt,
  innerleftmargin=3pt,
  innerrightmargin=3pt,
  needspace=3\baselineskip,
}%
%
```

4.6 Commands for the formatting

`\preResultSkip` Default skip at the beginning of a result

```
%% === Formating commands ============================
\newcommand{\preResultSkip}{}%\vspace*{-0.5\baselineskip}
%
```

`latexresult` Environment to print the result in a box

```
\newenvironment{latexresult}{%
\demoresultprefix
\nopagebreak[4]
\preResultSkip
\mdframed[%
  style=DemoStyleFrames,
  backgroundcolor=demo@resultbackcolor,%
  usetwoside=false,
]%
}{
\endmdframed
\noindent
}
%
```

`\resultline` Single Line for results

```
\newcommand{\resultline}{%
\nopagebreak[4]
%% Insert single line
\mdframed[%
  style=DemoStyleFrames,
  skipabove=3pt,
  skipbelow=3pt,
  topline=true,bottomline=false,leftline=false,rightline=false,
  backgroundcolor=white,%
]\mbox{}\endmdframed
\nopagebreak[4]
}
%
```

4.7 Low level commands for printing of code and result

\printlatexcode Prints the code using \lstinputlisting

```
%% === Output commands ================================
%% Print Code with prefix
\newcommand{\printlatexcode}[1][\democodefile]{%
\def\demoInputFile{#1}%
\IfFileExists{\demoInputFile.tex}{%
\democodeprefix%
\lstinputlisting[style=demostyle,nolol=true]{\demoInputFile}}{}%
}%
%
```

\printlatexresult Prints the result enclosed in the `latexresult` environment. The evaluation of the code is simply achieved by loading the file with \input.

```
%% Print Result with standard box
\newcommand{\printlatexresult}[1][\democodefile]{%
\def\demoInputFile{#1}%
\begin{latexresult}%
\IfFileExists{\demoInputFile.tex}{\input{\demoInputFile.tex}}{}%
\end{latexresult}%
}%
%
```

\printlatexresultlines Like \printlatexresult but with lines above and below instead of a surrounding box.

```
%% Print result with lines
\newcommand{\printlatexresultlines}{%
\demoresultprefix
\nopagebreak[4] \resultline \nopagebreak[4]
\IfFileExists{\democodefile}{\input{\democodefile}}{}%
\nopagebreak[4] \resultline \nopagebreak[4]
}%
%
```

4.8 Output of code and result

\PrintCodeAndResultsParallel

```
%% === Output commands for code and result ============
\newcommand{\PrintCodeAndResultsParallel}{%
\nopagebreak[4]
\vspace*{0.5em}\par\noindent
\begin{minipage}[t]{0.48\linewidth}
\printlatexcode
\end{minipage} \hfill
\begin{minipage}[t]{0.48\linewidth}
\printlatexresult
\end{minipage}
\par\noindent
}
%
```

```
\PrintCodeAndResultsStacked
\newcommand{\PrintCodeAndResultsStacked}{%
\nopagebreak[4]
\vspace*{0.5em}\par\noindent
\printlatexcode%
\printlatexresult%
\par\noindent
}%
%

intCodeAndResultsStackedLines
\newcommand{\PrintCodeAndResultsStackedLines}{%
\nopagebreak[4]
\vspace*{0.5em}\par\noindent
\printlatexcode%
\printlatexresultlines%
\vspace*{0.5em}\par\noindent
}%
%

\PrintCodeAndResultsNone
\newcommand{\PrintCodeAndResultsNone}{%
\nopagebreak[4]
\vspace*{0.5em}\par\noindent
\printlatexcode%
%

\demoresultprefix
\nopagebreak[4]
\par\noindent
\IfFileExists{\democodefile}{\input{\democodefile}}{}%
%

\vspace*{0.5em}\par\noindent
}%
%

\PrintCodeAndResultsPage
\newcommand{\PrintCodeAndResultsPage}{%
\nopagebreak[4]
\vspace*{0.5em}\par\noindent
\printlatexcode%
\demoresultprefix: Shown on the following page.
\newpage
\IfFileExists{\democodefile}{\input{\democodefile}}{}%
\newpage
}%
%

} % end of \AtBeginDocument
%
```

Index

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