tutorial.do.txt

```
TITLE: Doconce: Document Once, Include Anywhere
AUTHOR: Hans Petter Langtangen at Simula Research Laboratory and University of O
slo
DATE: today


 * When writing a note, report, manual, etc., do you find it difficult
   to choose the typesetting format? That is, to choose between plain
   (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML,
   reStructuredText, Sphinx, XML, etc.  Would it be convenient to
   start with some very simple text-like format that easily converts
   to the formats listed above, and then at some later stage
   eventually go with a particular format?

 * Do you need to write documents in varying formats but find it
   difficult to remember all the typesetting details of various
   formats like "LaTeX":"http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCa
rd.v2.0.pdf", "HTML":"http://www.htmlcodetutorial.com/", "reStructuredText":"htt
p://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html", "Sphinx":"http
://sphinx.pocoo.org/contents.html", and "wiki":"http://code.google.com/p/support
/wiki/WikiSyntax"? Would it be convenient
   to generate the typesetting details of a particular format from a
   very simple text-like format with minimal tagging?

 * Do you have the same information scattered around in different
   documents in different typesetting formats? Would it be a good idea
   to write things once, in one format, stored in one place, and
   include it anywhere?

If any of these questions are of interest, you should keep on reading.


======= The Doconce Concept  =======

Doconce is two things:

  o Doconce is a very simple and minimally tagged markup language that
    looks like ordinary ASCII text (much like what you would use in an
    email), but the text can be transformed to numerous other formats,
    including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
    Epytext, and also plain text (where non-obvious formatting/tags are
    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
    latter to RTF and MS Word.
    (An experimental translator to Pandoc is under development, and from
    Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML,
    OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)


  o Doconce is a working strategy for never duplicating information.
    Text is written in a single place and then transformed to
    a number of different destinations of diverse type (software
    source code, manuals, tutorials, books, wikis, memos, emails, etc.).
    The Doconce markup language support this working strategy.
    The slogan is: "Write once, include anywhere".



Here are some Doconce features:
```

```
  * Doconce markup does include tags, so the format is more tagged than
    Markdown and Pandoc, but less than reST, and very much less than
    LaTeX and HTML.
  * Doconce can be converted to plain *untagged* text,
    often desirable for computer programs and email.
  * Doconce has good support for copying in parts of computer code,
    say in examples, directly from the source code files.
  * Doconce has full support for LaTeX math, and integrates very well
    with big LaTeX projects (books).
  * Doconce is almost self-explanatory and is a handy starting point
    for generating documents in more complicated markup languages, such
    as Google wiki, LaTeX, and Sphinx. A primary application of Doconce
    is just to make the initial versions of a Sphinx or wiki document.
  * Contrary to the similar Pandoc translator, Doconce integrates with
    Sphinx and Google wiki. However, if these formats are not of interest,
    Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

  * Large books written in LaTeX, but where many pieces (computer demos,
    projects, examples) can be written in Doconce to appear in other
    contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.

  * Software documentation, primarily Python doc strings, which one wants
    to appear as plain untagged text for viewing in Pydoc, as reStructuredText
    for use with Sphinx, as wiki text when publishing the software at
    web sites, and as LaTeX integrated in, e.g., a thesis.

  * Quick memos, which start as plain text in email, then some small
    amount of Doconce tagging is added, before the memos can appear as
    Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular
markup languages used quite some tagging.  Later, almost untagged
markup languages like Markdown and Pandoc became popular. Doconce is
not a replacement of Pandoc, which is a considerably more
sophisticated project. Moreover, Doconce was developed mainly to
fulfill the needs for a flexible source code base for books with much
mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting
and handling text through regular expressions. The possibility for
tweaking the layout is obviously limited since the text can go to
all sorts of sophisticated markup languages. Moreover, because of
limitations of regular expressions, some formatting of Doconce syntax
may face problems when transformed to HTML, LaTeX, Sphinx, and similar
formats.



======= What Does Doconce Look Like? =======

Doconce text looks like ordinary text, but there are some almost invisible
text constructions that allow you to control the formating. For example,

  * bullet lists arise from lines starting with an asterisk,

  * *emphasized words* are surrounded by asterisks,
```

tutorial.do.txt

```
  * _words in boldface_ are surrounded by underscores,

  * words from computer code are enclosed in back quotes and
    then typeset verbatim (monospace font),

  * section headings are recognied by equality (‘=‘) signs before
    and after the text, and the number of ‘=‘ signs indicates the
    level of the section (7 for main section, 5 for subsection,
    3 for subsubsection),

  * paragraph headings are recognized by a double underscore
    before and after the heading,

  * blocks of computer code can easily be included by placing
    ‘!bc‘ (begin code) and ‘!ec‘ (end code) commands at separate lines
    before and after the code block,

  * blocks of computer code can also be imported from source files,

  * blocks of LaTeX mathematics can easily be included by placing
    ‘!bt‘ (begin TeX) and ‘!et‘ (end TeX) commands at separate lines
    before and after the math block,

  * there is support for both LaTeX and text-like inline mathematics,

  * figures and movies with captions, simple tables,
    URLs with links, index list, labels and references are supported,

  * comments can be inserted throughout the text (‘#‘ at the beginning
    of a line),

  * with a simple preprocessor, Preprocess or Mako, one can include
    other documents (files) and large portions of text can be defined
    in or out of the text,

  * with the Mako preprocessor one can even embed Python
    code and use this to steer generation of Doconce text.

Here is an example of some simple text written in the Doconce format:
!bc
===== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and ‘computer‘ words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
```

```
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |---------------------------------|
  |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     |
  |---------------------------------|

# lines beginning with # are comment lines
!ec
The Doconce text above results in the following little document:

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

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and `computer` words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have numbered items instead of bullets, just use an `o`
(for ordered) instead of the asterisk:

  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,
```

```
  |-------------------------------|
  |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    |
  |-------------------------------|

===== Mathematics and Computer Code =====

Inline mathematics, such as $\nu = \sin(x)$|$v = sin(x)$,
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like $\nu = \sin(x)$|$v = sin(x)$ is
typeset as
!bc
$\nu = \sin(x)$|$v = sin(x)$
!ec
The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside
‘!bt‘ and ‘!et‘ (begin tex / end tex) instructions.
The result looks like this:
!bt
\begin{eqnarray}
{\partial u\over\partial t} &=& \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &=& \nabla\cdot(q(u)\nabla v) + g
\end{eqnarray}
!et
Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other formats (but can still be useful
for those who can read LaTeX syntax).

You can have blocks of computer code, starting and ending with
‘!bc‘ and ‘!ec‘ instructions, respectively. Such blocks look like
!bc cod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., ‘!bc xxx‘
where ‘xxx‘ is an identifier like ‘pycod‘ for code snippet in Python,
‘sys‘ for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file ‘.ptext2tex.cfg‘, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments):
!bc
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
!ec
By default, ‘pro‘ and ‘cod‘ are ‘python‘, ‘sys‘ is ‘console‘,
while ‘xpro‘ and ‘xcod‘ are computer language specific for ‘x‘
```

```
in ‘f‘ (Fortran), ‘c‘ (C), ‘cpp‘ (C++), and ‘py‘ (Python).
# ‘rb‘ (Ruby), ‘pl‘ (Perl), and ‘sh‘ (Unix shell).

# (Any sphinx code-block comment, whether inside verbatim code
# blocks or outside, yields a mapping between bc arguments
# and computer languages. In case of muliple definitions, the
# first one is used.)

One can also copy computer code directly from files, either the
complete file or specified parts.  Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!). A complete file is typeset
with ‘!bc pro‘, while a part of a file is copied into a ‘!bc cod‘
environment. What ‘pro‘ and ‘cod‘ mean is then defined through
a ‘.ptex2tex.cfg‘ file for LaTeX and a ‘sphinx code-blocks‘
comment for Sphinx.

Another document can be included by writing ‘#include "mynote.do.txt"‘
on a line starting with (another) hash sign.  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 the
right writing enviroment for plain text.


===== Macros (Newcommands), Cross-References, Index, and Bibliography =====
label{newcommands}

Doconce supports a type of macros via a LaTeX-style *newcommand*
construction.  The newcommands defined in a file with name
‘newcommand_replace.tex‘ are expanded when Doconce is filtered to
other formats, except for LaTeX (since LaTeX performs the expansion
itself).  Newcommands in files with names ‘newcommands.tex‘ and
‘newcommands_keep.tex‘ are kept unaltered when Doconce text is
filtered to other formats, except for the Sphinx format. Since Sphinx
understands LaTeX math, but not newcommands if the Sphinx output is
HTML, it makes most sense to expand all newcommands.  Normally, a user
will put all newcommands that appear in math blocks surrounded by
‘!bt‘ and ‘!et‘ in ‘newcommands_keep.tex‘ to keep them unchanged, at
least if they contribute to make the raw LaTeX math text easier to
read in the formats that cannot render LaTeX.  Newcommands used
elsewhere throughout the text will usually be placed in
‘newcommands_replace.tex‘ and expanded by Doconce.  The definitions of
newcommands in the ‘newcommands*.tex‘ files *must* appear on a single
line (multi-line newcommands are too hard to parse with regular
expressions).

Recent versions of Doconce also offer cross referencing, typically one
can define labels below (sub)sections, in figure captions, or in
equations, and then refer to these later. Entries in an index can be
defined and result in an index at the end for the LaTeX and Sphinx
formats. Citations to literature, with an accompanying bibliography in
a file, are also supported. The syntax of labels, references,
citations, and the bibliography closely resembles that of LaTeX,
making it easy for Doconce documents to be integrated in LaTeX
projects (manuals, books). For further details on functionality and
syntax we refer to the ‘doc/manual/manual.do.txt‘ file (see the
"demo page": "https://doconce.googlecode.com/hg/doc/demos/manual/index.html"
for various formats of this document).
```

tutorial.do.txt

```
# Example on including another Doconce file (using preprocess):

# #include "_doconce2anything.do.txt"


===== Demos =====

The current text is generated from a Doconce format stored in the file
!bc
docs/tutorial/tutorial.do.txt
!ec
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.
"Here": "https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html"
is a sample of how this tutorial looks in different formats.

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.

===== Dependencies =====

If you make use of preprocessor directives in the Doconce source,
either "Preprocess": "http://code.google.com/p/preprocess" or "Mako":
"http://www.makotemplates.org" must be installed.  To make LaTeX
documents (without going through the reStructuredText format) you also
need "ptex2tex": "http://code.google.com/p/ptex2tex" and some style
files that ‘ptex2tex‘ potentially makes use of.  Going from
reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX
requires "docutils": "http://docutils.sourceforge.net".  Making Sphinx
documents requires of course "Sphinx": "http://sphinx.pocoo.org".
All of the mentioned potential dependencies are pure Python packages
which are easily installed.
If translation to "Pandoc": "http://johnmacfarlane.net/pandoc/" is desired,
the Pandoc Haskell program must of course be installed.
```

Doconce: Document Once, Include Anywhere

Hans Petter Langtangen[1,2]

[1]Simula Research Laboratory
[2]University of Oslo

Oct 29, 2011

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

1 The Doconce Concept

Doconce is two things:

1. Doconce is a very simple and minimally tagged markup language that looks like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word. (An experimental translator to Pandoc is under development, and from Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML, OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)

2. Doconce is a working strategy for never duplicating information. Text is written in a single place and then transformed to a number of different destinations of diverse type (software source code, manuals, tutorials, books, wikis, memos, emails, etc.). The Doconce markup language support this working strategy. The slogan is: ”Write once, include anywhere”.

Here are some Doconce features:

- Doconce markup does include tags, so the format is more tagged than Markdown and Pandoc, but less than reST, and very much less than LaTeX and HTML.
- Doconce can be converted to plain *untagged* text, often desirable for computer programs and email.
- Doconce has good support for copying in parts of computer code, say in examples, directly from the source code files.
- Doconce has full support for LaTeX math, and integrates very well with big LaTeX projects (books).
- Doconce is almost self-explanatory and is a handy starting point for generating documents in more complicated markup languages, such as Google wiki, LaTeX, and Sphinx. A primary application of Doconce is just to make the initial versions of a Sphinx or wiki document.
- Contrary to the similar Pandoc translator, Doconce integrates with Sphinx and Google wiki. However, if these formats are not of interest, Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at web sites, and as LaTeX integrated in, e.g., a thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular markup languages used quite some tagging. Later, almost untagged markup languages like Markdown and Pandoc became popular. Doconce is not a replacement of Pandoc, which is a considerably more sophisticated project. Moreover,

Doconce was developed mainly to fulfill the needs for a flexible source code base for books with much mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting and handling text through regular expressions. The possibility for tweaking the layout is obviously limited since the text can go to all sorts of sophisticated markup languages. Moreover, because of limitations of regular expressions, some formatting of Doconce syntax may face problems when transformed to HTML, LaTeX, Sphinx, and similar formats.

2 What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim (monospace font),
- section headings are recognied by equality (=) signs before and after the text, and the number of = signs indicates the level of the section (7 for main section, 5 for subsection, 3 for subsubsection),
- paragraph headings are recognized by a double underscore before and after the heading,
- blocks of computer code can easily be included by placing bc! (begin code) and ec! (end code) commands at separate lines before and after the code block,
- blocks of computer code can also be imported from source files,
- blocks of LaTeX mathematics can easily be included by placing bt! (begin TeX) and et! (end TeX) commands at separate lines before and after the math block,
- there is support for both LaTeX and text-like inline mathematics,
- figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported,
- comments can be inserted throughout the text (# at the beginning of a line),

- with a simple preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text,
- with the Mako preprocessor one can even embed Python code and use this to steer generation of Doconce text.

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

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

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and `computer` words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |---------------------------------|
  |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     |
  |---------------------------------|

# lines beginning with # are comment lines
```

The Doconce text above results in the following little document:

2.1 A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2

- item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

1. item 1
2. item 2
3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in `tutorial.do.txt`.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to Section 2.1.

Doconce also allows inline comments such as (**hpl**: *here I will make some remarks to the text*) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see Section 3 for an example).

Tables are also supperted, e.g.,

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

2.2 Mathematics and Computer Code

Inline mathematics, such as $\nu = \sin(x)$, allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like $\nu = \sin(x)$ is typeset as

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

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside bt! and et! (begin tex / end tex) instructions. The result looks like this:

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

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

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other formats (but can still be useful for those who can read LaTeX syntax).

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

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., bc xxx! where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python).

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!). A complete file is typeset with bc pro!, while a part of a file is copied into a bc cod! environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

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

Doconce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands defined in a file with name `newcommand_replace.tex` are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names `newcommands.tex` and `newcommands_keep.tex` are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by bt! and et! in `newcommands_keep.tex` to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in `newcommands_replace.tex` and expanded by Doconce. The definitions of newcommands in the `newcommands*.tex`

files *must* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the `doc/manual/manual.do.txt` file (see the demo page for various formats of this document).

3 From Doconce to Other Formats

Transformation of a Doconce document `mydoc.do.txt` to various other formats applies the script `doconce format`:

Terminal

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

or just

Terminal

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

The `mako` or `preprocess` programs are always used to preprocess the file first, and options to `mako` or `preprocess` can be added after the filename. For example,

Terminal

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5     # preprocess
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. That is, in the last example, `FORMAT` is defined as `latex`. Inside the Doconce document one can then perform format specific actions through tests like `#if FORMAT == "latex"`.

Inline comments in the text are removed from the output by

Terminal

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

One can also remove such comments from the original Doconce file by running source code:

```
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.

3.1 HTML

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

Terminal

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

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

3.2 LaTeX

Making a LaTeX file `mydoc.tex` from `mydoc.do.txt` is done in two steps:

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

Terminal

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

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 2.3). If these files are present, they are included in the LaTeX document so that your commands are defined.

Step 2. Run `ptex2tex` (if you have it) to make a standard LaTeX file,

Terminal

```
Terminal> ptex2tex mydoc
```

or just perform a plain copy,

Terminal

```
Terminal> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run

Terminal

```
Terminal> ptex2tex -DHELVETICA mydoc
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through

Terminal

```
Terminal> ptex2tex -DLATEX_HEADING=traditional mydoc
```

A separate titlepage can be generate by

Terminal

```
Terminal> ptex2tex -DLATEX_HEADING=titlepage mydoc
```

The `ptex2tex` tool makes it possible to easily switch between many different fancy formattings of computer or verbatim 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 cod! for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

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

Terminal

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

If one wishes to use the `Minted_Python`, `Minted_Cpp`, etc., environments in `ptex2tex` for typesetting code, the `minted` LaTeX package is needed. This package is included by running `doconce format` with the `-DMINTED` option:

Terminal

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

In this case, `latex` must be run with the `-shell-escape` option:

Terminal

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

The `-shell-escape` option is required because the `minted.sty` style file runs the `pygments` program to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

3.3 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

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

3.4 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

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

We may now produce various other formats:

Terminal

```
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. That is, one can easily go from Doconce to Microsoft Word.

3.5 Sphinx

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

Terminal

```
Terminal> doconce sphinx_dir author="authors' names" \
          title="some title" version=1.0 dirname=sphinxdir \
          theme=mytheme file1 file2 file3 ...
```

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 files `file1`, `file2`, etc. that together represent the whole document. Note that none of the individual Doconce files `file1`, `file2`, etc. should include the rest as their union makes up the whole document. 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'`).

With a single-file document in `mydoc.do.txt` one often just runs

Terminal

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory `sphinx-rootdir` is made with relevant files.

The `doconce sphinx_dir` command generates a script `automake-sphinx.sh` for compiling the Sphinx document into an HTML document. This script copies directories named `figs` or `figures` over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, `automake-sphinx.sh` must be edited accordingly. One can either run `automake-sphinx.sh` or perform the steps in the script manually.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the `conf.py` configuration file for Sphinx is edited accordingly, and a script `make-themes.sh` can make HTML documents with one or more themes. For example, to realize the themes `fenics` and `pyramid`, one writes

Terminal

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

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

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

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

Terminal

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

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

Terminal

```
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
```

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

Terminal

```
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`.

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:

Terminal

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

Step 6. View the result:

Terminal

```
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.

3.6 Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by

Terminal

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

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

3.7 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.

3.8 Demos

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

```
docs/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. Here is a sample of how this tutorial looks in different formats.

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.

3.9 Dependencies

If you make use of preprocessor directives in the Doconce source, either Preprocess or Mako must be installed. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that `ptex2tex` potentially makes use of. Going from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course Sphinx. All of the mentioned potential dependencies are pure Python packages which are easily installed. If translation to Pandoc is desired, the Pandoc Haskell program must of course be installed.

Doconce: Document Once, Include Anywhere

Author: Hans Petter Langtangen

Date: Oct 29, 2011

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

The Doconce Concept

Doconce is two things:

1. Doconce is a very simple and minimally tagged markup language that looks like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word. (An experimental translator to Pandoc is under development, and from Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML, OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)
2. Doconce is a working strategy for never duplicating information. Text is written in a single place and then transformed to a number of different destinations of diverse type (software source code, manuals, tutorials, books, wikis, memos, emails, etc.). The Doconce markup language support this working strategy. The slogan is: "Write once, include anywhere".

Here are some Doconce features:

- Doconce markup does include tags, so the format is more tagged than Markdown and Pandoc, but less than reST, and very much less than LaTeX and HTML.
- Doconce can be converted to plain *untagged* text, often desirable for computer programs and email.
- Doconce has good support for copying in parts of computer code, say in examples, directly from the source code files.
- Doconce has full support for LaTeX math, and integrates very well with big LaTeX projects (books).
- Doconce is almost self-explanatory and is a handy starting point for generating documents in more complicated markup languages, such as Google wiki, LaTeX, and Sphinx. A primary application of Doconce is just to make the initial versions of a Sphinx or wiki document.
- Contrary to the similar Pandoc translator, Doconce integrates with Sphinx and Google wiki. However, if these formats are not of interest, Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at web sites, and as LaTeX integrated in, e.g., a thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular markup languages used quite some tagging. Later, almost untagged markup languages like Markdown and Pandoc became popular. Doconce is not a replacement of Pandoc, which is a considerably more sophisticated project. Moreover, Doconce was developed mainly to fulfill the needs for a flexible source code base for books with much mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting and handling text through regular expressions. The possibility for tweaking the layout is obviously limited since the text can go to all sorts of sophisticated markup languages. Moreover, because of limitations of regular expressions, some formatting of Doconce syntax may face problems when transformed to HTML, LaTeX, Sphinx, and similar formats.

What Does Doconce Look Like?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim (monospace font),
- section headings are recognied by equality (=) signs before and after the text, and the number of = signs indicates the level of the section (7 for main section, 5 for subsection, 3 for subsubsection),
- paragraph headings are recognized by a double underscore before and after the heading,
- blocks of computer code can easily be included by placing `!bc` (begin code) and `!ec` (end code) commands at separate lines before and after the code block,
- blocks of computer code can also be imported from source files,
- blocks of LaTeX mathematics can easily be included by placing `!bt` (begin TeX) and `!et` (end TeX) commands at separate lines before and after the math block,
- there is support for both LaTeX and text-like inline mathematics,
- figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported,
- comments can be inserted throughout the text (# at the beginning of a line),
- with a simple preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text,
- with the Mako preprocessor one can even embed Python code and use this to steer generation of Doconce text.

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

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

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and ‘computer‘ words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3
```

```
URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |---------------------------------|
  |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     |
  |---------------------------------|

# lines beginning with # are comment lines
```

The Doconce text above results in the following little document:

A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an `o` (for ordered) instead of the asterisk:

1. item 1
2. item 2
3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a “label” command right after the section title), as in the reference to the section A Subsection with Sample Text.

Doconce also allows inline comments such as (**hpl**: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from

the output by a command-line argument (see the section From Doconce to Other Formats for an example).

Tables are also supperted, e.g.,

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

Mathematics and Computer Code

Inline mathematics, such as v = sin(x), allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like v = sin(x) is typeset as:

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

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside `!bt` and `!et` (begin tex / end tex) instructions. The result looks like this:

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

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other formats (but can still be useful for those who can read LaTeX syntax).

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

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., `!bc xxx` where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python). .. `rb` (Ruby), `pl` (Perl), and `sh` (Unix shell).

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!). A complete file is typeset with `!bc pro`, while a part of a file is copied into a `!bc cod` environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

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

Doconce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands defined in a file with name `newcommand_replace.tex` are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names `newcommands.tex` and `newcommands_keep.tex` are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by `!bt` and `!et` in `newcommands_keep.tex` to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in `newcommands_replace.tex` and expanded by Doconce. The definitions of newcommands in the `newcommands*.tex` files *must* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the `doc/manual/manual.do.txt` file (see the demo page for various formats of this document).

From Doconce to Other Formats

Transformation of a Doconce document `mydoc.do.txt` to various other formats applies the script `doconce format`:

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

or just:

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

The `mako` or `preprocess` programs are always used to preprocess the file first, and options to `mako` or `preprocess` can be added after the filename. For example:

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5     # preprc
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. That is, in the last example, `FORMAT` is defined as `latex`. Inside the Doconce document one can then perform format specific actions through tests like `#if FORMAT == "latex"`.

Inline comments in the text are removed from the output by:

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

One can also remove such comments from the original Doconce file by running source code:

```
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.

HTML

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.

LaTeX

Making a LaTeX file `mydoc.tex` from `mydoc.do.txt` is done in two steps: .. Note: putting code blocks inside a list is not successful in many

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

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

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

Step 2. Run `ptex2tex` (if you have it) to make a standard LaTeX file:

```
Terminal> ptex2tex mydoc
```

or just perform a plain copy:

```
Terminal> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run:

```
Terminal> ptex2tex -DHELVETICA mydoc
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through:

```
Terminal> ptex2tex -DLATEX_HEADING=traditional mydoc
```

A separate titlepage can be generate by:

```
Terminal> ptex2tex -DLATEX_HEADING=titlepage mydoc
```

The `ptex2tex` tool makes it possible to easily switch between many different fancy formattings of computer or verbatim 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 cod` for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

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

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

If one wishes to use the `Minted_Python`, `Minted_Cpp`, etc., environments in `ptex2tex` for typesetting code, the `minted` LaTeX package is needed. This package is included by running `doconce format` with the `-DMINTED` option:

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

In this case, `latex` must be run with the `-shell-escape` option:

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

The `-shell-escape` option is required because the `minted.sty` style file runs the `pygments` program to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

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
```

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. That is, one can easily go from Doconce to Microsoft Word.

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 file1 file2 file3 ...
```

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 files `file1`, `file2`, etc. that together represent the whole document. Note that none of the individual Doconce files `file1`, `file2`, etc. should include the rest as their union makes up the whole document. 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'`).

With a single-file document in `mydoc.do.txt` one often just runs:

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory `sphinx-rootdir` is made with relevant files.

The `doconce sphinx_dir` command generates a script `automake-sphinx.sh` for compiling the Sphinx document into an HTML document. This script copies directories named `figs` or `figures` over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, `automake-sphinx.sh` must be edited accordingly. One can either run `automake-sphinx.sh` or perform the steps in the script manually.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the `conf.py` configuration file for Sphinx is edited accordingly, and a script `make-themes.sh` can make HTML documents with one or more themes. For example, to realize the themes `fenics` and `pyramid`, one writes:

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

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

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

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

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

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
```

Step 3. Copy the `tutorial.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`.

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
```

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.

Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by:

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

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

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.

Demos

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

```
docs/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. Here is a sample of how this tutorial looks in different formats.

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.

Dependencies

system-message

WARNING/2 in `tutorial.rst`, line 385
Duplicate explicit target name: “sphinx”.

If you make use of preprocessor directives in the Doconce source, either Preprocess or Mako must be installed. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that `ptex2tex` potentially makes use of. Going from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course Sphinx. All of the mentioned potential dependencies are pure Python packages which are easily installed. If translation to Pandoc is desired, the Pandoc Haskell program must of course be installed.

Doconce: Document Once, Include Anywhere Documentation

Release 1.0

Hans Petter Langtangen

October 29, 2011

CONTENTS

Contents:

CHAPTER

ONE

DOCONCE: DOCUMENT ONCE, INCLUDE ANYWHERE

Author Hans Petter Langtangen

Date Oct 29, 2011

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

CHAPTER

TWO

THE DOCONCE CONCEPT

Doconce is two things:

1. Doconce is a very simple and minimally tagged markup language that looks like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word. (An experimental translator to Pandoc is under development, and from Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML, OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)
2. Doconce is a working strategy for never duplicating information. Text is written in a single place and then transformed to a number of different destinations of diverse type (software source code, manuals, tutorials, books, wikis, memos, emails, etc.). The Doconce markup language support this working strategy. The slogan is: "Write once, include anywhere".

Here are some Doconce features:

- Doconce markup does include tags, so the format is more tagged than Markdown and Pandoc, but less than reST, and very much less than LaTeX and HTML.
- Doconce can be converted to plain *untagged* text, often desirable for computer programs and email.
- Doconce has good support for copying in parts of computer code, say in examples, directly from the source code files.
- Doconce has full support for LaTeX math, and integrates very well with big LaTeX projects (books).
- Doconce is almost self-explanatory and is a handy starting point for generating documents in more complicated markup languages, such as Google wiki, LaTeX, and Sphinx. A primary application of Doconce is just to make the initial versions of a Sphinx or wiki document.
- Contrary to the similar Pandoc translator, Doconce integrates with Sphinx and Google wiki. However, if these formats are not of interest, Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at web sites, and as LaTeX integrated in, e.g., a thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular markup languages used quite some tagging. Later, almost untagged markup languages like Markdown and Pandoc became popular. Doconce is not a replacement of Pandoc, which is a considerably more sophisticated project. Moreover, Doconce was developed mainly to fulfill the needs for a flexible source code base for books with much mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting and handling text through regular expressions. The possibility for tweaking the layout is obviously limited since the text can go to all sorts of sophisticated markup languages. Moreover, because of limitations of regular expressions, some formatting of Doconce syntax may face problems when transformed to HTML, LaTeX, Sphinx, and similar formats.

CHAPTER

THREE

WHAT DOES DOCONCE LOOK LIKE?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim (monospace font),
- section headings are recognied by equality (=) signs before and after the text, and the number of = signs indicates the level of the section (7 for main section, 5 for subsection, 3 for subsubsection),
- paragraph headings are recognized by a double underscore before and after the heading,
- blocks of computer code can easily be included by placing `!bc` (begin code) and `!ec` (end code) commands at separate lines before and after the code block,
- blocks of computer code can also be imported from source files,
- blocks of LaTeX mathematics can easily be included by placing `!bt` (begin TeX) and `!et` (end TeX) commands at separate lines before and after the math block,
- there is support for both LaTeX and text-like inline mathematics,
- figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported,
- comments can be inserted throughout the text (`#` at the beginning of a line),
- with a simple preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text,
- with the Mako preprocessor one can even embed Python code and use this to steer generation of Doconce text.

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

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

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and `computer` words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,
```

```
  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |--------------------------------|
  |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    |
  |--------------------------------|

# lines beginning with # are comment lines
```

The Doconce text above results in the following little document:

3.1 A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

1. item 1
2. item 2
3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section *A Subsection with Sample Text*.

Doconce also allows inline comments such as (**hpl**: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section *From Doconce to Other Formats* for an example).

Tables are also supperted, e.g.,

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

3.2 Mathematics and Computer Code

Inline mathematics, such as $\nu = \sin(x)$, allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like $\nu = \sin(x)$ is typeset as

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

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside `!bt` and `!et` (begin tex / end tex) instructions. The result looks like this:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u + f, \\ \frac{\partial v}{\partial t} &= \nabla \cdot (q(u)\nabla v) + g \end{aligned} \tag{3.1}$$

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other formats (but can still be useful for those who can read LaTeX syntax).

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

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., `!bc xxx` where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python). .. `rb` (Ruby), `pl` (Perl), and `sh` (Unix shell).

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!). A complete file is typeset with `!bc pro`, while a part of a file is copied into a `!bc cod` environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

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

Doconce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands defined in a file with name `newcommand_replace.tex` are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names `newcommands.tex` and `newcommands_keep.tex` are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by `!bt` and `!et` in `newcommands_keep.tex` to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in `newcommands_replace.tex` and expanded by Doconce. The definitions of newcommands in the `newcommands*.tex` files *must* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the `doc/manual/manual.do.txt` file (see the demo page for various formats of this document).

CHAPTER

FOUR

FROM DOCONCE TO OTHER FORMATS

Transformation of a Doconce document `mydoc.do.txt` to various other formats applies the script `doconce format`:

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

or just

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

The `mako` or `preprocess` programs are always used to preprocess the file first, and options to `mako` or `preprocess` can be added after the filename. For example,

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5     # preprocess
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. That is, in the last example, `FORMAT` is defined as `latex`. Inside the Doconce document one can then perform format specific actions through tests like `#if FORMAT == "latex"`.

Inline comments in the text are removed from the output by

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

One can also remove such comments from the original Doconce file by running source code:

```
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.

4.1 HTML

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.

4.2 LaTeX

Making a LaTeX file `mydoc.tex` from `mydoc.do.txt` is done in two steps: .. Note: putting code blocks inside a list is not successful in many

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

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

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files `newcommands.tex`, `newcommands_keep.tex`, or `newcommands_replace.tex` (see the section *Macros (Newcommands), Cross-References, Index, and Bibliography*). If these files are present, they are included in the LaTeX document so that your commands are defined.

Step 2. Run `ptex2tex` (if you have it) to make a standard LaTeX file,

```
Terminal> ptex2tex mydoc
```

or just perform a plain copy,

```
Terminal> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run

```
Terminal> ptex2tex -DHELVETICA mydoc
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through

```
Terminal> ptex2tex -DLATEX_HEADING=traditional mydoc
```

A separate titlepage can be generate by

```
Terminal> ptex2tex -DLATEX_HEADING=titlepage mydoc
```

The `ptex2tex` tool makes it possible to easily switch between many different fancy formattings of computer or verbatim 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 cod` for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

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

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

If one wishes to use the `Minted_Python`, `Minted_Cpp`, etc., environments in `ptex2tex` for typesetting code, the `minted` LaTeX package is needed. This package is included by running `doconce format` with the `-DMINTED` option:

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

In this case, `latex` must be run with the `-shell-escape` option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
```

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

The `-shell-escape` option is required because the `minted.sty` style file runs the `pygments` program to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

4.3 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
```

4.4 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. That is, one can easily go from Doconce to Microsoft Word.

4.5 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 file1 file2 file3 ...
```

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 files `file1`, `file2`, etc. that together represent the whole document. Note that none of the individual Doconce files `file1`, `file2`, etc. should include the rest as their union makes up the whole document. 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'`).

With a single-file document in `mydoc.do.txt` one often just runs

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory `sphinx-rootdir` is made with relevant files.

The `doconce sphinx_dir` command generates a script `automake-sphinx.sh` for compiling the Sphinx document into an HTML document. This script copies directories named `figs` or `figures` over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, `automake-sphinx.sh` must be edited accordingly. One can either run `automake-sphinx.sh` or perform the steps in the script manually.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the `conf.py` configuration file for Sphinx is edited accordingly, and a script `make-themes.sh` can make HTML documents with one or more themes. For example, to realize the themes `fenics` and `pyramid`, one writes

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

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

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

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

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

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
```

Step 3. Copy the `tutorial.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`.

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
```

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.

4.6 Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by

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

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

4.7 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.

4.8 Demos

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

```
docs/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. Here is a sample of how this tutorial looks in different formats.

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.

4.9 Dependencies

If you make use of preprocessor directives in the Doconce source, either Preprocess or Mako must be installed. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that `ptex2tex` potentially makes use of. Going from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course Sphinx. All of the mentioned potential dependencies are pure Python packages which are easily installed. If translation to Pandoc is desired, the Pandoc Haskell program must of course be installed.

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

tutorial.txt

```
Doconce: Document Once, Include Anywhere
========================================

Hans Petter Langtangen [1, 2]

[1] Simula Research Laboratory
[2] University of Oslo


Date: Oct 29, 2011

 * When writing a note, report, manual, etc., do you find it difficult
   to choose the typesetting format? That is, to choose between plain
   (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML,
   reStructuredText, Sphinx, XML, etc.  Would it be convenient to
   start with some very simple text-like format that easily converts
   to the formats listed above, and then at some later stage
   eventually go with a particular format?

 * Do you need to write documents in varying formats but find it
   difficult to remember all the typesetting details of various
   formats like LaTeX (http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard
.v2.0.pdf), HTML (http://www.htmlcodetutorial.com/), reStructuredText (http://do
cutils.sourceforge.net/docs/ref/rst/restructuredtext.html), Sphinx (http://sphin
x.pocoo.org/contents.html), and wiki (http://code.google.com/p/support/wiki/Wiki
Syntax)? Would it be convenient
   to generate the typesetting details of a particular format from a
   very simple text-like format with minimal tagging?

 * Do you have the same information scattered around in different
   documents in different typesetting formats? Would it be a good idea
   to write things once, in one format, stored in one place, and
   include it anywhere?

If any of these questions are of interest, you should keep on reading.


The Doconce Concept
===================

Doconce is two things:

 1. Doconce is a very simple and minimally tagged markup language that
    looks like ordinary ASCII text (much like what you would use in an
    email), but the text can be transformed to numerous other formats,
    including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
    Epytext, and also plain text (where non-obvious formatting/tags are
    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
    latter to RTF and MS Word.
    (An experimental translator to Pandoc is under development, and from
    Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML,
    OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)

 2. Doconce is a working strategy for never duplicating information.
    Text is written in a single place and then transformed to
    a number of different destinations of diverse type (software
    source code, manuals, tutorials, books, wikis, memos, emails, etc.).
    The Doconce markup language support this working strategy.
```

tutorial.txt

```
    The slogan is: "Write once, include anywhere".

Here are some Doconce features:

  * Doconce markup does include tags, so the format is more tagged than
    Markdown and Pandoc, but less than reST, and very much less than
    LaTeX and HTML.

  * Doconce can be converted to plain *untagged* text,
    often desirable for computer programs and email.

  * Doconce has good support for copying in parts of computer code,
    say in examples, directly from the source code files.

  * Doconce has full support for LaTeX math, and integrates very well
    with big LaTeX projects (books).

  * Doconce is almost self-explanatory and is a handy starting point
    for generating documents in more complicated markup languages, such
    as Google wiki, LaTeX, and Sphinx. A primary application of Doconce
    is just to make the initial versions of a Sphinx or wiki document.

  * Contrary to the similar Pandoc translator, Doconce integrates with
    Sphinx and Google wiki. However, if these formats are not of interest,
    Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

  * Large books written in LaTeX, but where many pieces (computer demos,
    projects, examples) can be written in Doconce to appear in other
    contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.

  * Software documentation, primarily Python doc strings, which one wants
    to appear as plain untagged text for viewing in Pydoc, as reStructuredText
    for use with Sphinx, as wiki text when publishing the software at
    web sites, and as LaTeX integrated in, e.g., a thesis.

  * Quick memos, which start as plain text in email, then some small
    amount of Doconce tagging is added, before the memos can appear as
    Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular
markup languages used quite some tagging.  Later, almost untagged
markup languages like Markdown and Pandoc became popular. Doconce is
not a replacement of Pandoc, which is a considerably more
sophisticated project. Moreover, Doconce was developed mainly to
fulfill the needs for a flexible source code base for books with much
mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting
and handling text through regular expressions. The possibility for
tweaking the layout is obviously limited since the text can go to
all sorts of sophisticated markup languages. Moreover, because of
limitations of regular expressions, some formatting of Doconce syntax
may face problems when transformed to HTML, LaTeX, Sphinx, and similar
formats.
```

tutorial.txt

```
What Does Doconce Look Like?
============================

Doconce text looks like ordinary text, but there are some almost invisible
text constructions that allow you to control the formating. For example,

  * bullet lists arise from lines starting with an asterisk,

  * *emphasized words* are surrounded by asterisks,

  * _words in boldface_ are surrounded by underscores,

  * words from computer code are enclosed in back quotes and
    then typeset verbatim (monospace font),

  * section headings are recognied by equality (=) signs before
    and after the text, and the number of = signs indicates the
    level of the section (7 for main section, 5 for subsection,
    3 for subsubsection),

  * paragraph headings are recognized by a double underscore
    before and after the heading,

  * blocks of computer code can easily be included by placing
    !bc (begin code) and !ec (end code) commands at separate lines
    before and after the code block,

  * blocks of computer code can also be imported from source files,

  * blocks of LaTeX mathematics can easily be included by placing
    !bt (begin TeX) and !et (end TeX) commands at separate lines
    before and after the math block,

  * there is support for both LaTeX and text-like inline mathematics,

  * figures and movies with captions, simple tables,
    URLs with links, index list, labels and references are supported,

  * comments can be inserted throughout the text (# at the beginning
    of a line),

  * with a simple preprocessor, Preprocess or Mako, one can include
    other documents (files) and large portions of text can be defined
    in or out of the text,

  * with the Mako preprocessor one can even embed Python
    code and use this to steer generation of Doconce text.

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


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

        Ordinary text looks like ordinary text, and the tags used for
        _boldface_ words, *emphasized* words, and ‘computer‘ words look
        natural in plain text.  Lists are typeset as you would do in an email,

          * item 1
```

tutorial.txt

```
          * item 2
          * item 3

        Lists can also have automatically numbered items instead of bullets,

          o item 1
          o item 2
          o item 3

        URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl"
.
        If the word is URL, the URL itself becomes the link name,
        as in "URL":"tutorial.do.txt".

        References to sections may use logical names as labels (e.g., a
        "label" command right after the section title), as in the reference to
        Section ref{my:first:sec}.

        Doconce also allows inline comments such as [hpl: here I will make
        some remarks to the text] for allowing authors to make notes. Inline
        comments can be removed from the output by a command-line argument
        (see Section ref{doconce2formats} for an example).

        Tables are also supperted, e.g.,

          |---------------------------------|
          |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     |
          |---------------------------------|

        # lines beginning with # are comment lines

The Doconce text above results in the following little document:

A Subsection with Sample Text
-----------------------------

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and computer words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1

  * item 2

  * item 3

Lists can also have numbered items instead of bullets, just use an o
(for ordered) instead of the asterisk:

 1. item 1

 2. item 2

 3. item 3
```

tutorial.txt

```
URLs with a link word are possible, as in hpl (http://folk.uio.no/hpl).
If the word is URL, the URL itself becomes the link name,
as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
the section "A Subsection with Sample Text".

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see the section "From Doconce to Other Formats" for an example).

Tables are also supperted, e.g.,

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

Mathematics and Computer Code
-----------------------------

Inline mathematics, such as v = sin(x),
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like v = sin(x) is
typeset as::


        $\nu = \sin(x)$|$v = sin(x)$

The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside::

The result looks like this::

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

Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other formats (but can still be useful
for those who can read LaTeX syntax).

You can have blocks of computer code, starting and ending with::


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

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

It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., !bc xxx
where xxx is an identifier like pycod for code snippet in Python,
sys for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file .ptext2tex.cfg, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments)::


        # sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console

By default, pro and cod are python, sys is console,
while xpro and xcod are computer language specific for x
in f (Fortran), c (C), cpp (C++), and py (Python).


One can also copy computer code directly from files, either the
complete file or specified parts.  Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!). A complete file is typeset
with !bc pro, while a part of a file is copied into a !bc cod
environment. What pro and cod mean is then defined through
a .ptex2tex.cfg file for LaTeX and a sphinx code-blocks
comment for Sphinx.

Another document can be included by writing #include "mynote.do.txt"
on a line starting with (another) hash sign.  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 the
right writing enviroment for plain text.


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

Doconce supports a type of macros via a LaTeX-style *newcommand*
construction.  The newcommands defined in a file with name
newcommand_replace.tex are expanded when Doconce is filtered to
other formats, except for LaTeX (since LaTeX performs the expansion
itself).  Newcommands in files with names newcommands.tex and
newcommands_keep.tex are kept unaltered when Doconce text is
filtered to other formats, except for the Sphinx format. Since Sphinx
understands LaTeX math, but not newcommands if the Sphinx output is
HTML, it makes most sense to expand all newcommands.  Normally, a user
will put all newcommands that appear in math blocks surrounded by::

least if they contribute to make the raw LaTeX math text easier to
read in the formats that cannot render LaTeX.  Newcommands used
elsewhere throughout the text will usually be placed in
newcommands_replace.tex and expanded by Doconce.  The definitions of
newcommands in the newcommands*.tex files *must* appear on a single
line (multi-line newcommands are too hard to parse with regular
expressions).
```

```
Recent versions of Doconce also offer cross referencing, typically one
can define labels below (sub)sections, in figure captions, or in
equations, and then refer to these later. Entries in an index can be
defined and result in an index at the end for the LaTeX and Sphinx
formats. Citations to literature, with an accompanying bibliography in
a file, are also supported. The syntax of labels, references,
citations, and the bibliography closely resembles that of LaTeX,
making it easy for Doconce documents to be integrated in LaTeX
projects (manuals, books). For further details on functionality and
syntax we refer to the doc/manual/manual.do.txt file (see the
demo page (https://doconce.googlecode.com/hg/doc/demos/manual/index.html)
for various formats of this document).



From Doconce to Other Formats
=============================

Transformation of a Doconce document mydoc.do.txt to various other
formats applies the script doconce format::


        Terminal> doconce format format mydoc.do.txt

or just::


        Terminal> doconce format format mydoc

The mako or preprocess programs are always used to preprocess the
file first, and options to mako or preprocess can be added after the
filename. For example::


        Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5     # pre
process
        Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mak
o

The variable FORMAT is always defined as the current format when
running preprocess. That is, in the last example, FORMAT is
defined as latex. Inside the Doconce document one can then perform
format specific actions through tests like #if FORMAT == "latex".

Inline comments in the text are removed from the output by::


        Terminal> doconce format latex mydoc remove_inline_comments

One can also remove such comments from the original Doconce file
by running
source code::


        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.


HTML
----

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.

LaTeX
-----

Making a LaTeX file mydoc.tex from mydoc.do.txt is done in two steps:

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


        Terminal> doconce format latex mydoc

LaTeX-specific commands ("newcommands") in math formulas and similar
can be placed in files newcommands.tex, newcommands_keep.tex, or
newcommands_replace.tex (see the section "Macros (Newcommands), Cross-References
, Index, and Bibliography").
If these files are present, they are included in the LaTeX document
so that your commands are defined.

*Step 2.* Run ptex2tex (if you have it) to make a standard LaTeX file::


        Terminal> ptex2tex mydoc

or just perform a plain copy::


        Terminal> cp mydoc.p.tex mydoc.tex

Doconce generates a .p.tex file with some preprocessor macros
that can be used to steer certain properties of the LaTeX document.
For example, to turn on the Helvetica font instead of the standard
Computer Modern font, run::


        Terminal> ptex2tex -DHELVETICA mydoc

The title, authors, and date are by default typeset in a non-standard
way to enable a nicer treatment of multiple authors having
institutions in common. However, the standard LaTeX "maketitle" heading
is also available through::


        Terminal> ptex2tex -DLATEX_HEADING=traditional mydoc

A separate titlepage can be generate by::
```

```
        Terminal> ptex2tex -DLATEX_HEADING=titlepage mydoc


The ptex2tex tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim 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 cod for a code snippet, where cod is set to
a certain environment in .ptex2tex.cfg (e.g., CodeIntended).
There are over 30 styles to choose from.

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


        Terminal> latex mydoc
        Terminal> latex mydoc
        Terminal> makeindex mydoc   # if index
        Terminal> bibitem mydoc     # if bibliography
        Terminal> latex mydoc
        Terminal> dvipdf mydoc

If one wishes to use the Minted_Python, Minted_Cpp, etc., environments
in ptex2tex for typesetting code, the minted LaTeX package is needed.
This package is included by running doconce format with the
-DMINTED option::


        Terminal> ptex2tex -DMINTED mydoc

In this case, latex must be run with the
-shell-escape option::


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

The -shell-escape option is required because the minted.sty style
file runs the pygments program to format code, and this program
cannot be run from latex without the -shell-escape option.


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
```

tutorial.txt

```
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.
That is, one can easily go from Doconce to Microsoft Word.

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 file1 file2 file3 ...

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 files file1,
file2, etc. that together represent the whole document. Note that
none of the individual Doconce files file1, file2, etc. should
include the rest as their union makes up the whole document.
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').

With a single-file document in mydoc.do.txt one often just runs::


        Terminal> doconce sphinx_dir mydoc

and then an appropriate Sphinx directory sphinx-rootdir is made with
relevant files.

The doconce sphinx_dir command generates a script
automake-sphinx.sh for compiling the Sphinx document into an HTML
document.  This script copies directories named figs or figures
over to the Sphinx directory so that figures are accessible in the
Sphinx compilation.  If figures or movies are located in other
directories, automake-sphinx.sh must be edited accordingly. One
can either run automake-sphinx.sh or perform the steps in the
script manually.
```

tutorial.txt

```
Doconce comes with a collection of HTML themes for Sphinx documents.
These are packed out in the Sphinx directory, the conf.py
configuration file for Sphinx is edited accordingly, and a script
make-themes.sh can make HTML documents with one or more themes.
For example,
to realize the themes fenics and pyramid, one writes::


        Terminal> ./make-themes.sh fenics pyramid

The resulting directories with HTML documents are _build/html_fenics
and _build/html_pyramid, respectively. Without arguments,
make-themes.sh makes all available themes (!).

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

*Step 1.* Translate Doconce into the Sphinx dialect of
the reStructuredText format::


        Terminal> doconce format sphinx mydoc


*Step 2.* Create a Sphinx root directory with a conf.py file,
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


*Step 3.* Copy the tutorial.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.

*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


*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.


Google Code Wiki
----------------

There are several different wiki dialects, but Doconce only support the
one used by Google Code (http://code.google.com/p/support/wiki/WikiSyntax).
The transformation to this format, called gwiki to explicitly mark
it as the Google Code dialect, is done by::


        Terminal> doconce format gwiki mydoc.do.txt

You can then open a new wiki page for your Google Code project, copy
the mydoc.gwiki output file from doconce format and paste the
file contents into the wiki page. Press _Preview_ or _Save Page_ to
see the formatted result.

When the Doconce file contains figures, each figure filename must 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).
```

tutorial.txt

```
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.


Demos
-----

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


        docs/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.
Here (https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html)
is a sample of how this tutorial looks in different formats.

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.

Dependencies
------------

If you make use of preprocessor directives in the Doconce source,
either Preprocess (http://code.google.com/p/preprocess) or Mako (http://www.mako
templates.org) must be installed.  To make LaTeX
documents (without going through the reStructuredText format) you also
need ptex2tex (http://code.google.com/p/ptex2tex) and some style
files that ptex2tex potentially makes use of.  Going from
reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX
requires docutils (http://docutils.sourceforge.net).  Making Sphinx
documents requires of course Sphinx (http://sphinx.pocoo.org).
All of the mentioned potential dependencies are pure Python packages
which are easily installed.
If translation to Pandoc (http://johnmacfarlane.net/pandoc/) is desired,
the Pandoc Haskell program must of course be installed.
```

tutorial.epytext

```
TITLE: Doconce: Document Once, Include Anywhere
BY: Hans Petter Langtangen (Simula Research Laboratory, and University of Oslo)D
ATE: today


 − When writing a note, report, manual, etc., do you find it difficult
   to choose the typesetting format? That is, to choose between plain
   (email−like) text, wiki, Word/OpenOffice, LaTeX, HTML,
   reStructuredText, Sphinx, XML, etc.  Would it be convenient to
   start with some very simple text−like format that easily converts
   to the formats listed above, and then at some later stage
   eventually go with a particular format?
 − Do you need to write documents in varying formats but find it
   difficult to remember all the typesetting details of various
   formats like U{LaTeX<http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCar
d.v2.0.pdf>}, U{HTML<http://www.htmlcodetutorial.com/>}, U{reStructuredText<http
://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html>}, U{Sphinx<http:
//sphinx.pocoo.org/contents.html>}, and U{wiki<http://code.google.com/p/support/
wiki/WikiSyntax>}? Would it be convenient
   to generate the typesetting details of a particular format from a
   very simple text−like format with minimal tagging?
 − Do you have the same information scattered around in different
   documents in different typesetting formats? Would it be a good idea
   to write things once, in one format, stored in one place, and
   include it anywhere?

If any of these questions are of interest, you should keep on reading.


The Doconce Concept
==================

Doconce is two things:

 1. Doconce is a very simple and minimally tagged markup language that
    looks like ordinary ASCII text (much like what you would use in an
    email), but the text can be transformed to numerous other formats,
    including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
    Epytext, and also plain text (where non−obvious formatting/tags are
    removed for clear reading in, e.g., emails). From reStructuredText
    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
    latter to RTF and MS Word.
    (An experimental translator to Pandoc is under development, and from
    Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML,
    OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)
 2. Doconce is a working strategy for never duplicating information.
    Text is written in a single place and then transformed to
    a number of different destinations of diverse type (software
    source code, manuals, tutorials, books, wikis, memos, emails, etc.).
    The Doconce markup language support this working strategy.
    The slogan is: "Write once, include anywhere".

Here are some Doconce features:

  − Doconce markup does include tags, so the format is more tagged than
    Markdown and Pandoc, but less than reST, and very much less than
    LaTeX and HTML.
  − Doconce can be converted to plain I{untagged} text,
    often desirable for computer programs and email.
```

```
  - Doconce has good support for copying in parts of computer code,
    say in examples, directly from the source code files.
  - Doconce has full support for LaTeX math, and integrates very well
    with big LaTeX projects (books).
  - Doconce is almost self-explanatory and is a handy starting point
    for generating documents in more complicated markup languages, such
    as Google wiki, LaTeX, and Sphinx. A primary application of Doconce
    is just to make the initial versions of a Sphinx or wiki document.
  - Contrary to the similar Pandoc translator, Doconce integrates with
    Sphinx and Google wiki. However, if these formats are not of interest,
    Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

  - Large books written in LaTeX, but where many pieces (computer demos,
    projects, examples) can be written in Doconce to appear in other
    contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.
  - Software documentation, primarily Python doc strings, which one wants
    to appear as plain untagged text for viewing in Pydoc, as reStructuredText
    for use with Sphinx, as wiki text when publishing the software at
    web sites, and as LaTeX integrated in, e.g., a thesis.
  - Quick memos, which start as plain text in email, then some small
    amount of Doconce tagging is added, before the memos can appear as
    Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular
markup languages used quite some tagging.  Later, almost untagged
markup languages like Markdown and Pandoc became popular. Doconce is
not a replacement of Pandoc, which is a considerably more
sophisticated project. Moreover, Doconce was developed mainly to
fulfill the needs for a flexible source code base for books with much
mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting
and handling text through regular expressions. The possibility for
tweaking the layout is obviously limited since the text can go to
all sorts of sophisticated markup languages. Moreover, because of
limitations of regular expressions, some formatting of Doconce syntax
may face problems when transformed to HTML, LaTeX, Sphinx, and similar
formats.



What Does Doconce Look Like?
============================

Doconce text looks like ordinary text, but there are some almost invisible
text constructions that allow you to control the formating. For example,

  - bullet lists arise from lines starting with an asterisk,
  - I{emphasized words} are surrounded by asterisks,
  - B{words in boldface} are surrounded by underscores,
  - words from computer code are enclosed in back quotes and
    then typeset verbatim (monospace font),
  - section headings are recognied by equality (C{=}) signs before
    and after the text, and the number of C{=} signs indicates the
    level of the section (7 for main section, 5 for subsection,
    3 for subsubsection),
  - paragraph headings are recognized by a double underscore
```

tutorial.epytext

```
    before and after the heading,
  - blocks of computer code can easily be included by placing
    C{!bc} (begin code) and C{!ec} (end code) commands at separate lines
    before and after the code block,
  - blocks of computer code can also be imported from source files,
  - blocks of LaTeX mathematics can easily be included by placing
    C{!bt} (begin TeX) and C{!et} (end TeX) commands at separate lines
    before and after the math block,
  - there is support for both LaTeX and text-like inline mathematics,
  - figures and movies with captions, simple tables,
    URLs with links, index list, labels and references are supported,
  - comments can be inserted throughout the text (C{#} at the beginning
    of a line),
  - with a simple preprocessor, Preprocess or Mako, one can include
    other documents (files) and large portions of text can be defined
    in or out of the text,
  - with the Mako preprocessor one can even embed Python
    code and use this to steer generation of Doconce text.

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


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

        Ordinary text looks like ordinary text, and the tags used for
        _boldface_ words, *emphasized* words, and ‘computer‘ words look
        natural in plain text.  Lists are typeset as you would do in an email,

          * item 1
          * item 2
          * item 3

        Lists can also have automatically numbered items instead of bullets,

          o item 1
          o item 2
          o item 3

        URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl"
.
        If the word is URL, the URL itself becomes the link name,
        as in "URL":"tutorial.do.txt".

        References to sections may use logical names as labels (e.g., a
        "label" command right after the section title), as in the reference to
        Section ref{my:first:sec}.

        Doconce also allows inline comments such as [hpl: here I will make
        some remarks to the text] for allowing authors to make notes. Inline
        comments can be removed from the output by a command-line argument
        (see Section ref{doconce2formats} for an example).

        Tables are also supperted, e.g.,

          |---------------------------------|
          |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    |
          |------------------------------|

        # lines beginning with # are comment lines
```

The Doconce text above results in the following little document:

A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for
B{boldface} words, I{emphasized} words, and C{computer} words look
natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an C{o}
(for ordered) instead of the asterisk:

1. item 1
2. item 2
3. item 3

URLs with a link word are possible, as in U{hpl<http://folk.uio.no/hpl>}.
If the word is URL, the URL itself becomes the link name,
as in U{tutorial.do.txt<tutorial.do.txt>}.

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
the section "A Subsection with Sample Text".

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see the section "From Doconce to Other Formats" for an example).

Tables are also supperted, e.g.,

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

Mathematics and Computer Code

Inline mathematics, such as M{v = sin(x)},
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like M{v = sin(x)} is
typeset as::

```
            NOTE: A verbatim block has been removed because
                  it causes problems for Epytext.


The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside
C{!bt} and C{!et} (begin tex / end tex) instructions.
The result looks like this::


            NOTE: A verbatim block has been removed because
                  it causes problems for Epytext.


Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other formats (but can still be useful
for those who can read LaTeX syntax).

You can have blocks of computer code, starting and ending with
C{!bc} and C{!ec} instructions, respectively. Such blocks look like::


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

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

It is possible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., C{!bc xxx}
where C{xxx} is an identifier like C{pycod} for code snippet in Python,
C{sys} for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file C{.ptext2tex.cfg}, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments)::


        # sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console

By default, C{pro} and C{cod} are C{python}, C{sys} is C{console},
while C{xpro} and C{xcod} are computer language specific for C{x}
in C{f} (Fortran), C{c} (C), C{cpp} (C++), and C{py} (Python).


One can also copy computer code directly from files, either the
complete file or specified parts.  Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!). A complete file is typeset
with C{!bc pro}, while a part of a file is copied into a C{!bc cod}
environment. What C{pro} and C{cod} mean is then defined through
a C{.ptex2tex.cfg} file for LaTeX and a C{sphinx code-blocks}
comment for Sphinx.
```

```
Another document can be included by writing C{#include "mynote.do.txt"}
on a line starting with (another) hash sign.  Doconce documents have
extension C{do.txt}. The C{do} part stands for doconce, while the
trailing C{.txt} denotes a text document so that editors gives you the
right writing enviroment for plain text.


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

Doconce supports a type of macros via a LaTeX-style I{newcommand}
construction.  The newcommands defined in a file with name
C{newcommand_replace.tex} are expanded when Doconce is filtered to
other formats, except for LaTeX (since LaTeX performs the expansion
itself).  Newcommands in files with names C{newcommands.tex} and
C{newcommands_keep.tex} are kept unaltered when Doconce text is
filtered to other formats, except for the Sphinx format. Since Sphinx
understands LaTeX math, but not newcommands if the Sphinx output is
HTML, it makes most sense to expand all newcommands.  Normally, a user
will put all newcommands that appear in math blocks surrounded by
C{!bt} and C{!et} in C{newcommands_keep.tex} to keep them unchanged, at
least if they contribute to make the raw LaTeX math text easier to
read in the formats that cannot render LaTeX.  Newcommands used
elsewhere throughout the text will usually be placed in
C{newcommands_replace.tex} and expanded by Doconce.  The definitions of
newcommands in the C{newcommands*.tex} files I{must} appear on a single
line (multi-line newcommands are too hard to parse with regular
expressions).

Recent versions of Doconce also offer cross referencing, typically one
can define labels below (sub)sections, in figure captions, or in
equations, and then refer to these later. Entries in an index can be
defined and result in an index at the end for the LaTeX and Sphinx
formats. Citations to literature, with an accompanying bibliography in
a file, are also supported. The syntax of labels, references,
citations, and the bibliography closely resembles that of LaTeX,
making it easy for Doconce documents to be integrated in LaTeX
projects (manuals, books). For further details on functionality and
syntax we refer to the C{doc/manual/manual.do.txt} file (see the
U{demo page<https://doconce.googlecode.com/hg/doc/demos/manual/index.html>}
for various formats of this document).



From Doconce to Other Formats
=============================

Transformation of a Doconce document C{mydoc.do.txt} to various other
formats applies the script C{doconce format}::


        Terminal> doconce format format mydoc.do.txt

or just::


        Terminal> doconce format format mydoc
```

```
The C{mako} or C{preprocess} programs are always used to preprocess the
file first, and options to C{mako} or C{preprocess} can be added after the
filename. For example::


        Terminal> doconce format latex mydoc −Dextra_sections −DVAR1=5     # pre
process
        Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mak
o

The variable C{FORMAT} is always defined as the current format when
running C{preprocess}. That is, in the last example, C{FORMAT} is
defined as C{latex}. Inside the Doconce document one can then perform
format specific actions through tests like C{#if FORMAT == "latex"}.

Inline comments in the text are removed from the output by::


        Terminal> doconce format latex mydoc remove_inline_comments

One can also remove such comments from the original Doconce file
by running
source code::


        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.


HTML
----

Making an HTML version of a Doconce file C{mydoc.do.txt}
is performed by::


        Terminal> doconce format html mydoc

The resulting file C{mydoc.html} can be loaded into any web browser for viewing.

LaTeX
-----

Making a LaTeX file C{mydoc.tex} from C{mydoc.do.txt} is done in two steps:

I{Step 1.} Filter the doconce text to a pre−LaTeX form C{mydoc.p.tex} for
     C{ptex2tex}::


        Terminal> doconce format latex mydoc

LaTeX−specific commands ("newcommands") in math formulas and similar
can be placed in files C{newcommands.tex}, C{newcommands_keep.tex}, or
C{newcommands_replace.tex} (see the section "Macros (Newcommands), Cross−Referen
ces, Index, and Bibliography").
If these files are present, they are included in the LaTeX document
```

```
so that your commands are defined.

I{Step 2.} Run C{ptex2tex} (if you have it) to make a standard LaTeX file::


        Terminal> ptex2tex mydoc

or just perform a plain copy::


        Terminal> cp mydoc.p.tex mydoc.tex

Doconce generates a C{.p.tex} file with some preprocessor macros
that can be used to steer certain properties of the LaTeX document.
For example, to turn on the Helvetica font instead of the standard
Computer Modern font, run::


        Terminal> ptex2tex -DHELVETICA mydoc

The title, authors, and date are by default typeset in a non-standard
way to enable a nicer treatment of multiple authors having
institutions in common. However, the standard LaTeX "maketitle" heading
is also available through::


        Terminal> ptex2tex -DLATEX_HEADING=traditional mydoc

A separate titlepage can be generate by::


        Terminal> ptex2tex -DLATEX_HEADING=titlepage mydoc


The C{ptex2tex} tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim code in LaTeX
documents. After any C{!bc} command in the Doconce source you can
insert verbatim block styles as defined in your C{.ptex2tex.cfg}
file, e.g., C{!bc cod} for a code snippet, where C{cod} is set to
a certain environment in C{.ptex2tex.cfg} (e.g., C{CodeIntended}).
There are over 30 styles to choose from.

I{Step 3.} Compile C{mydoc.tex}
and create the PDF file::


        Terminal> latex mydoc
        Terminal> latex mydoc
        Terminal> makeindex mydoc   # if index
        Terminal> bibitem mydoc     # if bibliography
        Terminal> latex mydoc
        Terminal> dvipdf mydoc

If one wishes to use the C{Minted_Python}, C{Minted_Cpp}, etc., environments
in C{ptex2tex} for typesetting code, the C{minted} LaTeX package is needed.
This package is included by running C{doconce format} with the
C{-DMINTED} option::
```

```
        Terminal> ptex2tex -DMINTED mydoc

In this case, C{latex} must be run with the
C{-shell-escape} option::


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

The C{-shell-escape} option is required because the C{minted.sty} style
file runs the C{pygments} program to format code, and this program
cannot be run from C{latex} without the C{-shell-escape} option.


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


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 C{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 C{mydoc.odt} can be loaded into OpenOffice and
saved in, among other things, the RTF format or the Microsoft Word format.
That is, one can easily go from Doconce to Microsoft Word.

Sphinx
------

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


        Terminal> doconce sphinx_dir author="authors' names" \
```

```
                    title="some title" version=1.0 dirname=sphinxdir \
                    theme=mytheme file1 file2 file3 ...

The keywords C{author}, C{title}, and C{version} are used in the headings
of the Sphinx document. By default, C{version} is 1.0 and the script
will try to deduce authors and title from the doconce files C{file1},
C{file2}, etc. that together represent the whole document. Note that
none of the individual Doconce files C{file1}, C{file2}, etc. should
include the rest as their union makes up the whole document.
The default value of C{dirname} is C{sphinx-rootdir}. The C{theme}
keyword is used to set the theme for design of HTML output from
Sphinx (the default theme is C{'default'}).

With a single-file document in C{mydoc.do.txt} one often just runs::


        Terminal> doconce sphinx_dir mydoc

and then an appropriate Sphinx directory C{sphinx-rootdir} is made with
relevant files.

The C{doconce sphinx_dir} command generates a script
C{automake-sphinx.sh} for compiling the Sphinx document into an HTML
document.  This script copies directories named C{figs} or C{figures}
over to the Sphinx directory so that figures are accessible in the
Sphinx compilation.  If figures or movies are located in other
directories, C{automake-sphinx.sh} must be edited accordingly. One
can either run C{automake-sphinx.sh} or perform the steps in the
script manually.

Doconce comes with a collection of HTML themes for Sphinx documents.
These are packed out in the Sphinx directory, the C{conf.py}
configuration file for Sphinx is edited accordingly, and a script
C{make-themes.sh} can make HTML documents with one or more themes.
For example,
to realize the themes C{fenics} and C{pyramid}, one writes::


        Terminal> ./make-themes.sh fenics pyramid

The resulting directories with HTML documents are C{_build/html_fenics}
and C{_build/html_pyramid}, respectively. Without arguments,
C{make-themes.sh} makes all available themes (!).

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

I{Step 1.} Translate Doconce into the Sphinx dialect of
the reStructuredText format::


        Terminal> doconce format sphinx mydoc


I{Step 2.} Create a Sphinx root directory with a C{conf.py} file,
either manually or by using the interactive C{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


I{Step 3.} Copy the C{tutorial.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 C{mydoc.rst} in the C{sphinx-rootdir}
directory. Either edit C{mydoc.rst} so that figure file paths are correct,
or simply copy your figure directories to C{sphinx-rootdir}.

I{Step 4.} Edit the generated C{index.rst} file so that C{mydoc.rst}
is included, i.e., add C{mydoc} to the C{toctree} section so that it becomes::


        .. toctree::
           :maxdepth: 2

           mydoc

(The spaces before C{mydoc} are important!)

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


        make clean   # remove old versions
        make html


I{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 C{!bc}: C{cod} gives Python
(C{code-block:: python} in Sphinx syntax) and C{cppcod} gives C++, but
all such arguments can be customized both for Sphinx and LaTeX output.


Google Code Wiki
----------------

There are several different wiki dialects, but Doconce only support the
one used by U{Google Code<http://code.google.com/p/support/wiki/WikiSyntax>}.
The transformation to this format, called C{gwiki} to explicitly mark
it as the Google Code dialect, is done by::


        Terminal> doconce format gwiki mydoc.do.txt

You can then open a new wiki page for your Google Code project, copy
the C{mydoc.gwiki} output file from C{doconce format} and paste the
file contents into the wiki page. Press B{Preview} or B{Save Page} to
see the formatted result.

When the Doconce file contains figures, each figure filename must 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).


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
C{.rst} file is going to be filtered to LaTeX or HTML, it cannot know
if C{.eps} or C{.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 C{make.sh} files in C{docs/manual} and C{docs/tutorial}
constitute comprehensive examples on how such scripts can be made.


Demos
-----

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


        docs/tutorial/tutorial.do.txt

The file C{make.sh} in the C{tutorial} directory of the
Doconce source code contains a demo of how to produce a variety of
formats.  The source of this tutorial, C{tutorial.do.txt} is the
starting point.  Running C{make.sh} and studying the various generated
files and comparing them with the original C{tutorial.do.txt} file,
```

tutorial.epytext

```
gives a quick introduction to how Doconce is used in a real case.
U{Here<https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html>}
is a sample of how this tutorial looks in different formats.

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

Dependencies
------------

If you make use of preprocessor directives in the Doconce source,
either U{Preprocess<http://code.google.com/p/preprocess>} or U{Mako<http://www.m
akotemplates.org>} must be installed.  To make LaTeX
documents (without going through the reStructuredText format) you also
need U{ptex2tex<http://code.google.com/p/ptex2tex>} and some style
files that C{ptex2tex} potentially makes use of.  Going from
reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX
requires U{docutils<http://docutils.sourceforge.net>}.  Making Sphinx
documents requires of course U{Sphinx<http://sphinx.pocoo.org>}.
All of the mentioned potential dependencies are pure Python packages
which are easily installed.
If translation to U{Pandoc<http://johnmacfarlane.net/pandoc/>} is desired,
the Pandoc Haskell program must of course be installed.
```

tutorial.gwiki

```
#summary Doconce: Document Once, Include Anywhere
<wiki:toc max_depth="2" />
By *Hans Petter Langtangen*

==== Oct 29, 2011 ====

 * When writing a note, report, manual, etc., do you find it difficult   to choo
se the typesetting format? That is, to choose between plain   (email-like) text,
 wiki, Word/OpenOffice, LaTeX, HTML,   reStructuredText, Sphinx, XML, etc.  Woul
d it be convenient to   start with some very simple text-like format that easily
 converts   to the formats listed above, and then at some later stage   eventual
ly go with a particular format?
 * Do you need to write documents in varying formats but find it   difficult to
remember all the typesetting details of various   formats like [http://refcards.
com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf LaTeX], [http://www.htmlcodet
utorial.com/ HTML], [http://docutils.sourceforge.net/docs/ref/rst/restructuredte
xt.html reStructuredText], [http://sphinx.pocoo.org/contents.html Sphinx], and [
http://code.google.com/p/support/wiki/WikiSyntax wiki]? Would it be convenient
 to generate the typesetting details of a particular format from a   very simple
 text-like format with minimal tagging?
 * Do you have the same information scattered around in different   documents in
 different typesetting formats? Would it be a good idea   to write things once,
in one format, stored in one place, and   include it anywhere?

If any of these questions are of interest, you should keep on reading.



== The Doconce Concept ==

Doconce is two things:


 # Doconce is a very simple and minimally tagged markup language that    looks l
ike ordinary ASCII text (much like what you would use in an    email), but the t
ext can be transformed to numerous other formats,    including HTML, wiki, LaTeX
, PDF, reStructuredText (reST), Sphinx,    Epytext, and also plain text (where n
on-obvious formatting/tags are    removed for clear reading in, e.g., emails). F
rom reStructuredText    you can go to XML, HTML, LaTeX, PDF, OpenOffice, and fro
m the    latter to RTF and MS Word.    (An experimental translator to Pandoc is
under development, and from    Pandoc one can generate Markdown, reST, LaTeX, HT
ML, PDF, DocBook XML,    OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and oth
er formats.)
 # Doconce is a working strategy for never duplicating information.    Text is w
ritten in a single place and then transformed to    a number of different destin
ations of diverse type (software    source code, manuals, tutorials, books, wiki
s, memos, emails, etc.).    The Doconce markup language support this working str
ategy.    The slogan is: "Write once, include anywhere".

Here are some Doconce features:


  * Doconce markup does include tags, so the format is more tagged than    Markd
own and Pandoc, but less than reST, and very much less than    LaTeX and HTML.
  * Doconce can be converted to plain *untagged* text,    often desirable for co
mputer programs and email.
  * Doconce has good support for copying in parts of computer code,    say in ex
amples, directly from the source code files.
  * Doconce has full support for LaTeX math, and integrates very well    with bi
```

g LaTeX projects (books).
  * Doconce is almost self-explanatory and is a handy starting point for generating documents in more complicated markup languages, such as Google wiki, LaTeX, and Sphinx. A primary application of Doconce is just to make the initial versions of a Sphinx or wiki document.
  * Contrary to the similar Pandoc translator, Doconce integrates with Sphinx and Google wiki. However, if these formats are not of interest, Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

  * Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.
  * Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at web sites, and as LaTeX integrated in, e.g., a thesis.
  * Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular
markup languages used quite some tagging. Later, almost untagged
markup languages like Markdown and Pandoc became popular. Doconce is
not a replacement of Pandoc, which is a considerably more
sophisticated project. Moreover, Doconce was developed mainly to
fulfill the needs for a flexible source code base for books with much
mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting
and handling text through regular expressions. The possibility for
tweaking the layout is obviously limited since the text can go to
all sorts of sophisticated markup languages. Moreover, because of
limitations of regular expressions, some formatting of Doconce syntax
may face problems when transformed to HTML, LaTeX, Sphinx, and similar
formats.

== What Does Doconce Look Like? ==

Doconce text looks like ordinary text, but there are some almost invisible
text constructions that allow you to control the formating. For example,

  * bullet lists arise from lines starting with an asterisk,
  * *emphasized words* are surrounded by asterisks,
  * *words in boldface* are surrounded by underscores,
  * words from computer code are enclosed in back quotes and then typeset verbatim (monospace font),
  * section headings are recognied by equality (‘=‘) signs before and after the text, and the number of ‘=‘ signs indicates the level of the section (7 for main section, 5 for subsection, 3 for subsubsection),
  * paragraph headings are recognized by a double underscore before and after the heading,
  * blocks of computer code can easily be included by placing ‘!bc‘ (begin code) and ‘!ec‘ (end code) commands at separate lines before and after the code

```
 block,
  * blocks of computer code can also be imported from source files,
  * blocks of LaTeX mathematics can easily be included by placing    ‘!bt‘ (begi
n TeX) and ‘!et‘ (end TeX) commands at separate lines    before and after the ma
th block,
  * there is support for both LaTeX and text-like inline mathematics,
  * figures and movies with captions, simple tables,    URLs with links, index l
ist, labels and references are supported,
  * comments can be inserted throughout the text (‘#‘ at the beginning    of a l
ine),
  * with a simple preprocessor, Preprocess or Mako, one can include    other doc
uments (files) and large portions of text can be defined    in or out of the tex
t,
  * with the Mako preprocessor one can even embed Python    code and use this to
 steer generation of Doconce text.

Here is an example of some simple text written in the Doconce format:
{{{
===== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and ‘computer‘ words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |--------------------------------|
  |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    |
  |--------------------------------|

# lines beginning with # are comment lines
}}}
```

```
}}}
The Doconce text above results in the following little document:

==== A Subsection with Sample Text ====

Ordinary text looks like ordinary text, and the tags used for
*boldface* words, *emphasized* words, and ‘computer‘ words look
natural in plain text.  Lists are typeset as you would do in an email,


  * item 1
  * item 2
  * item 3

Lists can also have numbered items instead of bullets, just use an ‘o‘
(for ordered) instead of the asterisk:


 # item 1
 # item 2
 # item 3

URLs with a link word are possible, as in [http://folk.uio.no/hpl hpl].
If the word is URL, the URL itself becomes the link name,
as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
the section [#A_Subsection_with_Sample_Text].

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see the section [#From_Doconce_to_Other_Formats] for an example).

Tables are also supperted, e.g.,


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

==== Mathematics and Computer Code ====

Inline mathematics, such as ‘v = sin(x)‘,
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like ‘v = sin(x)‘ is
typeset as
{{{
$\nu = \sin(x)$|$v = sin(x)$
}}}
The pipe symbol acts as a delimiter between LaTeX code and the plain text
version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside
‘!bt‘ and ‘!et‘ (begin tex / end tex) instructions.
```

tutorial.gwiki

```
The result looks like this:
{{{
\begin{eqnarray}
{\partial u\over\partial t} &=& \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &=& \nabla\cdot(q(u)\nabla v) + g
\end{eqnarray}
}}}
Of course, such blocks only looks nice in LaTeX. The raw
LaTeX syntax appears in all other formats (but can still be useful
for those who can read LaTeX syntax).

You can have blocks of computer code, starting and ending with
‘!bc‘ and ‘!ec‘ instructions, respectively. Such blocks look like
{{{
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
}}}
It is possible to add a specification of a (ptex2tex−style)
environment for typesetting the verbatim code block, e.g., ‘!bc xxx‘
where ‘xxx‘ is an identifier like ‘pycod‘ for code snippet in Python,
‘sys‘ for terminal session, etc. When Doconce is filtered to LaTeX,
these identifiers are used as in ptex2tex and defined in a
configuration file ‘.ptext2tex.cfg‘, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments):
{{{
# sphinx code−blocks: pycod=python cod=py cppcod=c++ sys=console
}}}
By default, ‘pro‘ and ‘cod‘ are ‘python‘, ‘sys‘ is ‘console‘,
while ‘xpro‘ and ‘xcod‘ are computer language specific for ‘x‘
in ‘f‘ (Fortran), ‘c‘ (C), ‘cpp‘ (C++), and ‘py‘ (Python).
<wiki:comment> ‘rb‘ (Ruby), ‘pl‘ (Perl), and ‘sh‘ (Unix shell). </wiki:comment>

<wiki:comment> (Any sphinx code−block comment, whether inside verbatim code </wi
ki:comment>
<wiki:comment> blocks or outside, yields a mapping between bc arguments </wiki:c
omment>
<wiki:comment> and computer languages. In case of muliple definitions, the </wik
i:comment>
<wiki:comment> first one is used.) </wiki:comment>

One can also copy computer code directly from files, either the
complete file or specified parts.  Computer code is then never
duplicated in the documentation (important for the principle of
avoiding copying information!). A complete file is typeset
with ‘!bc pro‘, while a part of a file is copied into a ‘!bc cod‘
environment. What ‘pro‘ and ‘cod‘ mean is then defined through
a ‘.ptex2tex.cfg‘ file for LaTeX and a ‘sphinx code−blocks‘
comment for Sphinx.

Another document can be included by writing ‘#include "mynote.do.txt"‘
on a line starting with (another) hash sign.  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 the
```

```
right writing enviroment for plain text.

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

Doconce supports a type of macros via a LaTeX-style *newcommand*
construction.  The newcommands defined in a file with name
'newcommand_replace.tex' are expanded when Doconce is filtered to
other formats, except for LaTeX (since LaTeX performs the expansion
itself).  Newcommands in files with names 'newcommands.tex' and
'newcommands_keep.tex' are kept unaltered when Doconce text is
filtered to other formats, except for the Sphinx format. Since Sphinx
understands LaTeX math, but not newcommands if the Sphinx output is
HTML, it makes most sense to expand all newcommands.  Normally, a user
will put all newcommands that appear in math blocks surrounded by
'!bt' and '!et' in 'newcommands_keep.tex' to keep them unchanged, at
least if they contribute to make the raw LaTeX math text easier to
read in the formats that cannot render LaTeX.  Newcommands used
elsewhere throughout the text will usually be placed in
'newcommands_replace.tex' and expanded by Doconce.  The definitions of
newcommands in the 'newcommands*.tex' files *must* appear on a single
line (multi-line newcommands are too hard to parse with regular
expressions).

Recent versions of Doconce also offer cross referencing, typically one
can define labels below (sub)sections, in figure captions, or in
equations, and then refer to these later. Entries in an index can be
defined and result in an index at the end for the LaTeX and Sphinx
formats. Citations to literature, with an accompanying bibliography in
a file, are also supported. The syntax of labels, references,
citations, and the bibliography closely resembles that of LaTeX,
making it easy for Doconce documents to be integrated in LaTeX
projects (manuals, books). For further details on functionality and
syntax we refer to the 'doc/manual/manual.do.txt' file (see the
[https://doconce.googlecode.com/hg/doc/demos/manual/index.html demo page]
for various formats of this document).


<wiki:comment> Example on including another Doconce file (using preprocess): </w
iki:comment>



== From Doconce to Other Formats ==

Transformation of a Doconce document 'mydoc.do.txt' to various other
formats applies the script 'doconce format':
{{{
Terminal> doconce format format mydoc.do.txt
}}}
or just
{{{
Terminal> doconce format format mydoc
}}}
The 'mako' or 'preprocess' programs are always used to preprocess the
file first, and options to 'mako' or 'preprocess' can be added after the
filename. For example,
{{{
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5     # preprocess
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

```
}}}
The variable ‘FORMAT‘ is always defined as the current format when
running ‘preprocess‘. That is, in the last example, ‘FORMAT‘ is
defined as ‘latex‘. Inside the Doconce document one can then perform
format specific actions through tests like ‘#if FORMAT == "latex"‘.

Inline comments in the text are removed from the output by
{{{
Terminal> doconce format latex mydoc remove_inline_comments
}}}
One can also remove such comments from the original Doconce file
by running
source code:
{{{
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.

==== HTML ====

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.

==== LaTeX ====

Making a LaTeX file ‘mydoc.tex‘ from ‘mydoc.do.txt‘ is done in two steps:
<wiki:comment> Note: putting code blocks inside a list is not successful in many
 </wiki:comment>
<wiki:comment> formats − the text may be messed up. A better choice is a paragra
ph </wiki:comment>
<wiki:comment> environment, as used here. </wiki:comment>

*Step 1.* Filter the doconce text to a pre−LaTeX form ‘mydoc.p.tex‘ for
     ‘ptex2tex‘:
{{{
Terminal> doconce format latex mydoc
}}}
LaTeX−specific commands ("newcommands") in math formulas and similar
can be placed in files ‘newcommands.tex‘, ‘newcommands_keep.tex‘, or
‘newcommands_replace.tex‘ (see the section [#Macros_(Newcommands),_Cross−Referen
ces,_Index,_and_Bibliography]).
If these files are present, they are included in the LaTeX document
so that your commands are defined.

*Step 2.* Run ‘ptex2tex‘ (if you have it) to make a standard LaTeX file,
{{{
Terminal> ptex2tex mydoc
}}}
or just perform a plain copy,
{{{
Terminal> cp mydoc.p.tex mydoc.tex
}}}
Doconce generates a ‘.p.tex‘ file with some preprocessor macros
that can be used to steer certain properties of the LaTeX document.
```

```
For example, to turn on the Helvetica font instead of the standard
Computer Modern font, run
{{{
Terminal> ptex2tex −DHELVETICA mydoc
}}}
The title, authors, and date are by default typeset in a non−standard
way to enable a nicer treatment of multiple authors having
institutions in common. However, the standard LaTeX "maketitle" heading
is also available through
{{{
Terminal> ptex2tex −DLATEX_HEADING=traditional mydoc
}}}
A separate titlepage can be generate by
{{{
Terminal> ptex2tex −DLATEX_HEADING=titlepage mydoc
}}}

The ‘ptex2tex‘ tool makes it possible to easily switch between many
different fancy formattings of computer or verbatim 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 cod‘ for a code snippet, where ‘cod‘ is set to
a certain environment in ‘.ptex2tex.cfg‘ (e.g., ‘CodeIntended‘).
There are over 30 styles to choose from.

*Step 3.* Compile ‘mydoc.tex‘
and create the PDF file:
{{{
Terminal> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibitem mydoc     # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc
}}}
If one wishes to use the ‘Minted_Python‘, ‘Minted_Cpp‘, etc., environments
in ‘ptex2tex‘ for typesetting code, the ‘minted‘ LaTeX package is needed.
This package is included by running ‘doconce format‘ with the
‘−DMINTED‘ option:
{{{
Terminal> ptex2tex −DMINTED mydoc
}}}
In this case, ‘latex‘ must be run with the
‘−shell−escape‘ option:
{{{
Terminal> latex −shell−escape mydoc
Terminal> latex −shell−escape mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibitem mydoc     # if bibliography
Terminal> latex −shell−escape mydoc
Terminal> dvipdf mydoc
}}}
The ‘−shell−escape‘ option is required because the ‘minted.sty‘ style
file runs the ‘pygments‘ program to format code, and this program
cannot be run from ‘latex‘ without the ‘−shell−escape‘ option.

==== 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
}}}

==== 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.
That is, one can easily go from Doconce to Microsoft Word.

==== 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 file1 file2 file3 ...
}}}
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 files ‘file1‘,
‘file2‘, etc. that together represent the whole document. Note that
none of the individual Doconce files ‘file1‘, ‘file2‘, etc. should
include the rest as their union makes up the whole document.
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‘‘).

With a single-file document in ‘mydoc.do.txt‘ one often just runs
{{{
Terminal> doconce sphinx_dir mydoc
}}}
and then an appropriate Sphinx directory ‘sphinx-rootdir‘ is made with
relevant files.

The ‘doconce sphinx_dir‘ command generates a script
‘automake-sphinx.sh‘ for compiling the Sphinx document into an HTML
document.  This script copies directories named ‘figs‘ or ‘figures‘
over to the Sphinx directory so that figures are accessible in the
Sphinx compilation.  If figures or movies are located in other
directories, ‘automake-sphinx.sh‘ must be edited accordingly. One
can either run ‘automake-sphinx.sh‘ or perform the steps in the
script manually.
```

```
Doconce comes with a collection of HTML themes for Sphinx documents.
These are packed out in the Sphinx directory, the ‘conf.py‘
configuration file for Sphinx is edited accordingly, and a script
‘make-themes.sh‘ can make HTML documents with one or more themes.
For example,
to realize the themes ‘fenics‘ and ‘pyramid‘, one writes
{{{
Terminal> ./make-themes.sh fenics pyramid
}}}
The resulting directories with HTML documents are ‘_build/html_fenics‘
and ‘_build/html_pyramid‘, respectively. Without arguments,
‘make-themes.sh‘ makes all available themes (!).

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

*Step 1.* Translate Doconce into the Sphinx dialect of
the reStructuredText format:
{{{
Terminal> doconce format sphinx mydoc
}}}

*Step 2.* Create a Sphinx root directory with a ‘conf.py‘ file,
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
}}}

*Step 3.* Copy the ‘tutorial.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‘.

*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
}}}

*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.

==== Google Code Wiki ====

There are several different wiki dialects, but Doconce only support the
one used by [http://code.google.com/p/support/wiki/WikiSyntax Google Code].
The transformation to this format, called ‘gwiki‘ to explicitly mark
it as the Google Code dialect, is done by
{{{
Terminal> doconce format gwiki mydoc.do.txt
}}}
You can then open a new wiki page for your Google Code project, copy
the ‘mydoc.gwiki‘ output file from ‘doconce format‘ and paste the
file contents into the wiki page. Press *Preview* or *Save Page* to
see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

==== 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.

==== Demos ====

The current text is generated from a Doconce format stored in the file
{{{
docs/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.
[https://doconce.googlecode.com/hg/doc/demos/tutorial/index.html Here]
is a sample of how this tutorial looks in different formats.

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.

==== Dependencies ====

If you make use of preprocessor directives in the Doconce source,
either [http://code.google.com/p/preprocess Preprocess] or [http://www.makotempl
ates.org Mako] must be installed.  To make LaTeX
documents (without going through the reStructuredText format) you also
need [http://code.google.com/p/ptex2tex ptex2tex] and some style
files that ‘ptex2tex‘ potentially makes use of.  Going from
reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX
requires [http://docutils.sourceforge.net docutils].  Making Sphinx
documents requires of course [http://sphinx.pocoo.org Sphinx].
All of the mentioned potential dependencies are pure Python packages
which are easily installed.
If translation to [http://johnmacfarlane.net/pandoc/ Pandoc] is desired,
the Pandoc Haskell program must of course be installed.
```

Doconce: Document Once, Include Anywhere Documentation

Release 1.0

Hans Petter Langtangen

October 29, 2011

CONTENTS

Contents:

CHAPTER

ONE

DOCONCE: DOCUMENT ONCE, INCLUDE ANYWHERE

Author Hans Petter Langtangen

Date Oct 29, 2011

- When writing a note, report, manual, etc., do you find it difficult to choose the typesetting format? That is, to choose between plain (email-like) text, wiki, Word/OpenOffice, LaTeX, HTML, reStructuredText, Sphinx, XML, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and then at some later stage eventually go with a particular format?
- Do you need to write documents in varying formats but find it difficult to remember all the typesetting details of various formats like LaTeX, HTML, reStructuredText, Sphinx, and wiki? Would it be convenient to generate the typesetting details of a particular format from a very simple text-like format with minimal tagging?
- Do you have the same information scattered around in different documents in different typesetting formats? Would it be a good idea to write things once, in one format, stored in one place, and include it anywhere?

If any of these questions are of interest, you should keep on reading.

CHAPTER

TWO

THE DOCONCE CONCEPT

Doconce is two things:

1. Doconce is a very simple and minimally tagged markup language that looks like ordinary ASCII text (much like what you would use in an email), but the text can be transformed to numerous other formats, including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx, Epytext, and also plain text (where non-obvious formatting/tags are removed for clear reading in, e.g., emails). From reStructuredText you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the latter to RTF and MS Word. (An experimental translator to Pandoc is under development, and from Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML, OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)
2. Doconce is a working strategy for never duplicating information. Text is written in a single place and then transformed to a number of different destinations of diverse type (software source code, manuals, tutorials, books, wikis, memos, emails, etc.). The Doconce markup language support this working strategy. The slogan is: "Write once, include anywhere".

Here are some Doconce features:

- Doconce markup does include tags, so the format is more tagged than Markdown and Pandoc, but less than reST, and very much less than LaTeX and HTML.
- Doconce can be converted to plain *untagged* text, often desirable for computer programs and email.
- Doconce has good support for copying in parts of computer code, say in examples, directly from the source code files.
- Doconce has full support for LaTeX math, and integrates very well with big LaTeX projects (books).
- Doconce is almost self-explanatory and is a handy starting point for generating documents in more complicated markup languages, such as Google wiki, LaTeX, and Sphinx. A primary application of Doconce is just to make the initial versions of a Sphinx or wiki document.
- Contrary to the similar Pandoc translator, Doconce integrates with Sphinx and Google wiki. However, if these formats are not of interest, Pandoc is obviously a superior tool.

Doconce was particularly written for the following sample applications:

- Large books written in LaTeX, but where many pieces (computer demos, projects, examples) can be written in Doconce to appear in other contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.
- Software documentation, primarily Python doc strings, which one wants to appear as plain untagged text for viewing in Pydoc, as reStructuredText for use with Sphinx, as wiki text when publishing the software at web sites, and as LaTeX integrated in, e.g., a thesis.
- Quick memos, which start as plain text in email, then some small amount of Doconce tagging is added, before the memos can appear as Sphinx web pages, MS Word documents, or in wikis.

History: Doconce was developed in 2006 at a time when most popular markup languages used quite some tagging. Later, almost untagged markup languages like Markdown and Pandoc became popular. Doconce is not a replacement of Pandoc, which is a considerably more sophisticated project. Moreover, Doconce was developed mainly to fulfill the needs for a flexible source code base for books with much mathematics and computer code.

Disclaimer: Doconce is a simple tool, largely based on interpreting and handling text through regular expressions. The possibility for tweaking the layout is obviously limited since the text can go to all sorts of sophisticated markup languages. Moreover, because of limitations of regular expressions, some formatting of Doconce syntax may face problems when transformed to HTML, LaTeX, Sphinx, and similar formats.

CHAPTER

THREE

WHAT DOES DOCONCE LOOK LIKE?

Doconce text looks like ordinary text, but there are some almost invisible text constructions that allow you to control the formating. For example,

- bullet lists arise from lines starting with an asterisk,
- *emphasized words* are surrounded by asterisks,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim (monospace font),
- section headings are recognied by equality (=) signs before and after the text, and the number of = signs indicates the level of the section (7 for main section, 5 for subsection, 3 for subsubsection),
- paragraph headings are recognized by a double underscore before and after the heading,
- blocks of computer code can easily be included by placing `!bc` (begin code) and `!ec` (end code) commands at separate lines before and after the code block,
- blocks of computer code can also be imported from source files,
- blocks of LaTeX mathematics can easily be included by placing `!bt` (begin TeX) and `!et` (end TeX) commands at separate lines before and after the math block,
- there is support for both LaTeX and text-like inline mathematics,
- figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported,
- comments can be inserted throughout the text (`#` at the beginning of a line),
- with a simple preprocessor, Preprocess or Mako, one can include other documents (files) and large portions of text can be defined in or out of the text,
- with the Mako preprocessor one can even embed Python code and use this to steer generation of Doconce text.

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

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

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and `computer` words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,
```

```
  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in "hpl":"http://folk.uio.no/hpl".
If the word is URL, the URL itself becomes the link name,
as in "URL":"tutorial.do.txt".

References to sections may use logical names as labels (e.g., a
"label" command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |--------------------------------|
  |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    |
  |--------------------------------|

# lines beginning with # are comment lines
```

The Doconce text above results in the following little document:

3.1 A Subsection with Sample Text

Ordinary text looks like ordinary text, and the tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Lists are typeset as you would do in an email,

- item 1
- item 2
- item 3

Lists can also have numbered items instead of bullets, just use an o (for ordered) instead of the asterisk:

1. item 1
2. item 2
3. item 3

URLs with a link word are possible, as in hpl. If the word is URL, the URL itself becomes the link name, as in tutorial.do.txt.

References to sections may use logical names as labels (e.g., a "label" command right after the section title), as in the reference to the section *A Subsection with Sample Text*.

Doconce also allows inline comments such as (**hpl**: here I will make some remarks to the text) for allowing authors to make notes. Inline comments can be removed from the output by a command-line argument (see the section *From Doconce to Other Formats* for an example).

Tables are also supperted, e.g.,

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

3.2 Mathematics and Computer Code

Inline mathematics, such as $\nu = \sin(x)$, allows the formula to be specified both as LaTeX and as plain text. This results in a professional LaTeX typesetting, but in other formats the text version normally looks better than raw LaTeX mathematics with backslashes. An inline formula like $\nu = \sin(x)$ is typeset as

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

The pipe symbol acts as a delimiter between LaTeX code and the plain text version of the formula.

Blocks of mathematics are better typeset with raw LaTeX, inside `!bt` and `!et` (begin tex / end tex) instructions. The result looks like this:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u + f, \\ \frac{\partial v}{\partial t} &= \nabla \cdot (q(u)\nabla v) + g \end{aligned} \tag{3.1}$$

Of course, such blocks only looks nice in LaTeX. The raw LaTeX syntax appears in all other formats (but can still be useful for those who can read LaTeX syntax).

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

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

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

It is possible to add a specification of a (ptex2tex-style) environment for typesetting the verbatim code block, e.g., `!bc xxx` where `xxx` is an identifier like `pycod` for code snippet in Python, `sys` for terminal session, etc. When Doconce is filtered to LaTeX, these identifiers are used as in ptex2tex and defined in a configuration file `.ptext2tex.cfg`, while when filtering to Sphinx, one can have a comment line in the Doconce file for mapping the identifiers to legal language names for Sphinx (which equals the legal language names for Pygments):

```
# sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console
```

By default, `pro` and `cod` are `python`, `sys` is `console`, while `xpro` and `xcod` are computer language specific for `x` in `f` (Fortran), `c` (C), `cpp` (C++), and `py` (Python). .. `rb` (Ruby), `pl` (Perl), and `sh` (Unix shell).

One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!). A complete file is typeset with `!bc pro`, while a part of a file is copied into a `!bc cod` environment. What `pro` and `cod` mean is then defined through a `.ptex2tex.cfg` file for LaTeX and a `sphinx code-blocks` comment for Sphinx.

Another document can be included by writing `#include "mynote.do.txt"` on a line starting with (another) hash sign. 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 the right writing enviroment for plain text.

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

Doconce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands defined in a file with name `newcommand_replace.tex` are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names `newcommands.tex` and `newcommands_keep.tex` are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by `!bt` and `!et` in `newcommands_keep.tex` to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in `newcommands_replace.tex` and expanded by Doconce. The definitions of newcommands in the `newcommands*.tex` files *must* appear on a single line (multi-line newcommands are too hard to parse with regular expressions).

Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the `doc/manual/manual.do.txt` file (see the demo page for various formats of this document).

CHAPTER

FOUR

FROM DOCONCE TO OTHER FORMATS

Transformation of a Doconce document `mydoc.do.txt` to various other formats applies the script `doconce format`:

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

or just

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

The `mako` or `preprocess` programs are always used to preprocess the file first, and options to `mako` or `preprocess` can be added after the filename. For example,

```
Terminal> doconce format latex mydoc -Dextra_sections -DVAR1=5     # preprocess
Terminal> doconce format latex yourdoc extra_sections=True VAR1=5  # mako
```

The variable `FORMAT` is always defined as the current format when running `preprocess`. That is, in the last example, `FORMAT` is defined as `latex`. Inside the Doconce document one can then perform format specific actions through tests like `#if FORMAT == "latex"`.

Inline comments in the text are removed from the output by

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

One can also remove such comments from the original Doconce file by running source code:

```
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.

4.1 HTML

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.

4.2 LaTeX

Making a LaTeX file `mydoc.tex` from `mydoc.do.txt` is done in two steps: .. Note: putting code blocks inside a list is not successful in many

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

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

LaTeX-specific commands ("newcommands") in math formulas and similar can be placed in files `newcommands.tex`, `newcommands_keep.tex`, or `newcommands_replace.tex` (see the section *Macros (Newcommands), Cross-References, Index, and Bibliography*). If these files are present, they are included in the LaTeX document so that your commands are defined.

Step 2. Run `ptex2tex` (if you have it) to make a standard LaTeX file,

```
Terminal> ptex2tex mydoc
```

or just perform a plain copy,

```
Terminal> cp mydoc.p.tex mydoc.tex
```

Doconce generates a `.p.tex` file with some preprocessor macros that can be used to steer certain properties of the LaTeX document. For example, to turn on the Helvetica font instead of the standard Computer Modern font, run

```
Terminal> ptex2tex -DHELVETICA mydoc
```

The title, authors, and date are by default typeset in a non-standard way to enable a nicer treatment of multiple authors having institutions in common. However, the standard LaTeX "maketitle" heading is also available through

```
Terminal> ptex2tex -DLATEX_HEADING=traditional mydoc
```

A separate titlepage can be generate by

```
Terminal> ptex2tex -DLATEX_HEADING=titlepage mydoc
```

The `ptex2tex` tool makes it possible to easily switch between many different fancy formattings of computer or verbatim 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 cod` for a code snippet, where `cod` is set to a certain environment in `.ptex2tex.cfg` (e.g., `CodeIntended`). There are over 30 styles to choose from.

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

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

If one wishes to use the `Minted_Python`, `Minted_Cpp`, etc., environments in `ptex2tex` for typesetting code, the `minted` LaTeX package is needed. This package is included by running `doconce format` with the `-DMINTED` option:

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

In this case, `latex` must be run with the `-shell-escape` option:

```
Terminal> latex -shell-escape mydoc
Terminal> latex -shell-escape mydoc
```

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

The `-shell-escape` option is required because the `minted.sty` style file runs the `pygments` program to format code, and this program cannot be run from `latex` without the `-shell-escape` option.

4.3 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
```

4.4 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. That is, one can easily go from Doconce to Microsoft Word.

4.5 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 file1 file2 file3 ...
```

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 files `file1`, `file2`, etc. that together represent the whole document. Note that none of the individual Doconce files `file1`, `file2`, etc. should include the rest as their union makes up the whole document. 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'`).

With a single-file document in `mydoc.do.txt` one often just runs

```
Terminal> doconce sphinx_dir mydoc
```

and then an appropriate Sphinx directory `sphinx-rootdir` is made with relevant files.

The `doconce sphinx_dir` command generates a script `automake-sphinx.sh` for compiling the Sphinx document into an HTML document. This script copies directories named `figs` or `figures` over to the Sphinx directory so that figures are accessible in the Sphinx compilation. If figures or movies are located in other directories, `automake-sphinx.sh` must be edited accordingly. One can either run `automake-sphinx.sh` or perform the steps in the script manually.

Doconce comes with a collection of HTML themes for Sphinx documents. These are packed out in the Sphinx directory, the `conf.py` configuration file for Sphinx is edited accordingly, and a script `make-themes.sh` can make HTML documents with one or more themes. For example, to realize the themes `fenics` and `pyramid`, one writes

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

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

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

Step 1. Translate Doconce into the Sphinx dialect of the reStructuredText format:

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

Step 2. Create a Sphinx root directory with a `conf.py` file, 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
```

Step 3. Copy the `tutorial.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`.

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
```

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.

4.6 Google Code Wiki

There are several different wiki dialects, but Doconce only support the one used by Google Code. The transformation to this format, called `gwiki` to explicitly mark it as the Google Code dialect, is done by

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

You can then open a new wiki page for your Google Code project, copy the `mydoc.gwiki` output file from `doconce format` and paste the file contents into the wiki page. Press **Preview** or **Save Page** to see the formatted result.

When the Doconce file contains figures, each figure filename must 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).

4.7 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.

4.8 Demos

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

```
docs/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. Here is a sample of how this tutorial looks in different formats.

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.

4.9 Dependencies

If you make use of preprocessor directives in the Doconce source, either Preprocess or Mako must be installed. To make LaTeX documents (without going through the reStructuredText format) you also need ptex2tex and some style files that `ptex2tex` potentially makes use of. Going from reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX requires docutils. Making Sphinx documents requires of course Sphinx. All of the mentioned potential dependencies are pure Python packages which are easily installed. If translation to Pandoc is desired, the Pandoc Haskell program must of course be installed.

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

tutorial.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE document PUBLIC "+//IDN docutils.sourceforge.net//DTD Docutils Generic
//EN//XML" "http://docutils.sourceforge.net/docs/ref/docutils.dtd">
<!-- Generated by Docutils 0.9 -->
<document source="tutorial.rst"><comment xml:space="preserve">Automatically gene
rated reST file from Doconce source
(http://code.google.com/p/doconce/)</comment><section ids="doconce-document-once
-include-anywhere" names="doconce:\ document\ once,\ include\ anywhere"><title>D
oconce: Document Once, Include Anywhere</title><field_list><field><field_name>Au
thor</field_name><field_body><paragraph>Hans Petter Langtangen</paragraph></fiel
d_body></field><field><field_name>Date</field_name><field_body><paragraph>Oct 29
, 2011</paragraph><bullet_list bullet="*"><list_item><paragraph>When writing a n
ote, report, manual, etc., do you find it difficult
to choose the typesetting format? That is, to choose between plain
(email-like) text, wiki, Word/OpenOffice, LaTeX, HTML,
reStructuredText, Sphinx, XML, etc.  Would it be convenient to
start with some very simple text-like format that easily converts
to the formats listed above, and then at some later stage
eventually go with a particular format?</paragraph></list_item><list_item><parag
raph>Do you need to write documents in varying formats but find it
difficult to remember all the typesetting details of various
formats like <reference name="LaTeX" refuri="http://refcards.com/docs/silvermanj
/amslatex/LaTeXRefCard.v2.0.pdf">LaTeX</reference><target ids="latex" names="lat
ex" refuri="http://refcards.com/docs/silvermanj/amslatex/LaTeXRefCard.v2.0.pdf"/
>, <reference name="HTML" refuri="http://www.htmlcodetutorial.com/">HTML</refere
nce><target ids="html" names="html" refuri="http://www.htmlcodetutorial.com/"/>,
 <reference name="reStructuredText" refuri="http://docutils.sourceforge.net/docs
/ref/rst/restructuredtext.html">reStructuredText</reference><target ids="restruc
turedtext" names="restructuredtext" refuri="http://docutils.sourceforge.net/docs
/ref/rst/restructuredtext.html"/>, <reference name="Sphinx" refuri="http://sphin
x.pocoo.org/contents.html">Sphinx</reference><target dupnames="sphinx" ids="sphi
nx" refuri="http://sphinx.pocoo.org/contents.html"/>, and <reference name="wiki"
 refuri="http://code.google.com/p/support/wiki/WikiSyntax">wiki</reference><targ
et ids="wiki" names="wiki" refuri="http://code.google.com/p/support/wiki/WikiSyn
tax"/>? Would it be convenient
to generate the typesetting details of a particular format from a
very simple text-like format with minimal tagging?</paragraph></list_item><list_
item><paragraph>Do you have the same information scattered around in different
documents in different typesetting formats? Would it be a good idea
to write things once, in one format, stored in one place, and
include it anywhere?</paragraph></list_item></bullet_list></field_body></field><
/field_list><paragraph>If any of these questions are of interest, you should kee
p on reading.</paragraph></section><section ids="the-doconce-concept" names="the
\ doconce\ concept"><title>The Doconce Concept</title><paragraph>Doconce is two
things:</paragraph><block_quote><enumerated_list enumtype="arabic" prefix="" suf
fix="."><list_item><paragraph>Doconce is a very simple and minimally tagged mark
up language that
looks like ordinary ASCII text (much like what you would use in an
email), but the text can be transformed to numerous other formats,
including HTML, wiki, LaTeX, PDF, reStructuredText (reST), Sphinx,
Epytext, and also plain text (where non-obvious formatting/tags are
removed for clear reading in, e.g., emails). From reStructuredText
you can go to XML, HTML, LaTeX, PDF, OpenOffice, and from the
latter to RTF and MS Word.
(An experimental translator to Pandoc is under development, and from
Pandoc one can generate Markdown, reST, LaTeX, HTML, PDF, DocBook XML,
OpenOffice, GNU Texinfo, MediaWiki, RTF, Groff, and other formats.)</paragraph><
/list_item><list_item><paragraph>Doconce is a working strategy for never duplica
ting information.
```

tutorial.xml

```
Text is written in a single place and then transformed to
a number of different destinations of diverse type (software
source code, manuals, tutorials, books, wikis, memos, emails, etc.).
The Doconce markup language support this working strategy.
The slogan is: &quot;Write once, include anywhere&quot;.</paragraph></list_item>
</enumerated_list></block_quote><paragraph>Here are some Doconce features:</para
graph><block_quote><bullet_list bullet="*"><list_item><paragraph>Doconce markup
does include tags, so the format is more tagged than
Markdown and Pandoc, but less than reST, and very much less than
LaTeX and HTML.</paragraph></list_item><list_item><paragraph>Doconce can be conv
erted to plain <emphasis>untagged</emphasis> text,
often desirable for computer programs and email.</paragraph></list_item><list_it
em><paragraph>Doconce has good support for copying in parts of computer code,
say in examples, directly from the source code files.</paragraph></list_item><li
st_item><paragraph>Doconce has full support for LaTeX math, and integrates very
well
with big LaTeX projects (books).</paragraph></list_item><list_item><paragraph>Do
conce is almost self-explanatory and is a handy starting point
for generating documents in more complicated markup languages, such
as Google wiki, LaTeX, and Sphinx. A primary application of Doconce
is just to make the initial versions of a Sphinx or wiki document.</paragraph></
list_item><list_item><paragraph>Contrary to the similar Pandoc translator, Docon
ce integrates with
Sphinx and Google wiki. However, if these formats are not of interest,
Pandoc is obviously a superior tool.</paragraph></list_item></bullet_list></bloc
k_quote><paragraph>Doconce was particularly written for the following sample app
lications:</paragraph><block_quote><bullet_list bullet="*"><list_item><paragraph
>Large books written in LaTeX, but where many pieces (computer demos,
projects, examples) can be written in Doconce to appear in other
contexts in other formats, including plain HTML, Sphinx, wiki, or MS Word.</para
graph></list_item><list_item><paragraph>Software documentation, primarily Python
 doc strings, which one wants
to appear as plain untagged text for viewing in Pydoc, as reStructuredText
for use with Sphinx, as wiki text when publishing the software at
web sites, and as LaTeX integrated in, e.g., a thesis.</paragraph></list_item><l
ist_item><paragraph>Quick memos, which start as plain text in email, then some s
mall
amount of Doconce tagging is added, before the memos can appear as
Sphinx web pages, MS Word documents, or in wikis.</paragraph></list_item></bulle
t_list></block_quote><paragraph>History: Doconce was developed in 2006 at a time
 when most popular
markup languages used quite some tagging.  Later, almost untagged
markup languages like Markdown and Pandoc became popular. Doconce is
not a replacement of Pandoc, which is a considerably more
sophisticated project. Moreover, Doconce was developed mainly to
fulfill the needs for a flexible source code base for books with much
mathematics and computer code.</paragraph><paragraph>Disclaimer: Doconce is a si
mple tool, largely based on interpreting
and handling text through regular expressions. The possibility for
tweaking the layout is obviously limited since the text can go to
all sorts of sophisticated markup languages. Moreover, because of
limitations of regular expressions, some formatting of Doconce syntax
may face problems when transformed to HTML, LaTeX, Sphinx, and similar
formats.</paragraph></section><section ids="what-does-doconce-look-like" names="
what\ does\ doconce\ look\ like?"><title>What Does Doconce Look Like?</title><pa
ragraph>Doconce text looks like ordinary text, but there are some almost invisib
le
text constructions that allow you to control the formating. For example,</paragr
aph><block_quote><bullet_list bullet="*"><list_item><paragraph>bullet lists aris
```

tutorial.xml

```
e from lines starting with an asterisk,</paragraph></list_item><list_item><parag
raph><emphasis>emphasized words</emphasis> are surrounded by asterisks,</paragra
ph></list_item><list_item><paragraph><strong>words in boldface</strong> are surr
ounded by underscores,</paragraph></list_item><list_item><paragraph>words from c
omputer code are enclosed in back quotes and
then typeset verbatim (monospace font),</paragraph></list_item><list_item><parag
raph>section headings are recognied by equality (<literal>=</literal>) signs bef
ore
and after the text, and the number of <literal>=</literal> signs indicates the
level of the section (7 for main section, 5 for subsection,
3 for subsubsection),</paragraph></list_item><list_item><paragraph>paragraph hea
dings are recognized by a double underscore
before and after the heading,</paragraph></list_item><list_item><paragraph>block
s of computer code can easily be included by placing
<literal>!bc</literal> (begin code) and <literal>!ec</literal> (end code) comman
ds at separate lines
before and after the code block,</paragraph></list_item><list_item><paragraph>bl
ocks of computer code can also be imported from source files,</paragraph></list_
item><list_item><paragraph>blocks of LaTeX mathematics can easily be included by
 placing
<literal>!bt</literal> (begin TeX) and <literal>!et</literal> (end TeX) commands
 at separate lines
before and after the math block,</paragraph></list_item><list_item><paragraph>th
ere is support for both LaTeX and text-like inline mathematics,</paragraph></lis
t_item><list_item><paragraph>figures and movies with captions, simple tables,
URLs with links, index list, labels and references are supported,</paragraph></l
ist_item><list_item><paragraph>comments can be inserted throughout the text (<li
teral>#</literal> at the beginning
of a line),</paragraph></list_item><list_item><paragraph>with a simple preproces
sor, Preprocess or Mako, one can include
other documents (files) and large portions of text can be defined
in or out of the text,</paragraph></list_item><list_item><paragraph>with the Mak
o preprocessor one can even embed Python
code and use this to steer generation of Doconce text.</paragraph></list_item></
bullet_list></block_quote><paragraph>Here is an example of some simple text writ
ten in the Doconce format:</paragraph><literal_block xml:space="preserve">=====
A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, and the tags used for
_boldface_ words, *emphasized* words, and ‘computer‘ words look
natural in plain text.  Lists are typeset as you would do in an email,

  * item 1
  * item 2
  * item 3

Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3

URLs with a link word are possible, as in &quot;hpl&quot;:&quot;http://folk.uio.
no/hpl&quot;.
If the word is URL, the URL itself becomes the link name,
as in &quot;URL&quot;:&quot;tutorial.do.txt&quot;.

References to sections may use logical names as labels (e.g., a
```

tutorial.xml

```
&quot;label&quot; command right after the section title), as in the reference to
Section ref{my:first:sec}.

Doconce also allows inline comments such as [hpl: here I will make
some remarks to the text] for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see Section ref{doconce2formats} for an example).

Tables are also supperted, e.g.,

  |--------------------------------|
  |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    |
  |--------------------------------|

# lines beginning with # are comment lines</literal_block><paragraph>The Doconce
 text above results in the following little document:</paragraph><target refid="
my-first-sec"/><section ids="a-subsection-with-sample-text my-first-sec" names="
a\ subsection\ with\ sample\ text my:first:sec"><title>A Subsection with Sample
Text</title><paragraph>Ordinary text looks like ordinary text, and the tags used
 for
<strong>boldface</strong> words, <emphasis>emphasized</emphasis> words, and <lit
eral>computer</literal> words look
natural in plain text.  Lists are typeset as you would do in an email,</paragrap
h><block_quote><bullet_list bullet="*"><list_item><paragraph>item 1</paragraph><
/list_item><list_item><paragraph>item 2</paragraph></list_item><list_item><parag
raph>item 3</paragraph></list_item></bullet_list></block_quote><paragraph>Lists
can also have numbered items instead of bullets, just use an <literal>o</literal
>
(for ordered) instead of the asterisk:</paragraph><block_quote><enumerated_list
enumtype="arabic" prefix="" suffix="."><list_item><paragraph>item 1</paragraph><
/list_item><list_item><paragraph>item 2</paragraph></list_item><list_item><parag
raph>item 3</paragraph></list_item></enumerated_list></block_quote><paragraph>UR
Ls with a link word are possible, as in <reference name="hpl" refuri="http://fol
k.uio.no/hpl">hpl</reference><target ids="hpl" names="hpl" refuri="http://folk.u
io.no/hpl"/>.
If the word is URL, the URL itself becomes the link name,
as in <reference name="tutorial.do.txt" refuri="tutorial.do.txt">tutorial.do.txt
</reference><target ids="tutorial-do-txt" names="tutorial.do.txt" refuri="tutori
al.do.txt"/>.</paragraph><paragraph>References to sections may use logical names
 as labels (e.g., a
&quot;label&quot; command right after the section title), as in the reference to
the section <reference name="A Subsection with Sample Text" refid="a-subsection-
with-sample-text">A Subsection with Sample Text</reference>.</paragraph><paragra
ph>Doconce also allows inline comments such as (<strong>hpl</strong>: here I wil
l make
some remarks to the text) for allowing authors to make notes. Inline
comments can be removed from the output by a command-line argument
(see the section <reference name="From Doconce to Other Formats" refid="from-doc
once-to-other-formats">From Doconce to Other Formats</reference> for an example)
.</paragraph><paragraph>Tables are also supperted, e.g.,</paragraph><table><tgro
up cols="3"><colspec colwidth="12"/><colspec colwidth="12"/><colspec colwidth="1
2"/><thead><row><entry><paragraph>time</paragraph></entry><entry><paragraph>velo
city</paragraph></entry><entry><paragraph>acceleration</paragraph></entry></row>
</thead><tbody><row><entry><paragraph>0.0</paragraph></entry><entry><paragraph>1
.4186</paragraph></entry><entry><paragraph>-5.01</paragraph></entry></row><row><
```

tutorial.xml

```
entry><paragraph>2.0</paragraph></entry><entry><paragraph>1.376512</paragraph></
entry><entry><paragraph>11.919</paragraph></entry></row><row><entry><paragraph>4
.0</paragraph></entry><entry><paragraph>1.1E+1</paragraph></entry><entry><paragr
aph>14.717624</paragraph></entry></row></tbody></tgroup></table></section><secti
on ids="mathematics-and-computer-code" names="mathematics\ and\ computer\ code">
<title>Mathematics and Computer Code</title><paragraph>Inline mathematics, such
as v = sin(x),
allows the formula to be specified both as LaTeX and as plain text.
This results in a professional LaTeX typesetting, but in other formats
the text version normally looks better than raw LaTeX mathematics with
backslashes. An inline formula like v = sin(x) is
typeset as:</paragraph><literal_block xml:space="preserve">$\nu = \sin(x)$|$v =
sin(x)$</literal_block><paragraph>The pipe symbol acts as a delimiter between La
TeX code and the plain text
version of the formula.</paragraph><paragraph>Blocks of mathematics are better t
ypeset with raw LaTeX, inside
<literal>!bt</literal> and <literal>!et</literal> (begin tex / end tex) instruct
ions.
The result looks like this:</paragraph><literal_block xml:space="preserve">\begi
n{eqnarray}
{\partial u\over\partial t} &amp;=&amp; \nabla^2 u + f, label{myeq1}\\
{\partial v\over\partial t} &amp;=&amp; \nabla\cdot(q(u)\nabla v) + g
\end{eqnarray}</literal_block><paragraph>Of course, such blocks only looks nice
in LaTeX. The raw
LaTeX syntax appears in all other formats (but can still be useful
for those who can read LaTeX syntax).</paragraph><paragraph>You can have blocks
of computer code, starting and ending with
<literal>!bc</literal> and <literal>!ec</literal> instructions, respectively. Su
ch blocks look like:</paragraph><literal_block xml:space="preserve">from math im
port sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)</literal_block><paragraph>It is po
ssible to add a specification of a (ptex2tex-style)
environment for typesetting the verbatim code block, e.g., <literal>!bc xxx</lit
eral>
where <literal>xxx</literal> is an identifier like <literal>pycod</literal> for
code snippet in Python,
<literal>sys</literal> for terminal session, etc. When Doconce is filtered to La
TeX,
these identifiers are used as in ptex2tex and defined in a
configuration file <literal>.ptext2tex.cfg</literal>, while when filtering
to Sphinx, one can have a comment line in the Doconce file for
mapping the identifiers to legal language names for Sphinx (which equals
the legal language names for Pygments):</paragraph><literal_block xml:space="pre
serve"># sphinx code-blocks: pycod=python cod=py cppcod=c++ sys=console</literal
_block><paragraph>By default, <literal>pro</literal> and <literal>cod</literal>
are <literal>python</literal>, <literal>sys</literal> is <literal>console</liter
al>,
while <literal>xpro</literal> and <literal>xcod</literal> are computer language
specific for <literal>x</literal>
in <literal>f</literal> (Fortran), <literal>c</literal> (C), <literal>cpp</liter
al> (C++), and <literal>py</literal> (Python).
.. <literal>rb</literal> (Ruby), <literal>pl</literal> (Perl), and <literal>sh</
literal> (Unix shell).</paragraph><comment xml:space="preserve">(Any sphinx code
-block comment, whether inside verbatim code</comment><comment xml:space="preser
ve">blocks or outside, yields a mapping between bc arguments</comment><comment x
```

tutorial.xml

ml:space="preserve">and computer languages. In case of muliple definitions, the</comment><comment xml:space="preserve">first one is used.)</comment><paragraph>One can also copy computer code directly from files, either the complete file or specified parts. Computer code is then never duplicated in the documentation (important for the principle of avoiding copying information!). A complete file is typeset with <literal>!bc pro</literal>, while a part of a file is copied into a <literal>!bc cod</literal> environment. What <literal>pro</literal> and <literal>cod</literal> mean is then defined through a <literal>.ptex2tex.cfg</literal> file for LaTeX and a <literal>sphinx code-blocks</literal> comment for Sphinx.</paragraph><paragraph>Another document can be included by writing <literal>#include "mynote.do.txt"</literal> on a line starting with (another) hash sign. Doconce documents have extension <literal>do.txt</literal>. The <literal>do</literal> part stands for doconce, while the trailing <literal>.txt</literal> denotes a text document so that editors gives you the right writing enviroment for plain text.</paragraph><target refid="newcommands"/></section><section ids="macros-newcommands-cross-references-index-and-bibliography newcommands" names="macros\ (newcommands),\ cross-references,\ index,\ and\ bibliography newcommands"><title>Macros (Newcommands), Cross-References, Index, and Bibliography</title><paragraph>Doconce supports a type of macros via a LaTeX-style <emphasis>newcommand</emphasis> construction. The newcommands defined in a file with name <literal>newcommand_replace.tex</literal> are expanded when Doconce is filtered to other formats, except for LaTeX (since LaTeX performs the expansion itself). Newcommands in files with names <literal>newcommands.tex</literal> and <literal>newcommands_keep.tex</literal> are kept unaltered when Doconce text is filtered to other formats, except for the Sphinx format. Since Sphinx understands LaTeX math, but not newcommands if the Sphinx output is HTML, it makes most sense to expand all newcommands. Normally, a user will put all newcommands that appear in math blocks surrounded by <literal>!bt</literal> and <literal>!et</literal> in <literal>newcommands_keep.tex</literal> to keep them unchanged, at least if they contribute to make the raw LaTeX math text easier to read in the formats that cannot render LaTeX. Newcommands used elsewhere throughout the text will usually be placed in <literal>newcommands_replace.tex</literal> and expanded by Doconce. The definitions of newcommands in the <literal>newcommands*.tex</literal> files <emphasis>must</emphasis> appear on a single line (multi-line newcommands are too hard to parse with regular expressions).</paragraph><paragraph>Recent versions of Doconce also offer cross referencing, typically one can define labels below (sub)sections, in figure captions, or in equations, and then refer to these later. Entries in an index can be defined and result in an index at the end for the LaTeX and Sphinx formats. Citations to literature, with an accompanying bibliography in a file, are also supported. The syntax of labels, references, citations, and the bibliography closely resembles that of LaTeX, making it easy for Doconce documents to be integrated in LaTeX projects (manuals, books). For further details on functionality and syntax we refer to the <literal>doc/manual/manual.do.txt</literal> file (see the <reference name="demo page" refuri="https://doconce.googlecode.com/hg/doc/demos/manual/index.html">demo page</reference><target ids="demo-page" names="demo\ page" refuri="https://doconce.googlecode.com/hg/doc/demos/manual/index.html"/>

tutorial.xml

```
for various formats of this document).</paragraph><comment xml:space="preserve">
Example on including another Doconce file (using preprocess):</comment><target r
efid="doconce2formats"/></section></section><section ids="from-doconce-to-other-
formats doconce2formats" names="from\ doconce\ to\ other\ formats doconce2format
s"><title>From Doconce to Other Formats</title><paragraph>Transformation of a Do
conce document <literal>mydoc.do.txt</literal> to various other
formats applies the script <literal>doconce format</literal>:</paragraph><litera
l_block xml:space="preserve">Terminal&gt; doconce format format mydoc.do.txt</li
teral_block><paragraph>or just:</paragraph><literal_block xml:space="preserve">T
erminal&gt; doconce format format mydoc</literal_block><paragraph>The <literal>m
ako</literal> or <literal>preprocess</literal> programs are always used to prepr
ocess the
file first, and options to <literal>mako</literal> or <literal>preprocess</liter
al> can be added after the
filename. For example:</paragraph><literal_block xml:space="preserve">Terminal&g
t; doconce format latex mydoc -Dextra_sections -DVAR1=5     # preprocess
Terminal&gt; doconce format latex yourdoc extra_sections=True VAR1=5  # mako</li
teral_block><paragraph>The variable <literal>FORMAT</literal> is always defined
as the current format when
running <literal>preprocess</literal>. That is, in the last example, <literal>FO
RMAT</literal> is
defined as <literal>latex</literal>. Inside the Doconce document one can then pe
rform
format specific actions through tests like <literal>#if FORMAT == &amp;quot;latex&qu
ot;</literal>.</paragraph><paragraph>Inline comments in the text are removed fro
m the output by:</paragraph><literal_block xml:space="preserve">Terminal&gt; doc
once format latex mydoc remove_inline_comments</literal_block><paragraph>One can
 also remove such comments from the original Doconce file
by running
source code:</paragraph><literal_block xml:space="preserve">Terminal&gt; doconce
 remove_inline_comments mydoc</literal_block><paragraph>This action is convenien
t when a Doconce document reaches its final form
and comments by different authors should be removed.</paragraph><section dupname
s="html" ids="id1"><title>HTML</title><paragraph>Making an HTML version of a Doc
once file <literal>mydoc.do.txt</literal>
is performed by:</paragraph><literal_block xml:space="preserve">Terminal&gt; doc
once format html mydoc</literal_block><paragraph>The resulting file <literal>myd
oc.html</literal> can be loaded into any web browser for viewing.</paragraph></s
ection><section dupnames="latex" ids="id2"><title>LaTeX</title><paragraph>Making
 a LaTeX file <literal>mydoc.tex</literal> from <literal>mydoc.do.txt</literal>
is done in two steps:
.. Note: putting code blocks inside a list is not successful in many</paragraph>
<comment xml:space="preserve">formats - the text may be messed up. A better choi
ce is a paragraph</comment><comment xml:space="preserve">environment, as used he
re.</comment><definition_list><definition_list_item><term><emphasis>Step 1.</emp
hasis> Filter the doconce text to a pre-LaTeX form <literal>mydoc.p.tex</literal
> for</term><definition><paragraph><literal>ptex2tex</literal>:</paragraph><lite
ral_block xml:space="preserve">Terminal&gt; doconce format latex mydoc</literal_
block></definition></definition_list_item></definition_list><paragraph>LaTeX-spe
cific commands (&quot;newcommands&quot;) in math formulas and similar
can be placed in files <literal>newcommands.tex</literal>, <literal>newcommands_
keep.tex</literal>, or
<literal>newcommands_replace.tex</literal> (see the section <reference name="Mac
ros (Newcommands), Cross-References, Index, and Bibliography" refid="macros-newc
ommands-cross-references-index-and-bibliography">Macros (Newcommands), Cross-Ref
erences, Index, and Bibliography</reference>).
If these files are present, they are included in the LaTeX document
so that your commands are defined.</paragraph><paragraph><emphasis>Step 2.</emph
asis> Run <literal>ptex2tex</literal> (if you have it) to make a standard LaTeX
```

tutorial.xml

file:</paragraph><literal_block xml:space="preserve">Terminal> ptex2tex mydoc
</literal_block><paragraph>or just perform a plain copy:</paragraph><literal_blo
ck xml:space="preserve">Terminal> cp mydoc.p.tex mydoc.tex</literal_block><pa
ragraph>Doconce generates a <literal>.p.tex</literal> file with some preprocesso
r macros
that can be used to steer certain properties of the LaTeX document.
For example, to turn on the Helvetica font instead of the standard
Computer Modern font, run:</paragraph><literal_block xml:space="preserve">Termin
al> ptex2tex −DHELVETICA mydoc</literal_block><paragraph>The title, authors,
and date are by default typeset in a non−standard
way to enable a nicer treatment of multiple authors having
institutions in common. However, the standard LaTeX "maketitle" headin
g
is also available through:</paragraph><literal_block xml:space="preserve">Termin
al> ptex2tex −DLATEX_HEADING=traditional mydoc</literal_block><paragraph>A se
parate titlepage can be generate by:</paragraph><literal_block xml:space="preser
ve">Terminal> ptex2tex −DLATEX_HEADING=titlepage mydoc</literal_block><paragr
aph>The <literal>ptex2tex</literal> tool makes it possible to easily switch betw
een many
different fancy formattings of computer or verbatim code in LaTeX
documents. After any <literal>!bc</literal> command in the Doconce source you ca
n
insert verbatim block styles as defined in your <literal>.ptex2tex.cfg</literal>
file, e.g., <literal>!bc cod</literal> for a code snippet, where <literal>cod</l
iteral> is set to
a certain environment in <literal>.ptex2tex.cfg</literal> (e.g., <literal>CodeIn
tended</literal>).
There are over 30 styles to choose from.</paragraph><paragraph><emphasis>Step 3.
</emphasis> Compile <literal>mydoc.tex</literal>
and create the PDF file:</paragraph><literal_block xml:space="preserve">Terminal
> latex mydoc
Terminal> latex mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibitem mydoc   # if bibliography
Terminal> latex mydoc
Terminal> dvipdf mydoc</literal_block><paragraph>If one wishes to use the <li
teral>Minted_Python</literal>, <literal>Minted_Cpp</literal>, etc., environments
in <literal>ptex2tex</literal> for typesetting code, the <literal>minted</litera
l> LaTeX package is needed.
This package is included by running <literal>doconce format</literal> with the
<literal>−DMINTED</literal> option:</paragraph><literal_block xml:space="preserv
e">Terminal> ptex2tex −DMINTED mydoc</literal_block><paragraph>In this case,
<literal>latex</literal> must be run with the
<literal>−shell−escape</literal> option:</paragraph><literal_block xml:space="pr
eserve">Terminal> latex −shell−escape mydoc
Terminal> latex −shell−escape mydoc
Terminal> makeindex mydoc   # if index
Terminal> bibitem mydoc   # if bibliography
Terminal> latex −shell−escape mydoc
Terminal> dvipdf mydoc</literal_block><paragraph>The <literal>−shell−escape</
literal> option is required because the <literal>minted.sty</literal> style
file runs the <literal>pygments</literal> program to format code, and this progr
am
cannot be run from <literal>latex</literal> without the <literal>−shell−escape</
literal> option.</paragraph></section><section ids="plain−ascii−text" names="pla
in\ ascii\ text"><title>Plain ASCII Text</title><paragraph>We can go from Doconc
e "back to" plain untagged text suitable for viewing
in terminal windows, inclusion in email text, or for insertion in
computer source code:</paragraph><literal_block xml:space="preserve">Terminal&gt

tutorial.xml

```
; doconce format plain mydoc.do.txt  # results in mydoc.txt</literal_block></sec
tion><section dupnames="restructuredtext" ids="id3"><title>reStructuredText</tit
le><paragraph>Going from Doconce to reStructuredText gives a lot of possibilitie
s to
go to other formats. First we filter the Doconce text to a
reStructuredText file <literal>mydoc.rst</literal>:</paragraph><literal_block xm
l:space="preserve">Terminal&gt; doconce format rst mydoc.do.txt</literal_block><
paragraph>We may now produce various other formats:</paragraph><literal_block xm
l:space="preserve">Terminal&gt; rst2html.py  mydoc.rst &gt; mydoc.html # html
Terminal&gt; rst2latex.py mydoc.rst &gt; mydoc.tex  # latex
Terminal&gt; rst2xml.py   mydoc.rst &gt; mydoc.xml  # XML
Terminal&gt; rst2odt.py   mydoc.rst &gt; mydoc.odt  # OpenOffice</literal_block>
<paragraph>The OpenOffice file <literal>mydoc.odt</literal> can be loaded into O
penOffice and
saved in, among other things, the RTF format or the Microsoft Word format.
That is, one can easily go from Doconce to Microsoft Word.</paragraph></section>
<section dupnames="sphinx" ids="id4"><title>Sphinx</title><paragraph>Sphinx docu
ments demand quite some steps in their creation. We have automated
most of the steps through the <literal>doconce sphinx_dir</literal> command:</pa
ragraph><literal_block xml:space="preserve">Terminal&gt; doconce sphinx_dir auth
or=&quot;authors' names&quot; \
          title=&quot;some title&quot; version=1.0 dirname=sphinxdir \
          theme=mytheme file1 file2 file3 ...</literal_block><paragraph>The keyw
ords <literal>author</literal>, <literal>title</literal>, and <literal>version</
literal> are used in the headings
of the Sphinx document. By default, <literal>version</literal> is 1.0 and the sc
ript
will try to deduce authors and title from the doconce files <literal>file1</lite
ral>,
<literal>file2</literal>, etc. that together represent the whole document. Note
that
none of the individual Doconce files <literal>file1</literal>, <literal>file2</l
iteral>, etc. should
include the rest as their union makes up the whole document.
The default value of <literal>dirname</literal> is <literal>sphinx-rootdir</lite
ral>. The <literal>theme</literal>
keyword is used to set the theme for design of HTML output from
Sphinx (the default theme is <literal>'default'</literal>).</paragraph><paragrap
h>With a single-file document in <literal>mydoc.do.txt</literal> one often just
runs:</paragraph><literal_block xml:space="preserve">Terminal&gt; doconce sphinx
_dir mydoc</literal_block><paragraph>and then an appropriate Sphinx directory <l
iteral>sphinx-rootdir</literal> is made with
relevant files.</paragraph><paragraph>The <literal>doconce sphinx_dir</literal>
command generates a script
<literal>automake-sphinx.sh</literal> for compiling the Sphinx document into an
HTML
document.  This script copies directories named <literal>figs</literal> or <lite
ral>figures</literal>
over to the Sphinx directory so that figures are accessible in the
Sphinx compilation.  If figures or movies are located in other
directories, <literal>automake-sphinx.sh</literal> must be edited accordingly. O
ne
can either run <literal>automake-sphinx.sh</literal> or perform the steps in the
script manually.</paragraph><paragraph>Doconce comes with a collection of HTML t
hemes for Sphinx documents.
These are packed out in the Sphinx directory, the <literal>conf.py</literal>
configuration file for Sphinx is edited accordingly, and a script
<literal>make-themes.sh</literal> can make HTML documents with one or more theme
s.
```

tutorial.xml

```
For example,
to realize the themes <literal>fenics</literal> and <literal>pyramid</literal>,
one writes:</paragraph><literal_block xml:space="preserve">Terminal&gt; ./make-t
hemes.sh fenics pyramid</literal_block><paragraph>The resulting directories with
 HTML documents are <literal>_build/html_fenics</literal>
and <literal>_build/html_pyramid</literal>, respectively. Without arguments,
<literal>make-themes.sh</literal> makes all available themes (!).</paragraph><pa
ragraph>If it is not desirable to use the autogenerated scripts explained
above, here are the complete manual procedure of generating a
Sphinx document from a file <literal>mydoc.do.txt</literal>.</paragraph><paragra
ph><emphasis>Step 1.</emphasis> Translate Doconce into the Sphinx dialect of
the reStructuredText format:</paragraph><literal_block xml:space="preserve">Term
inal&gt; doconce format sphinx mydoc</literal_block><paragraph><emphasis>Step 2.
</emphasis> Create a Sphinx root directory with a <literal>conf.py</literal> fil
e,
either manually or by using the interactive <literal>sphinx-quickstart</literal>
program. Here is a scripted version of the steps with the latter:</paragraph><li
teral_block xml:space="preserve">mkdir sphinx-rootdir
sphinx-quickstart &lt;&lt;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</literal_block><paragraph><emphasis>Step 3.</emphasis> Copy the <literal>tut
orial.rst</literal> file to the Sphinx root directory:</paragraph><literal_block
 xml:space="preserve">Terminal&gt; cp mydoc.rst sphinx-rootdir</literal_block><p
aragraph>If you have figures in your document, the relative paths to those will
be invalid when you work with <literal>mydoc.rst</literal> in the <literal>sphin
x-rootdir</literal>
directory. Either edit <literal>mydoc.rst</literal> so that figure file paths ar
e correct,
or simply copy your figure directories to <literal>sphinx-rootdir</literal>.</pa
ragraph><paragraph><emphasis>Step 4.</emphasis> Edit the generated <literal>inde
x.rst</literal> file so that <literal>mydoc.rst</literal>
is included, i.e., add <literal>mydoc</literal> to the <literal>toctree</literal
> section so that it becomes:</paragraph><literal_block xml:space="preserve">..
toctree::
   :maxdepth: 2

   mydoc</literal_block><paragraph>(The spaces before <literal>mydoc</literal> a
re important!)</paragraph><paragraph><emphasis>Step 5.</emphasis> Generate, for
instance, an HTML version of the Sphinx source:</paragraph><literal_block xml:sp
```

tutorial.xml

```
ace="preserve">make clean   # remove old versions
make html</literal_block><paragraph><emphasis>Step 6.</emphasis> View the result
:</paragraph><literal_block xml:space="preserve">Terminal&gt; firefox _build/htm
l/index.html</literal_block><paragraph>Note that verbatim code blocks can be typ
eset in a variety of ways
depending the argument that follows <literal>!bc</literal>: <literal>cod</litera
l> gives Python
(<literal>code-block:: python</literal> in Sphinx syntax) and <literal>cppcod</l
iteral> gives C++, but
all such arguments can be customized both for Sphinx and LaTeX output.</paragrap
h></section><section ids="google-code-wiki" names="google\ code\ wiki"><title>Go
ogle Code Wiki</title><paragraph>There are several different wiki dialects, but
 Doconce only support the
one used by <reference name="Google Code" refuri="http://code.google.com/p/suppo
rt/wiki/WikiSyntax">Google Code</reference><target ids="google-code" names="goog
le\ code" refuri="http://code.google.com/p/support/wiki/WikiSyntax"/>.
The transformation to this format, called <literal>gwiki</literal> to explicitly
 mark
it as the Google Code dialect, is done by:</paragraph><literal_block xml:space="
preserve">Terminal&gt; doconce format gwiki mydoc.do.txt</literal_block><paragra
ph>You can then open a new wiki page for your Google Code project, copy
the <literal>mydoc.gwiki</literal> output file from <literal>doconce format</lit
eral> and paste the
file contents into the wiki page. Press <strong>Preview</strong> or <strong>Save
 Page</strong> to
see the formatted result.</paragraph><paragraph>When the Doconce file contains f
igures, each figure filename must 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).</paragraph></section><section ids="tweaking-th
e-doconce-output" names="tweaking\ the\ doconce\ output"><title>Tweaking the Doc
once Output</title><paragraph>Occasionally, one would like to tweak the output i
n a certain format
from Doconce. One example is figure filenames when transforming
Doconce to reStructuredText. Since Doconce does not know if the
<literal>.rst</literal> file is going to be filtered to LaTeX or HTML, it cannot
 know
if <literal>.eps</literal> or <literal>.png</literal> is the most appropriate im
age 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 <literal>make.sh</literal> files in <literal>docs/manual</literal
> and <literal>docs/tutorial</literal>
constitute comprehensive examples on how such scripts can be made.</paragraph></
section><section ids="demos" names="demos"><title>Demos</title><paragraph>The cu
rrent text is generated from a Doconce format stored in the file:</paragraph><li
teral_block xml:space="preserve">docs/tutorial/tutorial.do.txt</literal_block><p
aragraph>The file <literal>make.sh</literal> in the <literal>tutorial</literal>
directory of the
Doconce source code contains a demo of how to produce a variety of
formats.  The source of this tutorial, <literal>tutorial.do.txt</literal> is the
starting point.  Running <literal>make.sh</literal> and studying the various gen
erated
files and comparing them with the original <literal>tutorial.do.txt</literal> fi
le,
gives a quick introduction to how Doconce is used in a real case.
<reference name="Here" refuri="https://doconce.googlecode.com/hg/doc/demos/tutor
```

tutorial.xml

```
ial/index.html">Here</reference><target ids="here" names="here" refuri="https://
doconce.googlecode.com/hg/doc/demos/tutorial/index.html"/>
is a sample of how this tutorial looks in different formats.</paragraph><paragra
ph>There is another demo in the <literal>docs/manual</literal> directory which
translates the more comprehensive documentation, <literal>manual.do.txt</literal
>, to
various formats. The <literal>make.sh</literal> script runs a set of translation
s.</paragraph></section><section ids="dependencies" names="dependencies"><title>
Dependencies</title><system_message backrefs="id5" level="2" line="385" source="
tutorial.rst" type="WARNING"><paragraph>Duplicate explicit target name: &amp;quot;sp
hinx&amp;quot;.</paragraph></system_message><paragraph>If you make use of preprocess
or directives in the Doconce source,
either <reference name="Preprocess" refuri="http://code.google.com/p/preprocess"
>Preprocess</reference><target ids="preprocess" names="preprocess" refuri="http:
//code.google.com/p/preprocess"/> or <reference name="Mako" refuri="http://www.m
akotemplates.org">Mako</reference><target ids="mako" names="mako" refuri="http:/
/www.makotemplates.org"/> must be installed.  To make LaTeX
documents (without going through the reStructuredText format) you also
need <reference name="ptex2tex" refuri="http://code.google.com/p/ptex2tex">ptex2
tex</reference><target ids="ptex2tex" names="ptex2tex" refuri="http://code.googl
e.com/p/ptex2tex"/> and some style
files that <literal>ptex2tex</literal> potentially makes use of.  Going from
reStructuredText to formats such as XML, OpenOffice, HTML, and LaTeX
requires <reference name="docutils" refuri="http://docutils.sourceforge.net">doc
utils</reference><target ids="docutils" names="docutils" refuri="http://docutils
.sourceforge.net"/>.  Making Sphinx
documents requires of course <reference name="Sphinx" refuri="http://sphinx.poco
o.org">Sphinx</reference><target dupnames="sphinx" ids="id5" refuri="http://sphi
nx.pocoo.org"/>.
All of the mentioned potential dependencies are pure Python packages
which are easily installed.
If translation to <reference name="Pandoc" refuri="http://johnmacfarlane.net/pan
doc/">Pandoc</reference><target ids="pandoc" names="pandoc" refuri="http://johnm
acfarlane.net/pandoc/"/> is desired,
the Pandoc Haskell program must of course be installed.</paragraph></section></s
ection></document>
```