DocOnce: Document Once, Include Anywhere

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Jun 7, 2015

1 Some DocOnce Features

- Strong support for texts with much math and code.
- Same source can produce a variety of output formats:
  - traditional LaTeX B/W documents for printing
  - color LaTeX PDF documents
  - color LaTeX PDF documents for viewing on small phones
  - Sphinx HTML documents with 20+ different designs
  - Plain HTML, Boostrap HTML, or with a template, or with another template, or solarized
  - HTML for Google or Wordpress blog posts
  - MediaWiki (Wikipedia, Wikibooks, etc)
  - Markdown
  - IPython notebook

  Other formats include plain untagged text (for email), Creole wiki (for Bitbucket wikis), Google wiki (for Googlecode), reStructuredText, and Epytext.
- Integration with Mako enables use of variables, functions, if-tests, and loops to parameterize the text and generate various versions of the text for different purposes.
- Computer code can be copied directly from parts of source code files.

- Running text can quickly be edited to slide formats (reveal.js and deck.js, based on HTML5+CSS3).
- Special exercise environments with support for hints, answers, subexercises, etc.
- Automatic inline embedding of YouTube and Vimeo movies.
- Good support for admonitions in various LaTeX and HTML styles for warnings, questions, hints, remarks, summaries, etc.

2 What Does DocOnce Look Like?

DocOnce text looks like ordinary text (much like Markdown[1]), but there are some almost invisible text constructions that allow you to control the formating. Here are some examples.

- Bullet lists automatically arise from lines starting with `*`, or `o` if the list is to be enumerated.
- *Emphasized words* are surrounded by `*`. **Words in boldface** are surrounded by underscores.
- Words from computer code are enclosed in backticks and then typeset `verbatim (in a monospace font)`.
- Section and paragraph headings are recognied special decorating characters (= or _) before and after the heading. The length of the decoration determines the level of the section.
- Blocks of computer code are included by surrounding the blocks with `!bc` (begin code) and `!ec` (end code) tags on separate lines.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics are included by surrounding the blocks with `!bt` (begin TeX) and `!et` (end TeX) tags on separate lines.
- There is support for both LaTeX and text-like inline mathematics such that formulas make sense also when not rendered by LaTeX or MathJax.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported. YouTube and Vimeo videos are automatically embedded in web documents.

[1] In fact, DocOnce allows basic GitHub/extended Markdown syntax as input. This is attractive for newcomers from Markdown or writers who also write Markdown documents (or uses Markdown frequently at GitHub).

- The abstract of a document starts with *Abstract* as paragraph heading, and all text up to the next heading makes up the abstract,
- Special comment lines are not visible in the output.
- Comments to authors can be inserted throughout the text and made visible or invisible as desired.
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files), large portions of text can be defined in or out of the text, and tailored format-specific constructs can easily be included. With Mako a single text can output its program examples in two or more languages.

2.1 What Can DocOnce Be Used For?

LaTeX is ideal for articles, thesis, and books, but the PDF files does not look fresh and modern on tablets and phones or big computer screens. For the latter type of media you need HTML-based documents with strong support for nice layouts. Tools like Sphinx, Markdown, or plain HTML with Bootstrap are then more appropriate than LaTeX, but involves a very different syntax. DocOnce lets you write one text in one place and then generate the most appropriate language for the media you want to target. DocOnce also has many extra features for supporting large documents with much code and mathematics, not found in any of other publishing tool.

2.2 Basic Syntax

Here is an example of some simple text written in the DocOnce format:

```
======= First a Section Heading with 7 = Characters =======

===== Then a Subsection Heading with 5 = Characters =====

=== Finally a Subsubection Heading with 3 = Characters ===

You can also have paragraphs with a paragraph heading surrounded
by double underscores are the beginning of a line.

__This is a paragraph heading.__
And here comes the text.

===== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, but must always start at the
beginning of lines. Tags used for _boldface_ words, *emphasized*
words, and `computer' words look natural in plain text.  Quotations
appear inside double backticks and double single quotes, as in ``this
example''.
```

```
Below the section title we have a *label*, which can be used to
refer to Section ref{my:first:sec}.
References to equations, such as XXX1, work in the same
LaTeX-inspired way.

Lists are typeset as you would do in email,

  * item 1
  * item 2,
    perhaps with a 2nd line
  * item 3

Note the consistent use of indentation (as in Python programming!).
Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3,
    but be careful with the indentation of the next lines!

__Hyperlinks.__
URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl".
If the word is just URL, the URL itself becomes the link name,
as in URL: "tutorial.do.txt". DocOnce distinguishes between paper
and screen output. In traditional paper output, in PDF generated from LaTeX
generated from DocOnce, the URLs of links appear as footnotes.
With screen output, all links are clickable hyperlinks, except in
the plain text format which does not support hyperlinks.

__Inline comments.__
DocOnce also allows inline comments of the form [name: comment] (with
a space after `name:`), e.g., such as [hpl: here I will make some
remarks to the text]. Inline comments can be removed from the output
by a command-line argument (see Section ref{doconce2formats} for an
example). Inline comments can also be used for detailed editing of
text, much like track changes in word, to illustrate how a text
is revised. (However, for seeing how others have revised the text, I
strongly recommend using Git for version control and running `git diff`
on the appropriate versions, or you can click on differences at
GitHub if the files are hosted there.)

__Footnotes.__ Adding a footnote[^footnote] is also possible.

[^footnote]: The syntax for footnotes is borrowed from Extended Markdown.

__Tables.__
Tables are also written in the plain text way, e.g.,

  |--c--------c-----------c--------|
  |time  | velocity | acceleration |
  |---r-------r-----------r--------|
  | 0.0  | 1.4186   | -5.01        |
  | 2.0  | 1.376512 | 11.919       |
  | 4.0  | 1.1E+1   | 14.717624    |
  |--------------------------------|

The characters `c`, `r`, and `l` can be inserted, as illustrated above,
for aligning the headings and the columns (center, right, left).

# Lines beginning with # are comment lines.
```

The DocOnce text above results in the following little document:

3 First a Section Heading with 7 = Characters

3.1 Then a Subsection Heading with 5 = Characters

Finally a Subsubection Heading with 3 = Characters. You can also have paragraphs with a paragraph heading surrounded by double underscores are the beginning of a line.

This is a paragraph heading. And here comes the text.

3.2 A Subsection with Sample Text

Ordinary text looks like ordinary text, but must always start at the beginning of lines. Tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Quotations appear inside double backticks and double single quotes, as in “this example”.

Below the section title we have a *label*, which can be used to refer to Section 3.2. References to equations, such as ’(1)’, work in the same LaTeX-inspired way.

Lists are typeset as you would do in email,

- item 1
- item 2, perhaps with a 2nd line
- item 3

Note the consistent use of indentation (as in Python programming!). Lists can also have automatically numbered items instead of bullets,

1. item 1
2. item 2
3. item 3, but be careful with the indentation of the next lines!

Hyperlinks. URLs with a link word are possible, as in hpl. If the word is just URL, the URL itself becomes the link name, as in `tutorial.do.txt`. DocOnce distinguishes between paper and screen output. In traditional paper output, in PDF generated from LaTeX generated from DocOnce, the URLs of links appear as footnotes. With screen output, all links are clickable hyperlinks, except in the plain text format which does not support hyperlinks.

Inline comments. DocOnce also allows inline comments of the form **name 1**: comment (with a space after `name:`), e.g., such as **hpl 2**: here I will make some remarks to the text . Inline comments can be removed from the output by a command-line argument (see Section 4 for an example). Inline comments can also be used for detailed editing of text, much like track changes in word, to illustrate how a text is revised. (However, for seeing how others have revised the text, I strongly recommend using Git for version control and running `git diff` on the appropriate versions, or you can click on differences at GitHub if the files are hosted there.)

Footnotes. Adding a footnote[2] is also possible.

Tables. Tables are also written in the plain text way, e.g.,

time	velocity	acceleration
0.0	1.4186	-5.01
2.0	1.376512	11.919
4.0	1.1E+1	14.717624

The characters `c`, `r`, and `l` can be inserted, as illustrated above, for aligning the headings and the columns (center, right, left).

3.3 Mathematics

Inline mathematics, such as $\nu = \sin(x)$ is written exactly as in LaTeX:

```
$\nu = \sin(x)$
```

Blocks of mathematics are typeset with raw LaTeX, inside `!bt` and `!et` (begin TeX, end TeX) directives:

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f,
label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f, \tag{1}$$

$$\frac{\partial v}{\partial t} = \nabla \cdot (q(u)\nabla v) + g \tag{2}$$

[2] The syntax for footnotes is borrowed from Extended Markdown.

Of course, such blocks only looks nice in formats with support for LaTeX mathematics (this includes `latex`, `pdflatex`, `html`, `sphinx`, `ipynb`, `pandoc`, and `mwiki`). Simpler formats have to just list the raw LaTeX syntax.

LaTeX writers who adopt DocOnce need to pay attention to the following:

- AMS LaTeX mathematics is supported, also for the `html`, `sphinx`, and `ipynb` formats.
- Only five equation environments can be used: `\[ ... \]`, `equation*`, `equation`, `align*`, and `align`.
- Newcommands in mathematical formulas are allowed (but not in the running text). Newcommands must be defined in files with names `newcommands*.tex`.
- MediaWiki (`mwiki`) does not support references to equations. DocOnce has extended Markdown, Sphinx, and HTML to better support equation referencing (across web pages for example).

Remark.

Although DocOnce allows user-defined styles in the preamble of LaTeX output, HTML-based output cannot make use of such styles. If-else constructs for the preprocessor can be used to allow special LaTeX environments for LaTeX output and alternative typesetting for other formats, but it is recommended to stay away from special environments in the text and write in a simpler fashion. For example, DocOnce has no special construction for algorithms, so these must be simulated by lists or verbatim blocks. Other constructions that should be avoided include margin notes, special tables, and `subfigure` (combine image files to one file instead, via `doconce combine_images`).

3.4 Computer Code

You can have blocks of computer code, starting and ending with `!bc` and `!ec` instructions, respectively.

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
```

Such blocks are formatted as

```
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to `sphinx`, `rst`, and formats close to plain text), not directly after a section/paragraph heading or a table.

Blocks of computer code has *named environments*, such as *pycod*. The *py* stands for Python and *cod* indicates a code snippet that cannot be run without more code. Another example is *fpro*, *f* for Fortran and *pro* for a complete program that will run as it stands. There is support for code in C, C++, Fortran, Java, Python, Perl, Ruby, JavaScript, HTML, and LaTeX,

One can also copy computer code directly from files, either the complete file or specified parts, e.g,

```
@@@CODE src/myprog.py fromto: def regression\(@import mymod
```

The copying is based on regular expressions and not on line numbers, which makes the specifications much more robust during software and document developing. With the `@@@CODE` command, computer code is never duplicated in the documentation (important for the principle of avoiding copying information!) and once the software is updated, the next compilation of the document is up-to-date.

Inclusion of files. Another DocOnce document or any file can be included by writing `# #include "mynote.do.txt"` at the beginning of a line. DocOnce documents have extension `do.txt`. The `do` part stands for doconce, while the trailing `.txt` denotes a text document so that editors gives you plain text editing capabilities.

3.5 Macros (Newcommands), Cross-References, Index, and Bibliography

DocOnce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands are defined in files with names `newcommands*.tex`, using standard LaTeX syntax. Only newcommands for use inside math environments are supported.

Labels, corss-references, citations, and support of an index and bibliography are much inspired by LaTeX syntax, but DocOnce features no backslashes. Use labels for sections and equations only, and preceed the reference by "Section" or "Chapter", or in case of an equation, surround the reference by parenthesis.

Here is an example:

```
===== My Section =====
label{sec:mysec}

idx{key equation} idx{$\u$ conservation}

We refer to Section ref{sec:yoursec} for background material on
the *key equation*. Here we focus on the extension

# \Ddt, \u and \mycommand are defined in newcommands_keep.tex

!bt
\begin{equation}
\Ddt{\u} = \mycommand{v},
label{mysec:eq:Dudt}
\end{equation}
!et
where $\Ddt{\u}$ is the material derivative of $\u$.
Equation (ref{mysec:eq:Dudt}) is important in a number
of contexts, see cite{Larsen_et_al_2002,Johnson_Friedman_2010a}.
Also, cite{Miller_2000} supports such a view.

As see in Figure ref{mysec:fig:myfig}, the key equation
features large, smooth regions *and* abrupt changes.

FIGURE: [fig/myfile, width=600 frac=0.9] My figure. label{mysec:fig:myfig}

===== References =====

BIBFILE: papers.pub
```

DocOnce applies Publish for specifying bibliographies because this tool has more functionality than BibTeX, but any BibTeX database can be automatically converted to the simple Publish format.

For further details on functionality and syntax we refer to the `doc/manual/manual.do.txt` file in the DocOnce source and a Sphinx version of this document.

4 From DocOnce to Other Formats

Transformation of a DocOnce document `mydoc.do.txt` to various other formats is done with the script `doconce format`:

```
Terminal> doconce format formatname mydoc.do.txt
```

or just drop the extension:

```
Terminal> doconce format formatname mydoc
```

4.1 Generating a makefile

Producing HTML, Sphinx, and in particular LaTeX documents from DocOnce sources requires a few commands. Often you want to produce several different formats. The relevant commands should then be placed in a script that acts as a “makefile”.

The `doconce makefile` can be used to automatically generate such a makefile, more precisely a Python script `make.py`, which carries out the commands explained below. If our DocOnce source is in `main_myproj.do.txt`, we run

```
doconce makefile main_myproj html pdflatex sphinx
```

to produce the necessary output for generating HTML, PDFLaTeX, and Sphinx. Usually, you need to edit `make.py` to really fit your needs. Some examples lines are inserted as comments to show various options that can be added to the basic commands. A handy feature of the generated `make.py` script is that it inserts checks for successful runs of the many `doconce` commands, and if something goes wrong, the script aborts.

4.2 Preprocessing

The `preprocess` and `mako` programs are used to preprocess the file. The DocOnce program detects whether `preprocess` and/or `mako` statements are present and runs the corresponding programs, first `preprocess` and then `mako`.

Variables to `preprocess` and/or `mako` can be added after the filename with the syntax `-DMYVAR`, `-DMYVAR=val` or `MYVAR=val`.

- The form `-DMYVAR` defines the variable `MYVAR` for `preprocess` (like the same syntax for the C preprocessor - `MYVAR` is defined, but has not specific value). When running `mako`, `-DMYVAR` means that `MYVAR` has the (Python) value `True`.
- The expressions `-DMYVAR=val` and `MYVAR=val` are equivalent. When running `preprocess`, `MYVAR` is defined and has the value `val` (`# #ifdef MYVAR` and `# #if MYVAR == "val"` are both true tests), while for `mako`, `MYVAR` exists as variable and has the value `val` (`% if MYVAR == "val"` is true).

Note that `MYVAR=False` defines `MYVAR` in `preprocess` and any test `# #ifdef MYVAR` is always true, regardless of the value one has set `MYVAR` to, so a better test is `# #if MYVAR == True`. In general, it is recommended to go with `preprocess` directives if the tests are very simple, as in `# #ifdef MYVAR` or `# #if FORMAT == "latex"`, otherwise use only `mako` syntax like `% if MYVAR or YOURVAR:` to incorporate `if` tests in the preprocessor phases.

Two examples on defining preprocessor variables are

```
Terminal> doconce format sphinx mydoc -Dextra_sections -DVAR1=5
Terminal> doconce format sphinx mydoc extra_sections=True VAR1=5
```

The variable `FORMAT` is always defined as the current format when running `preprocess` or `mako`. That is, in the above examples, `FORMAT` is defined as `sphinx`. Inside the DocOnce document one can then perform format specific actions through tests like `#if FORMAT == "sphinx"` (for `preprocess`) or `% if FORMAT == "sphinx":` (for `mako`).

The result of running `preprocess` on a DocOnce file `mydoc.do.txt` is available in a file `tmp_preprocess__mydoc.do.txt`. Similarly, the result of running `mako` is available in `tmp_mako__mydoc.do.txt`. By examining these files one can see exactly what the preprocessors have done.

4.3 Removal of inline comments

The command-line arguments `--no_preprocess` and `--no_mako` turn off running `preprocess` and `mako`, respectively.

Inline comments in the text are removed from the output by

```
Terminal> doconce format latex mydoc --skip_inline_comments
```

One can also remove all such comments from the original DocOnce file by running:

```
Terminal> doconce remove_inline_comments mydoc
```

This action is convenient when a DocOnce document reaches its final form and comments by different authors should be removed.

4.4 Notes

DocOnce does not have a tag for longer notes, because implementation of a "notes feature" is so easy using the `preprocess` or `mako` programs. Just introduce some variable, say `NOTES`, that you define through `-DNOTES` (or not) when running `doconce format ...`. Inside the document you place your notes between `# #ifdef NOTES` and `# #endif` preprocess tags. Alternatively you use `% if NOTES:` and `% endif` that `mako` will recognize. In the same way you may encapsulate unfinished material, extra material to be removed for readers but still nice to archive as part of the document for future revisions.

4.5 Demo of different formats

A simple scientific report is available in a lot of different formats. How to create the different formats is explained in more depth in the coming sections.

4.6 Useful Options for `doconce format`

The `doconce format` command used to translate a DocOnce document to an output format performs some syntax check and to notify the user about common problems. There are some useful options for turning on additional checks:

- `--labelcheck=on` (or `off`) to check that every `ref` reference has a corresponding `label` definition within the document (this check may lead to wrong diagnostics, e.g., when a label is defined in an external document and referred via generalized references, so the check must be used with care)

- `--urlcheck` checks that all URLs referred to in the document are valid.

Other useful options are

- `--os_prompt=PROMPT>` sets the prompt, here `PROMPT>`, as terminal prompt in output from running OS commands with the `@@@OSCMD` instruction. The value `None` gives no prompt.
- `--code_prefix=X` prefixes all `@@@CODE` imports with some path `X` (if the source files are located in some other directory)
- `--figure_prefix=X` and `--movie_prefix=X` prefix figure/movie file names with a path or URL
- `--sections_down` and `--sections_up` move all sections down or up (e.g., sections become subsections or chapters).
- `--tables2csv` translates each table to a CSV file.
- `--short_title=X` sets a short title `X` for the document.

Many more options, depending on the output format, are listed in the following sections.

4.7 HTML

Basics. Making an HTML version of a DocOnce file `mydoc.do.txt` is performed by

```
Terminal> doconce format html mydoc
```

The resulting file `mydoc.html` can be loaded into any web browser for viewing.

Typesetting of Code. If the Pygments package (including the `pygmentize` program) is installed, code blocks are typeset with aid of this package. The command-line argument `--no_pygments_html` turns off the use of Pygments and makes code blocks appear with plain (`pre`) HTML tags. The option `--pygments_html_linenos` turns on line numbers in Pygments-formatted code blocks. A specific Pygments style is set by `--pygments_html_style=style`, where `style` can be `default`, `emacs`, `perldoc`, and other valid names for Pygments styles.

Handling of Movies. MP4, WebM, and Ogg movies are typeset with the HTML5 `video` tag and the HTML code tries to load as many versions among MP4, WebM, and Ogg as exist (and the files are loaded in the mentioned order). If just the specified file is to be loaded, use the `--no_mp4_webm_ogg_alternatives` command-line option. Other movie formats, e.g., `.flv`, `.mpeg` and `.avi`, are embedded via the older `embed` tag.

HTML Styles. The HTML style can be defined either in the header of the HTML file, using a named built-in style; in an external CSS file; or in a template file.

An external CSS file `filename` used by setting the command-line argument `--css=filename`. There available built-in styles are specified as `--html_style=name`, where `name` can be

- `solarized`: the famous solarized style (yellowish),
- `blueish`: a simple style with blue headings (default),
- `blueish2`: a variant of *bluish*,
- `bloodish`: as `bluish`, but dark read as color,
- `bootstrap*` or `bootswatch*` in a lot of variants, see `doconce format --help` for a list of all styles.

There is a comprehensive demonstration of almost all available styles!

Using `--css=filename` where `filename` is a non-existing file makes DocOnce write the built-in style to that file. Otherwise the HTML links to the CSS stylesheet in `filename`. Several stylesheets can be specified: `--ccs=file1.css,file2.css,file3.css`.

HTML templates. Templates are HTML files with ready-made headers, footers, and style specifications where plain HTML text can be inserted in "slots" in the template file. Typically, there is a slot `%(main)s` for the main body of text, `%(title)s` for the title, and `%(date)s` for the date. Templates are designed beforehand and `doconce format` puts the translated HTML text into the template to form the complete HTML document.

DocOnce comes with a few ready-made HTML templates. The usage of templates is described in a separate document. That document describes how you your DocOnce-generated HTML file can have any specified layout.

The HTML file can be embedded in a template with your own tailored design, see a tutorial on this topic. The template file must contain valid HTML code and can have three "slots": `%(title)s` for a title, `%(date)s` for a date, and `%(main)s` for the main body of text. The latter is the DocOnce document translated to HTML. The title becomes the first heading in the DocOnce document, or the title (but a title is not recommended when using templates). The date is extracted from the `DATE:` line. With the template feature one can easily embed the text in the look and feel of a website. DocOnce comes with two templates in `bundled/html_styles`. Just copy the directory containing the template and the CSS and JavaScript files to your document directory, edit the template as needed (also check that paths to the `css` and `js` subdirectories are correct - according to how you store the template files), and run

```
Terminal> doconce format html mydoc --html_template=mytemplate.html
```

The template in `style_vagrant` also needs an extra option `--html_style=bootstrap`.

Splitting HTML documents. The `!split` instruction (on separate lines) signifies a pagebreak. A command `doconce split_html` is needed after `doconce format` to actually perform the split. The `doconce split_html` command has several options for setting the type of splitting, type of navigation buttons, etc. Just type `doconce split_html` to see the options. Here is an example with separate links for each page (pagination) at the top and bottom of each page:

```
Terminal> doconce format html mydoc --html_style=bootswatch_journal
Terminal> doconce split_html mydoc --nav_buttontop+bottom --pagination
```

The HTML File Collection. There are usually a range of files needed for an HTML document arising from a DocOnce source. The needed files are listed in `.basename_html_file_collection`, where `basename` is the filestem of the DocOnce file (i.e., the DocOnce source is in `basename.do.txt`).

Filenames. An HTML version of a DocOnce document is often made in different styles, calling for a need to rename the HTML output file. This is conveniently done by the `--html_output=basename` option, where `basename` is the filestem of the associated HTML files. The `.basename_html_file_collection` file lists all the needed files for the HTML document. Here is an example on making three versions of the HTML document: `mydoc_bloodish.html`, `mydoc_solarized`, and `mydoc_vagrant`.

```
Terminal> doconce format html mydoc --html_style=bloodish \
          --html_output=mydoc_bloodish
Terminal> doconce split_html mydoc_bloodish.html
Terminal> doconce format html mydoc --html_style=solarized \
          --html_output=mydoc_solarized \
          --pygments_html_style=perldoc --html_admon=apricot
Terminal> doconce format html mydoc --html_style=vagrant \
          --html_output=mydoc_vagrant --pygments_html_style=default \
          --html_template=templates/my_adapted_vagrant_template.html
Terminal> doconce split_html mydoc_vagrant.html
```

URL to files hosted on GitHub. The generated HTML code will have URLs to files in the DocOnce repo at GitHub. The type of URL is set with the `--html_raw_github_url=...` option:

- `--html_raw_github_url=safe` or `--html_raw_github_url=cdn.rawgit`: safe URL for high traffic production sites (default)
- `--html_raw_github_url=test` or `--html_raw_github_url=rawgit`: recommended URL for low traffic development sites - use this when developing HTML pages and the DocOnce GitHub links in the HTML files are also developed and subject to changes
- `--html_raw_github_url=github` or `--html_raw_github_url=raw.github`: URL directly to the raw GitHub file (`https://raw.github.com/hplgit/doconce/...`) that may fail to load properly in (e.g.) Internet explorer

- `--html_raw_github_url=githubusercontent` or `--html_raw_github_url=raw.githubusercontent`: as the one above, but using `https://raw.githubusercontent.com` instead

Other HTML options. Options for Bootstrap styles:

- `--html_code_style=on,off,inherit,transparent`: control the style of inline verbatim code `code` tags. With `off`, `inherit`, or `transparent` the verbatim text inherits foreground and background color from its surroundings, while `on` (default) means that the typesetting is css-specified. This option is most relevant for Bootstrap styles to avoid the redish typesetting of inline verbatim text.
- `--html_pre_style=on,off,inherit,transparent`: control the style of code blocks in `pre` tags. With `off`, `inherit`, or `transparent` the code blocks inherit foreground and background color from their surroundings, while `on` (default) means that code block colors are css-specified. This option is most relevant for Bootstrap styles to avoid white background in code blocks inside colorful admons.
- `--html_bootstrap_navbar=on,off`: turn the Bootstrap navigation bar on or off.
- `--html_bootstrap_jumbotron=on,off,h2`: turn the jumbotron intro on or off, and govern the size of the document title. Default is `on`, while `h2` means a jumbotron with `h2` (section) size of the title (normally the jumbotron has huge heading fonts so some jumbotrons look better with `h2` typesetting of the document title).
- `--html_quiz_button_text=X`: set a text on the answer button for Bootstrap-style quizzes. Without this option a small icon is used.

Other options:

- `--html_toc_depth=X`: controls the depth of the table of contents in documents. Default value of `X` is 3, meaning chapters, sections, and subsections. `X` as 0 gives the table of contents as a nested list in Bootstrap styles.
- `--html_toc_indent=X`: indent sections/subsections `X` spaces in the table of contents.
- `--html_body_font=`: specify font for text body. The value `?` lists available fonts.
- `--html_heading_font=`: specify font for headings. The value `?` lists available fonts.

- `--html_video_autoplay=True,False`: let videos play automatically (`True`, default) or not (`False`) when the HTML page is loaded.
- `--html_admon=X`: specify typesetting of admonitions. Values of `X` are `colors`, `gray`, `yellow`, `apricot`, `lyx`, `paragraph`. For Bootstrap styles only to other values are legal: `botstrap_panel` or `bootstrap_alert`. See demos for how these look like.
- `--html_admon_bg_color=X`: set the background color in admonitions.
- `--html_admon_bd_color=X`: set the boundary color in admonitions.
- `--html_admon_shadow`: add a shadow effect in admonitions.
- `--html_box_shadow`: add a shadow effect in box environments (`!bbox`).
- `--html_exercise_icon=X`: specify an icon to more easily notify exercises. `X` can be any filename `question_*.png` in the `bundled/html_images` directory in the DocOnce repo. With `X` as `default`, a default icon choice is made, based on the current style.
- `--html_exercise_icon_width=X`: set the width of the exercise icon image to `X` pixels.
- `--exercise_numbering=absolute, chapter`
- `--html_DOCTYPE`: insert `<!DOCTYPE HTML>` at the top of the HTML output file. This is normally recommended, but malformed CSS files will then not be loaded (so by default, the doctype is not specified). This option is necessary for correct rendering of Bootstrap styles in Internet Explorer.
- `--html_links_in_new_window`: open all links as new tabs.
- `--html_figure_hrule=X`: control the use of horizontal rules in figures. `X` is `top` by default; other values are `none` (no rules), `bottom` and `top+bottom`.

4.8 Blog Posts

DocOnce can be used for writing blog posts provided the blog site accepts raw HTML code. Google's Blogger service (`blogger.com` or `blogname.blogspot.com`) is particularly well suited since it also allows extensive LaTeX mathematics via MathJax.

1. Write the text of the blog post as a DocOnce document without any title, author, and date.
2. Generate HTML as described above.

3. Copy the text and paste it into the text area in the blog post (just delete the HTML code that initially pops up in the text area). Make sure the input format is HTML.

See a simple blog example and a scientific report for demonstrations of blog posts at `blogspot.no`.

Warning.

The comment field after the blog post does not recognize MathJax (LaTeX) mathematics or code with indentation. However, using a MathJax bookmarklet, e.g., at http://checkmyworking.com/misc/mathjax-bookmarklet/, one can get the mathematics properly rendered. The comment fields are not suitable for computer code, though, as HTML tags are not allowed.

Notice.

Figure files must be uploaded to some web site and the local filenames name must be replaced by the relevant URL. This is usually done by using the `--figure_prefix=http://project.github.io/...` option to give some URL as prefix to all figure names (a similar `--movie_prefix=` option exists as well).

Changing figure names in a blog post can also be done "manually" by some editing code in the script that compiles the DocOnce document to HTML format:

```
cp mydoc.do.txt mydoc2.do.txt
url="https//raw.github.com/someuser/someuser.github.com"
dir="master/project/dir1/dir2"
for figname in fig1 fig2 fig3; do
  doconce replace "[$figname," "[$site/$dir/$figname.png," \
          mydoc2.do.txt
done
doconce format html mydoc2
# Paste mydoc2.html into a new blog post page
```

Blog posts at Google can also be published automatically through email. A Python program can send the contents of the HTML file to the blog site's email address using the packages `smtplib` and `email`.

WordPress (`wordpress.com`) allows raw HTML code in blogs, but has very limited LaTeX support, basically only formulas. The `--wordpress` option to `doconce` modifies the HTML code such that all equations are typeset in a way that is acceptable to WordPress. Look at a simple doconce example and a scientific report to see blog posts with mathematics and code on WordPress.

Speaking of WordPress, the related project `http://pressbooks.com` can take raw HTML code (from DocOnce, for instance, but use the `--wordpress`

option) and produce very nice-looking books. There is support for LaTeX mathematics as in WordPress blog posts, meaning that one cannot refer to equations.

4.9 Pandoc and Markdown

Output in Pandoc's extended Markdown format results from

```
Terminal> doconce format pandoc mydoc
```

or (equivalent)

```
Terminal> doconce format markdown mydoc
```

The name of the output file is `mydoc.md`. From this format one can go to numerous other formats:

```
Terminal> pandoc -R -t mediawiki -o mydoc.mwk --toc mydoc.md
```

Pandoc supports `latex`, `html`, `odt` (OpenOffice), `docx` (Microsoft Word), `rtf`, `texinfo`, to mention some. The `-R` option makes Pandoc pass raw HTML or LaTeX to the output format instead of ignoring it, while the `--toc` option generates a table of contents. See the Pandoc documentation for the many features of the `pandoc` program.

Markdown to HTML conversion. The HTML output from `pandoc` needs adjustments to provide full support for MathJax LaTeX mathematics, and for this purpose one should use `doconce md2html`:

```
Terminal> doconce format pandoc mydoc
Terminal> doconce m2html mydoc
```

The result `mydoc.html` can be viewed in a browser.

Strict Markdown. The option `--strict_markdown_output` generates plain or strict Markdown without the many extension that Pandoc accepts in Markdown syntax.

GitHub-flavored Markdown. Adding the command-line option `github-md` turns on the GutHub-flavored Markdown dialect, which is used for the issue tracker on GitHub. A special feature is the support of task lists: unnumbered lists with `[x]` (task done) or `[ ]` (task not done). (Tables get typeset directly as HTML and the syntax for code highlighting is different from Pandoc extended Markdown.) Below is a typical response in a GitHub issue tracker where one first quotes the issue and then provides an answer:

```
!bquote
===== Problems with a function =====

There is a problem with the `f(x)` function

!bc pycod
def f(x):
    return 1 + x
!ec
This function should be quadratic.
!equote

OK, this is fixed:

!bc pycod
def f(x, a=1, b=1, c=1):
    return a*x**2 + b*x + c
!ec

===== Updated task list =====

   * [x] Offer an `f(x)` function
   * [ ] Extension to cubic functions
   * [x] Allowing general coefficient in the quadratic function

=== Remaining functionality ===

|------------------------------------------------------------|
| function | purpose                          | state |
|----l-----------l--------------------------l---------------|
|  `g(x)`  | Compute the Gaussian function. | Formula ready. |
|  `h(x)`  | Heaviside function.            | Formula ready. |
|  `I(x)`  | Indicator function.            | Nothing done yet. |
|------------------------------------------------------------|
```

Say this text is stored in a file `mycomments.do.txt`. Running

```
Terminal> doconce format pandoc mycomments --github_md
```

produces the Markdown file `mycomments.md`, which can be pasted into the Write field of the GitHub issue tracker. Turning on Preview shows the typesetting of the quote, compute code, inline verbatim, headings, the task list, and the table.

MultiMarkdown. The option `--multimarkdown_output` generates the MultiMarkdown version of Markdown (as opposed to Pandoc-extended Markdown (default), strict Markdown, or GitHub-flavored Markdown).

Strapdown rendering of Markdown text. Strapdown is a tool that can render Markdown text nicely in a web browser by just inserting an HTML header and footer in the Markdown file and load the file into a browser. The option `--strapdown` outputs the relevant header and footer. The output file must be renamed such that it gets the extension `.html`:

```
Terminal> doconce format pandoc mydoc --strict_markdown_output \
          --strapdown --bootstrap_bootwatch_theme=slate
Terminal> mv mydoc.md mydoc.html
```

The `--bootstrap_bootwatch_theme=theme` option is used to choose a Bootswatch theme whose names are found on the Strapdown page.

Using Pandoc to go from LaTeX to MS Word or HTML. Pandoc is useful to go from LaTeX mathematics to, e.g., HTML or MS Word. There are two ways (experiment to find the best one for your document): `doconce format pandoc` and then translating using `doconce md2latex` (which runs `pandoc`), or `doconce format latex`, and then going from LaTeX to the desired format using `pandoc`. Here is an example on the latter strategy:

```
Terminal> doconce format latex mydoc
Terminal> doconce ptex2tex mydoc
Terminal> doconce replace '\Verb!' '\verb!' mydoc.tex
Terminal> pandoc -f latex -t docx -o mydoc.docx mydoc.tex
```

When we go through `pandoc`, only single equations, `align`, or `align*` environments are well understood for output to HTML.

Note that DocOnce applies the `Verb` macro from the `fancyvrb` package while `pandoc` only supports the standard `verb` construction for inline verbatim text. Moreover, quite some additional `doconce replace` and `doconce subst` edits might be needed on the `.mkd` or `.tex` files to successfully have mathematics that is well translated to MS Word. Also when going to reStructuredText using Pandoc, it can be advantageous to go via LaTeX.

4.10 LaTeX

Notice.

XeLaTeX and PDFLaTeX are used very much in the same way as standard LaTeX. The minor differences are described in separate sections of the documentation of the DocOnce to LaTeX translation.

Making a LaTeX file `mydoc.tex` from `mydoc.do.txt` can be done in two ways:

1. direct translation to a `.tex` file
2. translation to a `.p.tex` file

In the latter case, one must apply the `ptex2tex` program or the simplified `doconce ptex2tex` program to translate the `.p.tex` file to a plain `.tex` file. This step involves the specification of how blocks of verbatim code should be typeset in LaTeX. Before 2015, DocOnce always translated to the `.p.tex` syntax and required the use of `ptex2tex` or `doconce ptex2tex`. Now, one can choose a direct translation, which is simpler and actually more versatile than even using the `ptex2tex` program.

Direct translation is specified by the `--latex_code_style=` command-line option. A separate document describes how this option is used and the demonstrates various possibilities that are available.

Here, we describe the old translation via a `.p.tex` file, i.e., first we compile the DocOnce source to the `ptex2tex` format, and then we compile the `ptex2tex` format to standard LaTeX. The `ptex2tex` format can be viewed as an extended LaTeX. For DocOnce users, the `ptex2tex` format essentially means that the file consists of

1. `if-else` statements for the `preprocess` processor such that LaTeX constructions can be activated or deactivated, and
2. all code environments can be typeset according to a `.ptex2tex.cfg` configuration file.

Point 2 is only of interest if you aim to use a special computer code formatting that requires you to use a configuration file and the `ptex2tex` program.

The reason for generating `ptex2tex` and not standard LaTeX directly from DocOnce was that the `ptex2tex` format shows a range of possible LaTeX constructions for controlling the layout. It can be instructive for LaTeX users to look at this code before choosing specific parts for some desired layout. Experts may also want to edit this code (which should be automated by a script such that the edits can be repeated when the DocOnce source is modified, see Step 2b below). (Direct control of the LaTeX layout in the `doconce format` program would not spit out alternative LaTeX constructs as is now done through the `ptex2tex` step.)

Going from `ptex2tex` format to standard LaTeX format is enabled by either the `ptex2tex` program or DocOnce's (simplified) version of it: `doconce ptex2tex`. Details are given below.

Information on typesetting of inline verbatim.

The `ptex2tex` and the `doconce ptex2tex` programs take inline verbatim code, typeset with backticks in DocOnce, and translate this to

```
\Verb!text!
```

Thereafter, if `text` does not contain illegal characters for the `\texttt` command, the latter is used instead since then LaTeX can insert linebreaks in the inline verbatim text and hence avoid overfull hboxes.

Step 1. Filter the doconce text to the `ptex2tex` pre-LaTeX form `mydoc.p.tex`:

```
Terminal> doconce format latex mydoc
```

Or filter the doconce text directly to valid LaTeX:

```
Terminal> doconce format latex mydoc --latex_code_style=vrb
```

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files `newcommands.tex`, `newcommands_keep.tex`, or `newcommands_replace.tex` (see Section 3.5). If these files are present, they are included in the LaTeX document so that your commands are defined.

An option `--device=paper` makes some adjustments for documents aimed at being printed. For example, links to web resources are associated with a footnote listing the complete web address (URL). (Very long URLs in footnotes can be shortened using services such as http://goo.gl/, http://tinyurl.com/, and https://bitly.com/.) The default, `--device=screen`, creates a PDF file for reading on a screen where links are just clickable.

There are many additional options (run `doconce format --help` and look for options starting with `--latex` to get a more verbose description):

- `--latex_code_style=lst,vrb,pyg`
- `--latex_font=helvetica,palatino`
- `--latex_papersize=a4,a6`
- `--latex_bibstyle=plain` (name of BibTeX style)
- `--latex_title_layout=titlepage, std, beamer, doconce_heading, Springer_collection`
- `--latex_style=std, Springer_lncse, Springer_llncs, Springer_lnup, Springer_T2, sia`
- `--latex_list_of_exercises=loe,toc,none` (LaTeX list of exercises, integrated into the table of contents, or no list)
- `--latex_fancy_header` (chapter/section headings at top of pages, style depends on value of `--latex_section_headings`)
- `--latex_section_headings=std,blue,strongblue,gray,gray-wide` (standard LaTeX, blue headings, strong blue headings, white in gray box, white in gray box that fills the page width)
- `--latex_colored_table_rows=blue, gray, no` (color of every two lines in tables)
- `--latex_todonotes` (inline comments typeset as “bubbles”)
- `--latex_double_spacing` (to ease hand-writing between the lines)
- `--latex_line_numbers` (to ease references to sentences)
- `--latex_labels_in_margin` (name of section, equation, citation labels in the margin)
- `--latex_preamble=filename` (user-specific preamble)
- `--latex_admon=mdfbox, graybox2, grayicon, yellowicon, paragraph, colors1, colors2`
- `--latex_admon_color=0.34,0.02,0.8` (background color in admons)

- `--latex_admon_envir_map=2` (code environment names in admons)
- `--exercise_numbering=absolute, chapter`
- `--latex_movie=media9, href, multimedia, movie15` (control typesetting of movies)
- `--latex_movie_controls=on`
- `--latex_external_movie_viewer` (for `movie15` package)
- `--xelatex` (prepare for XeLaTeX)

The overall LaTeX style is much governed by `--latex_title_layout` and `--latex_style`. For the former, `titlepage` gives a separate title page; `std` is just standard LaTeX handling of title, author, and date; `doconce_heading` is a more modern heading, `Springer_collection` is used with `--latex_style=Springer_lncse` for an edited book; `beamer` is needed if the DocOnce document is to be translated to LaTeX Beamer slides. For `--latex_style`, `std` gives standard LaTeX behavior; `Springer_lncse` is for Springer's LNCSE book series style (to be used with `--latex_title_layout=Springer_collection` if the book is an edited book); `Springer_llncs` is for Springer's Lecture Notes in Computer Science series (normally an edited book that also requres `--latex_title_layout=Springer_collection`); `Springer_lnup` for Springer's Lecture Notes for Undergraduate Physics books, `Springer_T2` for Springer's T2 book layout, `siamltex`, for the LaTeX style of papers in standard SIAM journals (also used far beyond SIAM journals and requires the stylefiles `siamltex.cls` and `siam10.clo`), `siamltexmm` for the new multimedia SIAM journal style (requires `siamltex.cls` and `siam10.clo`), `elsevier` for the style of papers to be submitted to Elsevier journals (`--latex_elsevier_journal=` can be used to set the journal name, and the style requires `elsarticle.cls` and `elsarticle-num.bst`).

The style of verbatim blocks of computer code is specified by `--latex_code_style=X`, where `X` can be set in a very flexible way. There are three main values, corresponding to three LaTeX tools for verbatim type setting:

- `vrb` for plain `Verbatim` style (`fancyvrb` LaTeX package)
- `pyg` for the Pygments style (`mintex` LaTeX package)
- `lst` for the Listings styles (`listingsutf8` LaTeX package)

A separate demo explains the many possible settings of `X`. Popular choices are minimalistic plain verbatim,

```
--latex_code_style=vrb
```

maybe with an added light blue background color,

```
--latex_code_style=vrb-blue1
```

or the default Pygments style,

```
--latex_code_style=pyg
```

or the Listings-based style with yellow background color

```
--latex_code_style=lst-yellow2
```

It is easy to specify different styles for different code environments, say blue background with plain verbatim style for code but a special terminal window for the `sys` environment:

```
"--latex_code_style=default:vrb-blue1@sys:vrb[frame=lines,label=\\fbox{{\tiny Terminal}},frames
```

During development of a manuscript, may prefer line numbers, double line spacing, frequent use of inline comments, and label names printed in the margin. This is enabled by the options `--latex_line_numbers --latex_double_spacing --latex_todonc` One may also (automatically) edit the `final` argument in the `documentclass` heading to `draft` as this will mark overful lines (hboxes).

Another useful option for LaTeX documents is `--no_ampersand_quote`, which prevents ampersands from getting a backskash. This is necessary if one inserts native latex code for tables inside `% if FORMAT in ('latex', 'pdflatex'):` (or similar `preprocess` syntax) tests.

Step 2. In case you *did not* specify the `--latex_code_style=` option, you must run `ptex2tex` (if you have installed the Python ptex2tex package) to make a standard LaTeX file,

```
Terminal> ptex2tex mydoc
```

If you do not have `ptex2tex`, or do not bother to make the required configuration file for `ptex2tex` (you may of course rely on the default file), a (simplified) version of `ptex2tex` that comes with DocOnce can be run:

```
Terminal> doconce ptex2tex mydoc
```

The `ptex2tex` command can set two preprocessor variables:

- `PREAMBLE` to turn the LaTeX preamble on or off (i.e., complete document or document to be included elsewhere - and note that the preamble is only included if the document has a title, author, and date)
- `MINTED` for inclusion of the minted package for typesetting of code with the Pygments tool (which requires `latex` or `pdflatex` to be run with the `-shell-escape` option); not used for `doconce ptex2tex` only in the `ptex2tex` program

If you are not satisfied with the generated DocOnce preamble, you can provide your own preamble by adding the command-line option `--latex_preamble=myfile`. In case `myfile` contains a documentclass definition, DocOnce assumes that the file contains the *complete* preamble you want (not that all the packages

listed in the default preamble are required and must be present in `myfile`). Otherwise, `myfile` is assumed to contain *additional* LaTeX code to be added to the DocOnce default preamble.

The `ptex2tex` tool makes it possible to easily switch between many different fancy formattings of computer code in LaTeX documents. After any `!bc` command in the DocOnce source you can insert verbatim block styles as defined in your `.ptex2tex.cfg` file, e.g., `!bc sys` for a terminal session, where `sys` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeTerminal`). There are about 40 styles to choose from, and you can easily add new ones.

The `doconce ptex2tex` allows specifications of code environments as well. Here is an example:

```
Terminal> doconce ptex2tex mydoc \
          "sys=\begin{quote}\begin{verbatim}@\end{verbatim}\end{quote}" \
          fpro=minted fcod=minted shcod=Verbatim envir=ans:nt
```

Note that `@` must be used to separate the begin and end LaTeX commands, unless only the environment name is given (such as `minted` above, which implies `\begin{minted}{fortran}` and `\end{minted}` as begin and end for blocks inside `!bc fpro` and `!ec`). Specifying `envir=ans:nt` means that all other environments are typeset with the `anslistings.sty` package, e.g., `!bc cppcod` will then result in `\begin{c++}`. A predefined shortcut as in `shcod=Verbatim-0.85` results in denser vertical spacing (baselinestretch 0.85 in LaTeX terminology), and `shcod=Verbatim-indent` implies indentation of the verbatim text. Alternatively, one can provide all desired parameters `\begin{Verbatim}` instruction using the syntax illustrated for the `sys` environments above.

If no environments like `sys`, `fpro`, or the common `envir` are defined on the command line, the plain `\begin{Verbatim}` and `\end{Verbatim}` instructions are used.

Step 2b (optional). Edit the `mydoc.tex` file to your needs. For example, you may want to substitute `section` by `section*` to avoid numbering of sections, you may want to insert linebreaks (and perhaps space) in the title, etc. This can be automatically edited with the aid of the `doconce replace` and `doconce subst` commands. The former works with substituting text directly, while the latter performs substitutions using regular expressions. You will use `doconce replace` to edit `section{` to `section*{`:

```
Terminal> doconce replace 'section{' 'section*{' mydoc.tex
```

For fixing the line break of a title, you may pick a word in the title, say "Using", and insert a break after than word. With `doconce subst` this is easy employing regular expressions with a group before "Using" and a group after:

```
Terminal> doconce subst 'title\{(.+)Using (.+)\}' \
          'title{\g<1> \\\\ [1.5mm] Using \g<2>' mydoc.tex
```

A lot of tailored fixes to the LaTeX document can be done by an appropriate set of text replacements and regular expression substitutions. You are anyway encourged to make a script for generating PDF from the LaTeX file so the `doconce subst` or `doconce replace` commands can be put inside the script.

Step 3. Compile `mydoc.tex` and create the PDF file:

```
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibtex mydoc      # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
```

See the next two sections for compilation with XeLaTeX or PDFLaTeX.

If one wishes to use the minted LaTeX package for typesetting code blocks (`Minted_Python`, `Minted_Cpp`, etc., in `ptex2tex` specified through the `*pro` and `*cod` variables in `.ptex2tex.cfg` or `$HOME/.ptex2tex.cfg`), the minted LaTeX package is needed. This package is automatically included by `doconce ptex2tex` if the minted style is used, while you have to include the `-DMINTED` preprocessor option when running the `ptex2tex` program:

```
Terminal> ptex2tex -DMINTED mydoc
```

If the minted style is used, `latex` (or `pdflatex` or `xelatex`) *must* be run with the `-shell-escape` option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibtex mydoc      # if bibliography
Terminal> latex -shell-escape mydoc
Terminal> dvipdf mydoc
```

4.11 PDFLaTeX

Running `pdflatex` instead of `latex` follows almost the same steps, but the start is

```
Terminal> doconce format latex mydoc
```

Then `ptex2tex` is run as explained above, and finally

```
Terminal> pdflatex -shell-escape mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibtex mydoc      # if bibliography
Terminal> pdflatex -shell-escape mydoc
```

4.12 XeLaTeX

XeLaTeX is an alternative to PDFLaTeX and is run in almost the same way, except for the `--xelatex` flag to `doconce format`:

```
Terminal> doconce format pdflatex mydoc --xelatex
Terminal> doconce ptex2tex mydoc
Terminal> xelatex mydoc
```

4.13 From PDF to e-book formats

PDF (as generated from LaTeX above) can be read on most devices today. However, for Kindle and other devices specialized for e-books you need to convert to their format. The Calibre program can produce epub, mobi, and other e-book formats from PDF, see a description.

4.14 Microsoft Word or LibreOffice

Transforming DocOnce files to Word format is best done with the aid of `pandoc`. A standard way is to first generate the Markdown format (`doconce format pandoc`) and then use `pandoc` to generate a `.docx` file:

```
Terminal> doconce format pandoc mydoc
Terminal> pandoc -t docx -o mydoc.docx mydoc.md
```

The transformation works well for simple text files, but LaTeX mathematics does not work.

4.15 Jupyter (IPython) Notebooks

DocOnce can generate json files for the Jupyter Notebook:

```
Terminal> doconce format ipynb mydoc  # results in mydoc.ipynb
```

Hidden code blocks. It is no guarantee that the notebook can be executed. For example, having the code

```
print sys.version
```

will not execute unless `sys` is imported. While a book may show such code and skip (potentially tedious) initializing statements, they must be present in the notebook. To this end, use the `!bc *hid` environment for hidden code. In the present example, we use `!bc pyhid` to specify Python code that needs to be executed, but that should normally be hidden (other formats, with the exception of certain interactive Sphinx documents, will hide such code).

```
!bc pyhid
import sys
!ec
```

The notebook will feature the `import sys` statement in a cell prior to the `print sys.version` cell, and the latter will work.

Similarly, if you import from your own modules, say `from mymod import hello`, the `mymod.py` must be accessible for the notebook. Suppose this file is in `src-test/mymod.py`. Then you need to add `src-test` to `sys.path` for the import statement to work:

```
!bc pyhid
sys.path.append('src-test')
!ec
```

Displaying code as plain text instead of executable cells. Some code blocks may just be there for explanation and are not meant to be executed. These can be marked by `!bc pycod-t` (or `!bc Xcod-t` for any supported programming language `X`):

```
!bc pycod-t
if isinstance(myvar, float):
    raise TypeError('myvar must be array, not %s' % type(myvar))
!ec
```

The code segment above will then be typeset as verbatim text and not an executable cell, and there is no need to worry about a missing definition of `myvar` (which would cause problems in an executable cell).

Interactive sessions with the `pyshell` or `ipy` environment will by default be broken up into many cells such that each output command ends a cell. By executing the cells, the input *and* output from the session is recovered. This is usually the behavior that is wanted, but there is an option `--ipynb_split_pyshell=off` that can be used to typeset the entire session with all input but no output in one cell (`print` statements will lead to output, but plain dumping of a variable will not lead to output like it does in a Python shell).

To have an interactive session typeset with input and output in plain text, use the `-t` extension to the environment: `pyshell-t` and `ipy-t`.

Figures. As with HTML files, you need to ensure that the notebook has access to figures and source code as requested.

Figures in notebooks can be typeset in various ways, specified by the `--ipynb_figure=` option, with the following values:

- `md`: plain Markdown syntax for a figure, with no possibility to adjust the size (default)
- `imgtag`: `<img ...>` tag in HTML taking the specified width into account
- `Image`: Python notebook cell with `Image` object

Movies. Typesetting of movies is specified by `--ipynb_movie=`, and valid options are

- `md`: raw HTML code with `iframe` tag - not relevant for the notebook
- `HTML`: raw HTML code with `iframe` tag embedded in the `HTML` object from the notebook (default)
- `HTML-YouTube`: as `HTML` but use an `IPython.display.YouTubeVideo` object to display YouTube videos
- `ipynb`: use `IPython.display.YouTubeVideo` object for YouTube videos, and use an `HTML` object with `video` tag for local movies

Admonitions. Typesetting of admonition is rather primitive in notebooks. We offer these different choices, set by the option `--ipynb_admon=`:

- `quote`: typeset admon as Markdown quote (special font and gray vertical bar on the left)
- `paragraph`: typeset admon as a plain paragraph with a heading if any (default)
- `hrule`: use a horozontal rule to surround the heading and the text

Note that quotes in `!bc quote` environments are always typeset as Markdown quotes.

4.16 Plain ASCII Text

We can go from DocOnce "back to" plain untagged text suitable for viewing in terminal windows, inclusion in email text, or for insertion in computer source code:

```
Terminal> doconce format plain mydoc.do.txt  # results in mydoc.txt
```

4.17 reStructuredText

Going from DocOnce to reStructuredText gives a lot of possibilities to go to other formats. First we filter the DocOnce text to a reStructuredText file `mydoc.rst`:

```
Terminal> doconce format rst mydoc.do.txt
```

We may now produce various other formats:

```
Terminal> rst2html.py  mydoc.rst > mydoc.html # html
Terminal> rst2latex.py mydoc.rst > mydoc.tex  # latex
Terminal> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Terminal> rst2odt.py   mydoc.rst > mydoc.odt  # OpenOffice
```

The OpenOffice file `mydoc.odt` can be loaded into OpenOffice and saved in, among other things, the RTF format or the Microsoft Word format. However, it is more convenient to use the program `unovonv` to convert between the many formats OpenOffice supports *on the command line*. Run

```
Terminal> unoconv --show
```

to see all the formats that are supported. For example, the following commands take `mydoc.odt` to Microsoft Office Open XML format, classic MS Word format, and PDF:

```
Terminal> unoconv -f ooxml mydoc.odt
Terminal> unoconv -f doc mydoc.odt
Terminal> unoconv -f pdf mydoc.odt
```

Remark about Mathematical Typesetting. At the time of this writing, there is no easy way to go from DocOnce and LaTeX mathematics to reST and further to OpenOffice and the "MS Word world". Mathematics is only fully supported by `latex` as output and to a wide extent also supported by the `sphinx` output format. Some links for going from LaTeX to Word are listed below.

- http://ubuntuforums.org/showthread.php?t=1033441
- http://tug.org/utilities/texconv/textopc.html
- http://nileshbansal.blogspot.com/2007/12/latex-to-openofficeword.html

4.18 Sphinx

Sphinx documents demand quite some steps in their creation. We have automated most of the steps through the `doconce sphinx_dir` command:

```
Terminal> doconce sphinx_dir author="authors' names" \
          title="some title" version=1.0 dirname=sphinxdir \
          theme=mytheme mydoc
```

The keywords `author`, `title`, and `version` are used in the headings of the Sphinx document. By default, `version` is 1.0 and the script will try to deduce authors and title from the doconce file `mydoc.do.txt` The default value of `dirname` is `sphinx-rootdir`. The `theme` keyword is used to set the theme for design of HTML output from Sphinx (the default theme is `'default'`).

One often just runs the simple command

```
Terminal> doconce sphinx_dir mydoc
```

which creates the Sphinx directory `sphinx-rootdir` with relevant files.

The `doconce sphinx_dir` command generates a script `automake_sphinx.py` for compiling the Sphinx document into an HTML document. Run

```
Terminal> python automake_sphinx.py
```

As the output also tells, you can see the Sphinx HTML version of the document by running

```
Terminal> google-chrome sphinx-rootdir/_build/html/index.html
```

or loading the `index.html` file manually into your favorite web browser.

If you cycle through editing the DocOnce file and watching the HTML output, you should observe that `automake_sphinx.py` does not recompile the DocOnce file if the Sphinx `.rst` version already exists. In each edit-and-watch cycle do

```
Terminal> rm mydoc.rst; python automake_sphinx.py
```

Tip.

If you are new to Sphinx and end up producing quite some Sphinx documents, you are encouraged to read the Sphinx documentation and study the `automake_sphinx.py` file. Maybe you want to do things differently.

The following paragraphs describes the many possibilities for steering the Sphinx output.

Links. The `automake_sphinx.py` script copies directories named `fig*` over to the Sphinx directory so that figures are accessible in the Sphinx compilation. It also examines `MOVIE:` and `FIGURE:` commands in the DocOnce file to find other image files and copies these too. I strongly recommend to put files to which there are local links (not `http:` or `file:` URLs) in a directory named `_static`. The `automake_sphinx.py` copies `_static*` to the Sphinx directory, which guarantees that the links to the local files will work in the Sphinx document.

There is a utility `doconce sphinxfix_localURLs` for checking links to local files and moving the files to `_static` and changing the links accordingly. For example, a link to `dir1/dir2/myfile.txt` is changed to `_static/myfile.txt` and `myfile.txt` is copied to `_static`. However, I recommend instead that you manually copy files to `_static` when you want to link to them, or let your script which compiles the DocOnce document do it automatically.

Themes. DocOnce comes with a rich collection of HTML themes for Sphinx documents, much larger than what is found in the standard Sphinx distribution. Additional themes include `agni`, `basicstrap`, `bootstrap`, `cloud`, `fenics`, `fenics_minimal`, `flask`, `haiku`, `impressjs`, `jal`, `pylons`, `redcloud`, `scipy_lectures`, `slim-agogo`, and `vlinux-theme`.

All the themes are packed out in the Sphinx directory, and the `doconce sphinx_dir` insert lots of extra code in the `conf.py` file to enable easy specification and customization of themes. For example, modules are loaded for the additional themes that come with DocOnce, code is inserted to allow customization of the look and feel of themes, etc. The `conf.py` file is a good starting point for fine-tuning your favorite team, and your own `conf.py` file can later be supplied and used when running `doconce sphinx_dir`: simply add the command-line option `conf.py=conf.py`.

A script `make-themes.sh` can make HTML documents with one or more themes. For example, to realize the themes `fenics`, `pyramid`, and `pylon` one writes

```
Terminal> ./make-themes.sh fenics pyramid pylon
```

The resulting directories with HTML documents are `_build/html_fenics` and `_build/html_pyramid`, respectively. Without arguments, `make-themes.sh` makes

all available themes (!). With `make-themes.sh` it is easy to check out various themes to find the one that is most attractive for your document.

You may supply your own theme and avoid copying all the themes that come with DocOnce into the Sphinx directory. Just specify `theme_dir=path` on the command line, where `path` is the relative path to the directory containing the Sphinx theme. You must also specify a configure file by `conf.py=path`, where `path` is the relative path to your `conf.py` file.

Example. Say you like the `scipy_lectures` theme, but you want a table of contents to appear *to the right*, much in the same style as in the `default` theme (where the table of contents is to the left). You can then run `doconce sphinx_dir`, invoke a text editor with the `conf.py` file, find the line `html_theme == 'scipy_lectures'`, edit the following `nosidebar` to `false` and `rightsidebar` to `true`. Alternatively, you may write a little script using `doconce replace` to replace a portion of text in `conf.py` by a new one:

```
doconce replace "elif html_theme == 'scipy_lectures':
    html_theme_options = {
        'nosidebar': 'true',
        'rightsidebar': 'false',
        'sidebarbgcolor': '#f2f2f2',
        'sidebartextcolor': '#20435c',
        'sidebarlinkcolor': '#20435c',
        'footerbgcolor': '#000000',
        'relbarbgcolor': '#000000',
    }" "elif html_theme == 'scipy_lectures':
    html_theme_options = {
        'nosidebar': 'false',
        'rightsidebar': 'true',
        'sidebarbgcolor': '#f2f2f2',
        'sidebartextcolor': '#20435c',
        'sidebarlinkcolor': '#20435c',
        'footerbgcolor': '#000000',
        'relbarbgcolor': '#000000',
    }" conf.py
```

Obviously, we could also have changed colors in the edit above. The final alternative is to save the edited `conf.py` file somewhere and reuse it the next time `doconce sphinx_dir` is run

```
doconce sphinx_dir theme=scipy_lectures \
                   conf.py=../some/path/conf.py mydoc
```

RunestoneInteractive books. The `doconce format sphinx` command accepts an option `--runestone` for generating RunestoneInteractive books (which build on Sphinx). You must run the generated `automake_sphinx.py` also with a `--runestone` option to generate these type of documents.

The manual Sphinx procedure. If it is not desirable to use the autogenerated scripts explained above, here is the complete manual procedure of generating a Sphinx document from a file `mydoc.do.txt`.

Step 1. Translate DocOnce into the Sphinx format:

```
Terminal> doconce format sphinx mydoc
```

Step 2. Create a Sphinx root directory either manually or by using the interactive `sphinx-quickstart` program. Here is a scripted version of the steps with the latter:

```
mkdir sphinx-rootdir
sphinx-quickstart <<EOF
sphinx-rootdir
n
_
Name of My Sphinx Document
Author
version
version
.rst
index
n
y
n
n
n
n
y
n
n
y
y
y
EOF
```

The autogenerated `conf.py` file may need some edits if you want to specific layout (Sphinx themes) of HTML pages. The `doconce sphinx_dir` generator makes an extended `conv.py` file where, among other things, several useful Sphinx extensions are included.

Step 3. Copy the `mydoc.rst` file to the Sphinx root directory:

```
Terminal> cp mydoc.rst sphinx-rootdir
```

If you have figures in your document, the relative paths to those will be invalid when you work with `mydoc.rst` in the `sphinx-rootdir` directory. Either edit `mydoc.rst` so that figure file paths are correct, or simply copy your figure directories to `sphinx-rootdir`. Links to local files in `mydoc.rst` must be modified to links to files in the `_static` directory, see comment above.

Step 4. Edit the generated `index.rst` file so that `mydoc.rst` is included, i.e., add `mydoc` to the `toctree` section so that it becomes

```
.. toctree::
   :maxdepth: 2

   mydoc
```

(The spaces before `mydoc` are important!)

Step 5. Generate, for instance, an HTML version of the Sphinx source:

```
make clean   # remove old versions
make html
```

Sphinx can generate a range of different formats: standalone HTML, HTML in separate directories with `index.html` files, a large single HTML file, JSON files, various help files (the qthelp, HTML, and Devhelp projects), epub, LaTeX, PDF (via LaTeX), pure text, man pages, and Texinfo files.

Step 6. View the result:

```
Terminal> firefox _build/html/index.html
```

Note that verbatim code blocks can be typeset in a variety of ways depending the argument that follows `!bc`: `cod` gives Python (`code-block:: python` in Sphinx syntax) and `cppcod` gives C++, but all such arguments can be customized both for Sphinx and LaTeX output.

4.19 Wiki Formats

There are many different wiki formats, but DocOnce only supports three: Google-code wiki, MediaWiki, and Creole Wiki. These formats are called `gwiki`, `mwiki`, and `cwiki`, respectively. Transformation from DocOnce to these formats is done by

```
Terminal> doconce format gwiki mydoc.do.txt
Terminal> doconce format mwiki mydoc.do.txt
Terminal> doconce format cwiki mydoc.do.txt
```

The produced MediaWiki can be tested in the sandbox of wikibooks.org. The format works well with Wikipedia, Wikibooks, and ShoutWiki, but not always well elsewhere (see this example).

Large MediaWiki documents can be made with the Book creator. From the MediaWiki format one can go to other formats with aid of mwlib. This means that one can easily use DocOnce to write Wikibooks and publish these in PDF and MediaWiki format, while at the same time, the book can also be published as a standard LaTeX book, a Sphinx web document, or a collection of HTML files.

The Googlecode wiki document, `mydoc.gwiki`, is most conveniently stored in a directory which is a clone of the wiki part of the Googlecode project. This is far easier than copying and pasting the entire text into the wiki editor in a web browser. Note that Google decided to close down its Googlecode service in 2015.

When the DocOnce file contains figures, each figure filename must in the `.gwiki` file be replaced by a URL where the figure is available. There are instructions in the file for doing this. Usually, one performs this substitution automatically (see next section).

4.20 Google Docs

Google Docs are normally made online in the interactive editor. However, you may upload a DocOnce document to Google Docs. This requires transforming the DocOnce document to one of the accepted formats for Google Docs:

- OpenOffice: `doconce format rst` and then run `rst2odt` (or `rst2odt.py`). Upload the `.odt` file, click *Open...* in Google Drive and choose *Google Docs* as viewer.
- MS Word: `doconce format pandoc` and then run `pandoc` to produce a `.docx` file that can be uploaded to Google Drive and opened in Google Docs.
- RTF: `doconce format pandoc` and then run `pandoc` to produce a `.rtf` file that can be uploaded to Google Drive and opened. Another possibility is to run `doconce format latex` and then latex2rtf (the support of mathematics has gotten worse).
- Plain text: `doconce format plain`. Upload the `.txt` file to Google Drive and open in Google Docs.
- HTML: `doconce format html`. Upload the `.html` file and open in Google Docs. Complicated HTML files can be misinterpreted by Google Docs.

This is not yet much tested. It remains to see how code becomes in Google Docs. Support for mathematics is probably impossible until Google Docs can import LaTeX files, but LaTeX mathematics can be embedded in Google Docs and the googledoc2latex script can convert a Google document to LaTeX.

4.21 Tweaking the DocOnce Output

Occasionally, one would like to tweak the output in a certain format from DocOnce. One example is figure filenames when transforming DocOnce to reStructuredText. Since DocOnce does not know if the `.rst` file is going to be filtered to LaTeX or HTML, it cannot know if `.eps` or `.png` is the most appropriate image filename. The solution is to use a text substitution command or code with, e.g., sed, perl, python, or scitools subst, to automatically edit the output file from DocOnce. It is then wise to run DocOnce and the editing commands from a script to automate all steps in going from DocOnce to the final format(s). The `make.sh` files in `docs/manual` and `docs/tutorial` constitute comprehensive examples on how such scripts can be made.

5 Options for the `doconce` commands

5.1 `doconce format` command-line options

The transformation of a DocOnce source to various format is done with the `doconce format` command, which has *a lot* of command-line options. These are printed out by `doconce format --help`. The output is listed here for convenience.

```
Terminal> doconce format --help

doconce format X doconcefile

where X can be any of the formats
html, latex, pdflatex, rst, sphinx, plain, gwiki, mwiki, cwiki,
pandoc, epytext.


--help

Print all options to the doconce program.


--debug

Write a debugging file _doconce_debugging.log with lots
of intermediate results


--no_abort

Do not abort the execution if syntax errors are found.


--verbose=...

Write progress of intermediate steps if they take longer than X seconds.
0: X=15
1: X=5
2: 0.5


--syntax_check=...

Values: on/off. Turns on/off fix of illegal constructions and the syntax check
(may be time consuming for large books).


--skip_inline_comments

Remove all inline comments of the form [ID: comment].


--exercise_numbering=...

absolute: exercises numbered as 1, 2, ... (default)
chapter: exercises numbered as 1.1, 1.2, ... , 3.1, 3.2, ..., B.1, B.2, etc.
         with a chapter or appendix prefix.
```

```
--exercises_in_zip

Place each exercises as an individual DocOnce file in a zip archive.


--exercises_in_zip_filename=...

Filenames of individual exercises in zip archive.
logical: use the (first) logical filename specified by file=...
number:  use either absolute exercise number or chapter.localnumber.


--encoding=...

Specify encoding (e.g., latin1 or utf-8).


--no_ampersand_quote

Turn off special treatment of ampersand (&). Needed, e.g., when native latex code for tables are inserted


--no_mako

Do not run the Mako preprocessor program.


--no_preprocess

Do not run the Preprocess preprocessor program.


--mako_strict_undefined

Make Mako report on undefined variables.


--no_header_footer

Do not include header and footer in (LaTeX and HTML) documents.


--no_emoji

Remove all emojis.


--runestone

Make a RunestoneInteractive version of a Sphinx document.


--max_bc_linelength=...

Strip lines in !bc environments that are longer than specified
(to prevent too long lines). Default: None (no length restriction).


--keep_pygments_html_bg
```

```
Do not allow change of background in code blocks in HTML.


--minted_latex_style=...

Specify the minted style to be used for typesetting code in LaTeX.
See pygmetize -L styles for legal names.


--pygments_html_style=...

Specify the minted/pygments style to be used for typesetting code
in HTML.
Default: default (other values: monokai, manni, rrt, perldoc,
borland, colorful, murphy, trac, tango, fruity, autumn, emacs,
vim, pastie, friendly, native, see pygmentize -L styles).
none, no, off: turn off pygments to typeset computer code in HTML,
use plain <pre> tags.
highlight.js: use highlight.js syntax highlighting, not pygments.


--pygments_html_linenos

Turn on line numbers in pygmentized computer code in HTML.
(In LaTeX line numbers can be added via doconce subst or
doconce replace such that the verbatim environments get
the linenos=true parameter.)


--html_output=...

Alternative basename of files associated with the HTML format.


--html_style=...

Name of theme for HTML style:
plain, blueish, blueish2, bloodish, tactile-black, tactile-red, rossant
solarized, solarized2_light, solarized2_dark,
bootstrap, bootswatch,
bootstrap_X,  X=bloodish, blue, bluegray, brown, cbc, FlatUI, red,
bootswatch_X, X=cerulean, cosmo, flatly, journal, lumen, readable,
                simplex, spacelab, united, yeti
                (dark:) amelia, cyborg, darkly, slate, spruce,
                superhero (demos at bootswatch.com)


--html_template=...

Specify an HTML template with header/footer in which the doconce
document is embedded. (Often preferred to run with --no_title)


--no_title

Comment out TITLE, AUTHOR, DATE.
Often used with HTML templates.


--html_code_style=...
```

```
off, inherit, or transparent: enable normal inline verbatim font
where foreground and background color is inherited from the
surroundnings (e.g., to avoid the red Boostrap color).
Default: on (use the css-specified typesetting of <pre> tags).
NOTE: the naming "html_code_style" is not optimal: it has nothing
to do with code block style, but the <code> tag for inline verbatim text
in the context of bootstrap css styles.



--html_pre_style=...

off, inherit, or transparent: let code blocks inside <pre> tags have
foreground and background color inherited from the surroundnings.
Default: on (use the css-specified typesetting of <pre> tags).
This option is most relevant for Bootstrap styles to
avoid white background in code blocks inside colorful admons.



--html_toc_depth=...

No of levels in the table of contents in HTML output. Default: 2 (includes subsections but not subsubsecti


--html_toc_indent=...

No of spaces for indentation of subsections in the table of
contents in HTML output. Default: 3 (0 gives toc as nested list
in Bootstrap-based styles).


--html_body_font=...

Specify HTML font for text body. =? lists available fonts.


--html_heading_font=...

Specify HTML font for headings. =? lists available fonts.


--html_video_autoplay=...

True for autoplay when HTML is loaded, otherwise False (default).


--html_admon=...

Type of admonition and color:
colors, gray, yellow, apricot, lyx, paragraph.
For html_style=bootstrap*,bootswatch*,
the two legal values are boostrap_panel, bootstrap_alert.


--html_admon_shadow

Add a shadow effect to HTML admon boxes (gray, yellow, apricot).


--html_admon_bg_color=...
```

```
Background color of admon in HTML.


--html_admon_bd_color=...

Boundary color of admon in HTML.


--css=...

Specify a .css style file for HTML output.
If the file does not exist, the default or specified style
(--html_style=) is written to it.


--html_box_shadow

Add a shadow effect in HTML box environments.


--html_exercise_icon=...

Specify a question icon (as a filename in the bundled/html_images
directory in the doconce repo) for being inserted to the right in exercises.
default: turn on predefined question icons according to the chosen style.
none: no icons (this is the default value).


--html_exercise_icon_width=...

Width of the icon image in pixels (must be used with --html_exercise_icon).


--html_raw_github_url=...

URLs to files hosted on the doconce github account.
Internet Explorer (and perhaps other browsers) will not show raw.github.com
files. Instead on should use rawgit.com. For development of HTML sites
in Safari and Chrome and can use rawgit.com.

Values of --html_raw_github_url=:
safe or cdn.rawgit: use this for ready-made sites with potentially some traffic.
The URL becomes https://cdn.rawgit.com/hplgit/doconce/...

test or rawgit: use this for test purposes and development with low traffic.
The URL becomes https://rawgit.com/hplgit/doconce/...

github or raw.github: the URL becomes https://raw.github.com and may fail to
load properly.

githubusercontent or raw.githubusercontent: The URL becomes
https://raw.githubusercontent.com and may fail to load properly.



--html_DOCTYPE

Insert <!DOCTYPE HTML> in the top of the HTML file.
This is required for Internet Explorer and Mozilla.
However, some of the CSS files used by DocOnce may not load properly if
```

```
they are not well formed. That is why no doctype is default in the
generated HTML files.


--html_links_in_new_window

Open HTML links in a new window/tab.


--html_quiz_button_text=...

Text on buttons for collapsing/expanding answers and explanations
in quizzes (with bootstrap styles).
Default: Empty (just pencil glyphion).


--html_bootstrap_navbar=...

Turns the Bootstrap navigation bar on/off. Default: on.


--html_bootstrap_jumbotron=...

Turns the Bootstrap jumbotron intro on/off and governs the
size of the document title. Default: on. Other values: h2, off
(h2 gives h2 heading instead of h1, off gives no jumbotron).


--html_figure_hrule=...

Set horizontal rule(s) above and/or below a figure.
none, off: no rules
top: rule at top (default)
bottom: rule at bottom
top+bottom: rule at top and bottom


--device=...

Set device to paper, screen, or other (paper impacts LaTeX output).


--number_all_equations

Switch latex environments such that all equations get a number.


--denumber_all_equations

Switch latex environments such no equations get a number (useful for removing equation labels in slides).


--latex_style=...

LaTeX style package used for the document.
std: standard LaTeX article or book style,
Springer_lncse: Springer's Lecture Notes in Computational Science and
   Engineering (LNCSE) style,
Springer_llncs: Springer's Lecture Notes in Computer Science style,
Springer_T2: Springer's T2 book style,
Springer_collection: Springer's style for chapters in LNCSE proceedings,
```

```
Korma_Script: Korma Script style,
siamltex: SIAM's standard LaTeX style for papers,
siamltexmm: SIAM's extended (blue) multimedia style for papers.


--latex_font=...

LaTeX font choice: helvetica, palatino, std (Computer Modern, default).


--latex_code_style=...

Typesetting of code blocks.
pyg: use pygments (minted), style is set with --minted_latex_style=
lst: use lstlistings
vrb: use Verbatim (default)

Specifications across languages:
pyg-blue1
lst, lst-yellowgray[style=redblue]
vrb[frame=lines,framesep=2.5mm,framerule=0.7pt]

Detailed specification for each language:
default:vrb-red1[frame=lines]@pycod:lst[style=redblue]@pypro:lst-blue1[style=default]@sys:vrb[f

Here, Verbatim[frame=lines] is used for all code environments, except
pycod, pypro and sys, which have their own specifications.
pycod: lst package with redblue style (and white background)
pypro: lst package with default style and blue1 background
style, sys: Verbatim with the specified arguments and white background.

(Note: @ is delimiter for the language specifications, syntax is
envir:package-background[style parameters]@)



--latex_code_leftmargin=...

Sets the left margin in code blocks. Default: 7 (mm).


--latex_code_bg=...

Background color code blocks. Default: white.


--latex_code_lststyles=...

Filename with LaTeX definitions of lst styles.


--latex_bibstyle=...

LaTeX bibliography style. Default: plain.


--section_numbering=...

Turn section numbering on/off. Default: off for all formats except latex and pdflatex (on for t
```

```
--latex_table_format=...

Default: quote. Other values: left, center, footnotesize, tiny.


--latex_title_layout=...

Layout of the title, authors, and date:
std: traditional LaTeX layout,
titlepage: separate page,
doconce_heading (default): authors with "footnotes" for institutions,
beamer: layout for beamer slides.


--latex_title_reference=...

latex code placed in a footnote for the title,
typically used for acknowledging publisher/source of original
version of the document.


--latex_encoding=...

Encoding for \usepackage[encoding]{inputenc}.
Values: utf8 (default) or latin1.


--latex_papersize=...

Geometry of page size: a6, a4, std (default).


--latex_list_of_exercises=...

LaTeX typesetting of list of exercises:
loe: special, separate list of exercises,
toc: exercises included as part of the table of contents,
none (default): no list of exercises.


--latex_movie=...

Specify package for handling movie/video content.
Default: href (hyperlink to movie file).
Other options: media9, movie15, multimedia (Beamer's \movie command).


--latex_movie_controls=...

Specify control panel for movies. Default: on. Other options: off.


--latex_external_movie_viewer

Allow external movie viewer for movie15 package.


--latex_fancy_header

Typesetting of headers on each page:
If article: section name to the left and page number to the right
```

```
on even page numbers, the other way around on odd page numners.
If book: section name to the left and page numner to the right
on even page numbers, chapter name to the right and page number to
the left on odd page numbers.


--latex_section_headings=...

Typesetting of title/section/subsection headings:
std (default): standard LaTeX,
blue: gray blue color,
strongblue: stronger blue color,
gray: white text on gray background, fit to heading width,
gray-wide: white text on gray background, wide as the textwidth.


--latex_colored_table_rows=...

Colors on every two line in tables: no (default), gray, blue.


--latex_line_numbers

Include line numbers for the running text (only active if there
are inline comments.


--latex_todonotes

Use the todonotes package to typeset inline comments.
Gives colored bubbles in the margin for small inline comments and
in the text for larger comments.


--latex_double_spacing

Sets the LaTeX linespacing to 1.5 (only active if there are
inline comments).


--latex_labels_in_margin

Print equation, section and other LaTeX labels in the margin.


--latex_index_in_margin

Place entries in the index also in the margin.


--latex_preamble=...

User-provided LaTeX preamble file, either complete or additions
to the doconce-generated preamble.


--latex_no_program_footnotelink

If --device=paper, this option removes footnotes with links to
computer programs.
```

```
--latex_admon=...

Type of admonition in LaTeX:
colors1:
(inspired by the NumPy User Guide) applies different colors
for the different admons with an embedded icon,

colors2:
like ‘colors1‘ but the text is wrapped around the icon,

mdfbox:
rounded boxes with a optional title and no icon (default),

graybox2:
box with square corners, gray background, and narrower
than mdfbox, if code it reduces to something like mdfbox
(mdframed based); the summary admon is in case of A4 format
only half of the text width with text wrapped around
(effective for proposals and articles),

grayicon:
box with gray icons and a default light gray background,

yellowicon:
box yellow icons and a default light yellow background,

paragraph:  plain paragraph with boldface heading.

Note: the colors in mdfbox and other boxes can customized.



--latex_admon_color=...

The color to be used as background in admonitions.
A single value applies to all admons:
Either rgb tuple or saturated color a la yellow!5:
  --latex_admon_color=0.1,0.1,0.4
  '--latex_admon_color=yellow!5'
Note the quotes, needed for bash, in the latter example.

Multiple values can be assigned, one for each admon (all admons must
be specified):
  '--latex_admon_color=warning:darkgreen!40!white;notice:darkgray!20!white;summary:tucorange!20!white;ques

If --latex_admon=mdfbox, the specification above with color1!X!color2
will automatically trigger 2*X as the background color of the frametitle.

There are predefined multiple values, e.g.,
  --latex_admon_color=colors1
gives red warnings, blue notice, orange questions, green summaries and
yellow blocks, automatically adjusted with darker frametitles for

If --latex_admon=mdfbox, the background of the title and
the color of the border of box can also be customized by
direct editing. For example, a dark blue border and light
blue title background is obtained by editing the .tex file as

doconce replace 'linecolor=black,' 'linecolor=darkblue,' mydoc.tex
doconce subst 'frametitlebackgroundcolor=.*?,' 'frametitlebackgroundcolor=blue!5,' mydoc.tex
```

```
Actually, this particular (and common) edit is automatically done by the option
  --latex_admon_color=bluestyle
  --latex_admon_color=yellowstyle
(the latter has color yellow!5 instead and yellow!20 for the border)



--latex_admon_title_no_period

By default, a period is added to title admons that do not have a period, question mark, or simi


--latex_admon_envir_map=...

Mapping of code envirs to new envir names inside admons, e.g.,
to get a different code typesetting inside admons. This is useful
if admons have a special color and the color background of code
blocks does not fit will with the color background inside admons.
Then it is natural to use a different verbatim code style inside
admons.
If specifying a number, say 2, as in --latex_admon_envir_map=2,
an envir like pycod gets the number appended: pycod2. One can
then in --latex_code_style= or in doconce ptex2tex or ptex2tex
specify the typesetting of pycod2 environments.
Otherwise the specification must be a mapping for each envir
that should be changed inside the admons:
--latex_admon_envir_map=pycod-pycod_yellow,fpro-fpro2
(from-to,from-to,... syntax).


--latex_subex_header_postfix=...

Default: ).
Gives headers a), b), etc. Can be set to period, colon, etc.


--xelatex

Use xelatex instead of latex/pdflatex.


--latex_double_hyphen

Replace single dash - by double dash -- in LaTeX output.
Somewhat intelligent, but may give unwanted edits. Use with great care!


--latex_elsevier_journal=...

Sets the journal name for the --latex_style=elsevier style.
Default: none (no journal name).


--ipynb_version=...

ipynb version 3 (default) or 4.


--ipynb_split_pyshell=...
```

```
Split interactive sessions into multiple cells after each output.
Applies to pyshell and ipy code environments.
on, True, yes: split (default).
off, False, no: do not split.
Note that pyshell-t and ipy-t environments just displays the session,
while default pyshell and ipy removes all output (all output from print
statements will come after the entire session).



--ipynb_cite=...

Typesetting of bibliography.
plain: simple native typesetting (same as pandoc) (default)
latex: ipynb support for latex-style bibliographies (not mature).


--ipynb_admon=...

Typesetting of admonitions (hint, remarks, box, notice, summary,
warning, question, block - quotes are typeset as quotes).
quote: as Markdown quote (default) with gray line on the left.
paragraph: just the content with the title as paragraph heading.
hrule: title with horizontal rule above and below, then text and
horozontal rule.


--ipynb_figure=...

How to typeset figures in ipynb:
md (plain Markdown syntax),
imgtag (<img src="..." width=...> tag, default)
Image (python cell with Image object).


--ipynb_movie=...

How to typeset movies in ipynb:
md (plain Markdown syntax, default)
HTML: python cell with notebook `HTML` object containing the raw HTML code
that is used in the DocOnce HTML format
ipynb: python cell with notebook `HTML` object with simple/standard
ipynb HTML code for showing a YouTube or local video with a <video>
tag.


--verbose

Write out all OS commands run by doconce.


--examples_as_exercises

Treat examples of the form "==== Example: ..."
as in exercise environments.


--solutions_at_end

Place solutions to exercises at the end of the document.
```

```
--without_solutions

Leave out solution environments from exercises.


--without_answers

Leave out answer environments from exercises.


--without_hints

Leave out hints from exercises.


--wordpress

Make HTML output for wordpress.com pages.


--tables2csv

Write each table to a CSV file table_X.csv,
where X is the table number (autonumbered in according to
appearance in the DocOnce source file).


--sections_up

Upgrade all sections: sections to chapters, subsections
to sections, etc.


--sections_down

Downgrade all sections: chapters to sections, sections
to subsections, etc.


--os_prompt=...

Terminal prompt in output from running OS commands (the
@@@OSCMD instruction). None or empty: no prompt, just the command;
nocmd: no command, just the output. Default is "Terminal>".


--code_skip_until=...

@@@CODE import: skip lines in files up to (and incuding) specified line.


--code_prefix=...

Prefix all @@@CODE imports with some path.


--figure_prefix=...

Prefix all figure filenames with, e.g., an URL.
```

```
--movie_prefix=...

Prefix all movie filenames with, e.g., an URL.


--no_mp4_webm_ogg_alternatives

Use just the specified (.mp4, .webm, .ogg) movie file;
do not allow alternatives in HTML5 video tag.
Used if the just the specified movie format should be played.


--handout

Makes slides output suited for printing.


--urlcheck

Check that all URLs referred to in the document are valid.


--labelcheck=...

Check that all ref{X} has a corresponding label{X}. Fake examples will fail this check and so will general
Turn on when useful. Values: off (default), on.


--short_title=...

Short version of the document's title.


--markdown

Allow Markdown (and some Extended Markdown) syntax as input.


--md2do_output=...

Dump to file the DocOnce code arising from converting from
Markdown. Default value is None (no dump).
Any filename can be specified: --md2do_output=myfile.do.txt


--github_md

Turn on github-flavored-markdown dialect of the pandoc translator


--strapdown

Wrap Markdown output in HTML header/footer such that the
output file (renamed as .html) can automatically be rendered as
an HTML via strapdownjs.com technology. Combine with --github_md
for richer output. Styles are set with --bootswatch_theme=cyborg
(for instance).


--bootswatch_theme=...
```

```
Bootswatch theme for use with --strapdown option.


--strict_markdown_output

Ensure strict/basic Markdown as output.


--multimarkdown_output

Allow MultiMarkdown as output.


--quiz_question_prefix=...

Prefix/title before question in quizzes. Default: "Question:".
Can also be set in square brackets for each individual question.
("Q: [] What is 1+1?"
results in no prefix/title before the "What is 1+1?".


--quiz_choice_prefix=...

Prefix/title before choices in quizzes.
Default for HTML: "Choice", resulting in numbered choices
"Choice 1:", "Choice 2:", etc.
A value with colon, period, or question mark (e.g., "Answer:")
leaves out the numbering.
Default for latex/pdflatex: letter or letter+checkbox.
Other values: number, number+checkbox, number+circle, letter+circle,
letter.
The checkbox or circle is always omitted if answers or solutions are
included (i.e., if none of the --without_answers and
--without_solutions is set).
The choice prefix can also be set in square brackets for each
individual choice.
("Cr: [] Two"
results in no prefix/title before the the answer "Two".



--quiz_horizontal_rule=...

on (default): <hr> before and after quiz in HTML. off: no <hr>.


--quiz_explanations=...

on/off
(some output formats do not support explanations with figures,
math and/or code, this option turns all explanations off.


--rst_uio

Univ. of Oslo version of rst files for their Vortex system.


--rst_mathjax
```

```
Use raw HTML with MathJax for LaTeX mathematics in rst files.


--sphinx_keep_splits

Respect user's !split commands. Default: Override user's !split
and insert new !split before all topmost sections. This is what
makes sense in a Sphinx Table of Contents if one wants to split
the document into multiple parts.


--oneline_paragraphs

Combine paragraphs to one line (does not work well).
```

5.2 Demos

The current text is generated from a DocOnce format stored in the file

```
doc/tutorial/tutorial.do.txt
```

The file `make.sh` in the `tutorial` directory of the DocOnce source code contains a demo of how to produce a variety of formats. The source of this tutorial, `tutorial.do.txt` is the starting point. Running `make.sh` and studying the various generated files and comparing them with the original `tutorial.do.txt` file, gives a quick introduction to how DocOnce is used in a real case.

There is another demo in the `docs/manual` directory which translates the more comprehensive documentation, `manual.do.txt`, to various formats. The `make.sh` script runs a set of translations.

6 Installation of DocOnce and its Dependencies

Below, we explain the manual installation of all software that may be needed when working with DocOnce documents. The impatient way to install everything that is needed is to use Anaconda Python and the `conda` program:

```
Terminal> conda install --channel johannr doconce
```

The `conda` package is available for Mac and Linux only.

If you do not want to use Anaconda and are on a Debian-based Linux computer (running, e.g., Ubuntu), you can instead run the Bash script `install_doconce.sh` or the equivalent Python script `install_doconce.py`. These scripts gives a comprehensive installation. Some users will prefer to install just what is needed for them, and this is explained below.

Version control systems are needed!

The coming installation instructions require that the version control systems Subversion, Mercurial, and Git are installed on your computer.

What about Mac and Windows?

DocOnce is primarily tested on GNU/Debian Linux systems, but also to a minor extent on Mac OS X. Experience with Windows is limited. Since most packages are Python-based and can be installed via `pip install` no problems should arise on Mac and Windows. However, some of the image processing tools and spell checking apply Unix-specific software.

You can omit reading the next sections if you rely on `conda` or `apt-get install` commands in the Bash script for installing DocOnce.

6.1 DocOnce

DocOnce itself is pure Python code hosted at https://github.com/hplgit/doconce. Installation can be done via

```
sudo pip install -e git+https://github.com/hplgit/doconce#egg=doconce
# or if doconce is already installed
sudo pip install -e git+https://github.com/hplgit/doconce#egg=doconce --upgrade
```

However, the recommended approach is to have a copy of the source on the local computer and run `setup.py`:

```
git clone git@github.com:hplgit/doconce.git
cd doconce
sudo python setup.py install
cd ..
```

Since DocOnce is frequently updated, it becomes necessary to ensure that you work with the most recent version:

```
cd doconce
git pull origin master
sudo python setup.py install
```

6.2 Dependencies

Producing HTML documents, plain text, pandoc-extended Markdown, and wikis can be done without installing any other software. However, if you want other formats as output (LaTeX, Sphinx, reStructuredText) and assisting utilities such as preprocesors, spellcheck, file differences, bibliographies, and so on, a lot of extra software must be installed.

Python v2.7. First you need Python version 2.7 and the `pip` installation program. Unless you already have these, we recommend to install a comprehensive Python bundle like Anaconda.

You do not need more software if you avoid using preprocessors, there is no bibliography, and you stick to the output formats LaTeX and HTML (you need of course LaTeX installed to process `.tex` files).

Preprocessors. If you make use of the Preprocess preprocessor, this program must be installed:

```
pip install -e svn+http://preprocess.googlecode.com/svn/trunk#egg=preprocess
```

A much more advanced alternative to Preprocess is Mako. Its installation is done by

```
pip install Mako
```

Note that neither Preprocess nor Mako is run if you do not have preprocessor directives in your DocOnce source. That is, you only need this extra software if you make active use of preprocessors.

Bibliography. The Python package Publish is needed if you use a bibliography in your document (`cite` commands and a `BIBFILE:` specification). The installation is done by

```
pip install -e hg+https://bitbucket.org/logg/publish#egg=publish
```

Image file handling. Different output formats require different formats of image files. For example, PDF or PNG is used for `pdflatex`, PostScript for `latex`, while HTML needs JPEG, GIF, or PNG formats. DocOnce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write

```
sudo apt-get install imagemagick
```

The convenience program `doconce combine_images`, for combining several images into one, will use `montage` and `convert` from ImageMagick and the `pdftk`, `pdfnup`, and `pdfcrop` programs from the `texlive-extra-utils` Debian package. The latter gets installed by

```
sudo apt-get install texlive-extra-utils
```

Automatic image conversion from EPS to PDF calls up `epstopdf`, which can be installed by

```
sudo apt-get install texlive-font-utils
```

Spellcheck. The utility `doconce spellcheck` applies the `ispell` program for spellcheck. On Debian (including Ubuntu) it is installed by

```
sudo apt-get install ispell
```

Ptex2tex for LaTeX Output. Originally, DocOnce relied on the ptex2tex program for very flexible choice of typesetting of verbatim code blocks. A simplified version, `doconce ptex2tex`, is bundled with DocOnce. However, even greater flexibility is now offered by the `--latex_code_style=` option to `doconce format` so unless you already are a `ptex2tex` user, it is recommended to forget about `ptex2tex` and just use the `--latex_code_style=` option.

The stand-alone `ptex2tex` program is installed by

```
svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install
```

It may happen that you need additional style files, you can run a script, `cp2texmf.sh`:

```
cd latex
sh cp2texmf.sh  # copy stylefiles to ~/texmf directory
cd ../..
```

This script copies some special stylefiles that that `ptex2tex` potentially makes use of. Some more standard stylefiles are also needed. These are installed by

```
sudo apt-get install texlive
```

on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the `~/texmf/tex/latex/misc` directory).

Note that the `doconce ptex2tex` command, which needs no installation beyond DocOnce itself, can be used as a simpler alternative to the `ptex2tex` program.

Pygments and the Minted Code Style. The *minted* LaTeX style is popular for typesetting code. This style requires the package Pygments to be installed. On Debian Linux, the simplest approach is to install `sphinx`:

```
pip install sphinx
```

All use of the minted style requires the `-shell-escape` command-line argument when running LaTeX, i.e., `pdflatex -shell-escape`.

Inline comments apply the `todonotes` LaTeX package if the option `--latex_todonotes` is given. The `todonotes` package requires several other packages: `xcolor`, `ifthen`, `xkeyval`, `tikz`, `calc`, `graphicx`, and `setspace`. The relevant Debian packages for installing all this are listed below.

LaTeX packages. Many LaTeX packages are potentially needed, depending on various constructions in the text and command-line options used when compling DocOnce to LaTeX. The standard packages always required are `relsize`, `makeidx`, `setspace`, `color`, `amsmath`, `amsfonts`, `xcolor`, `bm`, `microtype`, `inputenc`, and `hyperref`. Optional packages that might be included in the `.tex` output are `minted`, `listings`, `fancyvrb`, `xunicode`, `inputenc`, `helvet`, `mathpazo`, `wrapfig`, `calc`, `ifthen`, `xkeyval`, `tikz`, `graphicx`, `setspace`, `shadow`, `disable`, `todonotes`, `lineno`, `xr`, `framed`, `mdframe`, `movie15`, `a4paper`, and `a6paper`.

Relevant Debian packages that gives you all of these LaTeX packages are

```
texlive
texlive-extra-utils
texlive-latex-extra
texlive-font-utils
```

Alternatively, one may pull in `texlive-full` to get all available style files.

If you want to use the *anslistings* code environment with `ptex2tex` (`.ptex2tex.cfg` styles `Python_ANS`, `Python_ANSt`, `Cpp_ANS`, etc.) or `doconce ptex2tex` (`envir=ans` or `envir=ans:nt`), you need the `anslistings.sty` file. It can be obtained from the ptex2tex source. The same code style is in "modern DocOnce" just implemented by the command-line option

```
"--latex_code_style=default:lst[style=yellow2_fb]"
```

Sphinx or reStructuredText Output. Output to `sphinx` or `rst` requires the Sphinx software, installed by

```
pip install sphinx --upgrade
```

DocOnce comes with many Sphinx themes that are not part of the standard Sphinx source distribution:

- cloud and redcloud: `https://bitbucket.org/ecollins/cloud_sptheme`
- bootstrap: `https://github.com/ryan-roemer/sphinx-bootstrap-theme`
- solarized: `https://bitbucket.org/miiton/sphinxjp.themes.solarized`
- impressjs: `https://github.com/shkumagai/sphinxjp.themes.impressjs`
- sagecellserver: `https://github.com/kriskda/sphinx-sagecell`

Appropriate installation commands for these themes are

```
pip install -e hg+https://bitbucket.org/ecollins/cloud_sptheme#egg=cloud_sptheme
pip install -e git+https://github.com/ryan-roemer/sphinx-bootstrap-theme#egg=sphinx-bootstrap-theme
pip install -e hg+https://bitbucket.org/miiton/sphinxjp.themes.solarized#egg=sphinxjp.themes.solarized
pip install -e git+https://github.com/shkumagai/sphinxjp.themes.impressjs#egg=sphinxjp.themes.impressjs
pip install -e git+https://github.com/kriskda/sphinx-sagecell#egg=sphinx-sagecell
```

It can also be handy to have special typesetting of IPython sessions:

```
pip install -e git+https://bitbucket.org/hplbit/pygments-ipython-console#egg=pygments-ipython-c
```

To make OpenOffice or LibreOffice documents from `rst` output, you will need more software, typically the following on a Debian system:

```
sudo apt-get install unovonv libreoffice libreoffice-dmaths
```

Markdown and Pandoc Output. The DocOnce format `pandoc` outputs the document in various Markdown versions: the Pandoc extended Markdown format (which via the `pandoc` program can be translated to a range of other formats), strict Markdown, and GitHub-flavored Markdown. Installation of Pandoc, written in Haskell, is most easily done by

```
sudo apt-get install pandoc
```

on Debian (Ubuntu) systems.

Epydoc Output. When the output format is `epydoc` one needs that program too, installed by

```
svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..
```

Remark. Several of the packages above installed from source code are also available in Debian-based system through the `apt-get install` command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For `svn` directories, go to the directory, run `svn update`, and then `sudo python setup.py install`. For Mercurial (`hg`) directories, go to the directory, run `hg pull; hg update`, and then `sudo python setup.py install`.

Analyzing file differences. The `doconce diff file1 file prog` command for illustrating differences between two files `file1` and `file2` using the program `prog` requires `prog` to be installed. By default, `prog` is `difflib` which comes with Python and is always present if you have DocOnce installed. Another choice, `diff`, should be available on all Unix/Linux systems. Other choices, their URL, and their `sudo apt-get install` command on Debian (Ubuntu) systems appear in the table below.

Program	URL	Debian/Ubuntu install
`pdiff`	a2ps wdiff	`sudo apt-get install a2ps wdiff texlive-latex-extra texli`
`latexdiff`	latexdiff	`sudo apt-get install latexdiff`
`kdiff3`	kdiff3	`sudo apt-get install kdiff3`
`diffuse`	diffuse	`sudo apt-get install diffuse`
`xxdiff`	xxdiff	`sudo apt-get install xxdiff`
`meld`	meld	`sudo apt-get install meld`
`tkdiff.tcl`	tkdiff	not in Debian