Doconce: Document Once, Include Everywhere

H. P. Langtangen
Simula Research Laboratory and University of Oslo

July 15, 2009

- 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, Word/OpenOffice, LaTeX, HTML, reStructuredText, XML, wiki, etc. Would it be convenient to start with some very simple text-like format that easily converts to the formats listed above, and at some later stage eventually go with a particular format?
- Do you find it problematic that 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 place, and include it everywhere?

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

The Doconce Concept

Doconce is two things:

1. Doconce is a working strategy for documenting software in a single place and avoiding duplication of information. The slogan is: "Write once, include everywhere". This requires that what you write can be transformed to many different formats for a variety of documents (manuals, tutorials, books, doc strings, source code comments, etc.).
2. Doconce is a simple and minimally tagged markup language that can be used for the above purpose. The Doconce format look 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, reStructuredText, XML, OpenOffice/Word, Epytext, PDF, XML - and even plain text (with tags removed for clearer reading).

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 asterix,
- *emphasized words* are surrounded by an asterix,
- **words in boldface** are surrounded by underscores,
- words from computer code are enclosed in back quotes and then typeset verbatim,
- blocks of computer code can easily be included, also from source files,

- blocks of LaTeX mathematics can easily be included,
- there is support oforboth LaTeX and text-like inline mathematics,
- figures with captions, URLs with links, labels and references are supported,
- comments can be inserted throughout the text,
- a preprocessor (much like the C preprocessor) is integrated so other documents (files) can be included and large portions of text can be defined in or out of the text.

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

```
===== 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 automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3

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

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

1. (for ordered) instead of the asterix:
   (a) item 1
   (b) item 2
   (c) item 3

URLs with a link word are possible, as in hpl.

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)
```

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 "document once"!).

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.

Seeing More of What Doconce Is

After the quick syntax tour above, we recommend to read the Doconce source of the current tutorial and compare it with what you see in a browser, a PDF document, in plain text, and so forth. The Doconce source is found in the folder `doc/tutorial.do.txt` in the source code tree of Doconce. The Doconce example documentation displays both the source `tutorial.do.txt` and the result of many other formats.

A more complete documentation of and motivation for Doconce appears in the file `lib/doconce/doc/doconce.do.txt` in the Doconce source code tree. The same documentation appears in the doc string of the `doconce` module.

From Doconce to Other Formats

Transformation of a Doconce document to various other formats applies the script `doconce2format`:

```
Unix/DOS> doconce2format format doconce-file
```

For example, making an HTML version of a Doconce file `mydoc.do.txt` is performed by

```
Unix/DOS> doconce2format HTML mydoc.do.txt
```

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

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

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

```
Unix/DOS> doconce2format LaTeX mydoc.do.txt
```

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

   ```
   Unix/DOS> ptex2tex mydoc
   ```

   or just perform a plain copy,

   ```
   Unix/DOS> cp mydoc.p.tex mydoc.tex
   ```

   The `ptex2tex` tool makes it possible to easily switch between many different fancy formattings of computer or verbatim code in LaTeX documents.

3. Compile `mydoc.tex` the usual way and create the PDF file.

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:

```
Unix/DOS> doconce2format plain mydoc.do.txt  # results in mydoc.txt
```

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`:

```
Unix/DOS> doconce2format rst mydoc.do.txt
```

We may now produce various other formats:

```
Unix/DOS> rst2html.py  mydoc.rst > mydoc.html # HTML
Unix/DOS> rst2latex.py mydoc.rst > mydoc.tex  # LaTeX
Unix/DOS> rst2xml.py   mydoc.rst > mydoc.xml  # XML
Unix/DOS> 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, if desired.

The file `make.sh` in the same directory as the `doconce.do.txt` file shows how to run `doconce2format` on the `doconce.do.txt` file to obtain documents in various formats. To go from the LaTeX format to PDF, see `latex.sh`. Running this demo (`make.sh` and `latex.sh`) and studying the various generated files and comparing them with the original `doconce.do.txt` file, gives a quick introduction to how Doconce is used in a real case.

The Doconce Documentation Strategy for User Manuals

Doconce was particularly made for writing tutorials or user manuals associated with computer codes. The text is written in Doconce format in separate files. LaTeX, HTML, XML, and other versions of the text is easily produced by the `doconce2format` script and standard tools. A plain text version is often wanted for the computer source code, this is easy to make, and then one can use `#include` statements in the computer source code to automatically get the manual or tutorial text in comments or doc strings. Below is a worked example.

Consider a Python module in a `basename.p.py` file. The `.p.py` extension identifies this as a file that has to be preprocessed by the `preprocess` program (`preprocess` is much like the standard C preprocessor, but it works for TeX/LaTeX, Bash, Python, Perl, Ruby, Java, etc.). In a doc string in `basename.p.py` we do a preprocessor include in a comment line, say

```
#    #include "doc/doc1.dst.txt
```

The file `doc/doc1.dst.txt` is a file filtered to a specific format (typically plain text or Epytext) from an original "singleton" documentation file named `doc/doc1.do.txt`. The `.dst.txt` is the extension of a file filtered ready for being included in a doc string (`d` for doc, `st` for string).

For making an Epydoc manual, the `doc/doc1.do.txt` file is filtered to `doc/doc1.epytext` and renamed to `doc/doc1.dst.txt`. Then we run the preprocessor on the `basename.p.py` file and create a real Python file `basename.py`. Finally, we run Epydoc on this file.

The next step is to produce the final pure Python source code. For this purpose we filter `doc/doc1.do.txt` to plain text format (`doc/doc1.txt`) and rename to `doc/doc1.dst.txt`. The preprocessor transforms the `basename.p.py` file to a standard Python file `basename.py`. The doc strings are now in plain text and well suited for Pydoc or reading by humans. All these steps are automated by the `insertdocstr.py` script. Here are the corresponding Unix commands:

```
# make Epydoc API manual of basename module:
cd doc
doconce2format epytext doc1.do.txt
mv doc1.epytext doc1.dst.txt
cd ..
preprocess basename.p.py > basename.py
epydoc basename

# make ordinary Python module files with doc strings:
cd doc
doconce2format plain doc1.do.txt
mv doc1.txt doc1.dst.txt
cd ..
preprocess basename.p.py > basename.py

# can automate inserting doc strings in all .p.py files:
insertdocstr.py plain .
# (runs through all .do.txt files and filters them to plain format and
# renames to .dst.txt extension, then the script runs through all
# .p.py files and runs the preprocessor, which includes the .dst.txt
# files)
```

Warning

Doconce can be viewed is a unified interface to a variety of typesetting formats. This interface is minimal in the sense that a lot of typesetting features are not supported, for example, footnotes and bibliography. For many documents the simple Doconce format is sufficient, while in other cases you need more sophisticated formats. Then you can just filter the Doconce text to a more approprite format and continue working in this format only (reStructuredText is a good alternative: it is more tagged than Doconce and cannot be filtered to plain, untagged text, but it also has a lot more typesetting and tagging features than Doconce).