Tutorial on getting started with markdown/ReStructuredText

Date: 2017-07-21
Author: **Mark Galassi**

One of the problems with markdown/ReSTructuredText is that I have not found a tutorial that shows lightweight manner to starting using it on a single small markup file, getting a single output file.

Another confusing thing is that there are two sets of tools for processing the markup: the *docutils* and the *sphinx* tools. They both parse most of the same stuff, but with very different angles and purposes.

In this tutorial I will try to get you going and see if I can clarify it and show how to get started on lightweight examples, then moving on to a larger rest document.

Motivation

(You might want to skip this section and jump to the sections where you start running some examples.)

I have mostly written my documents in LaTeX, plain text, TeXinfo and Docbook. Each of these has strengths and limitations. ReStructuredText (from here on reST) seems to occupy an important area that stands between the plain text and the richer markup in LaTeX, TeXinfo and Docbook.

In this post I will discuss the boundary between LaTeX and plain text.

LaTeX is the clear choice for writing mathematical and scientific documents, but it works poorly for computer documentation. This is because TeX and LaTeX are aimed at typsetting text, which often does not make for good "cut and paste" of the text into a command line or into a computer program.

For example, LaTeX has the `lstlisting` package which allows you to do very nice typsetting of program listings and command line examples, but it is hard to select text from the output PDF document and insert it into a program or the command line. You end up with spurious text and weird spaces.

You can use plain text for computer instructions, as is done in many simple `README` files, but as soon as the document gets bigger you will want to give it some structure.

Lightweight markup languages, such as markdown/reST (from here on I will just say reST since I use that), fill that space: they allow you to build a full book out of a document, but at the same time the HTML output is simple and close to plain text, so you can select sections and paste them into your command line or your programs.

reST is widely adopted, being the Python documentation project standard as well as the new choice for the GNU Scientific Library book. Its tools are powerful, if a bit "young". There are also some agile tools for markdown, and these tools will convert most of your reST document.

What's the difference?

There is a good article comparing the two: http://zverovich.net/2016/06/16/rst-vs-markdown.html

After reading that article you see that the clear choice is to use the full reST language, rather than markdown, if your document has ambitions of being a full book. Markdown seems to be easier for smaller documents. In any case it's nice to know that most of your reST document will be parsed nicely by a program that underestands markdown.

Tools

I will mention various packages for GNU/Linux systems that process documentation formats and convert from one to another. You might want to install them. These are

docutils

Text processing system for reST. (Ubuntu GNU/Linux: `sudo apt-get install python-docutils rst2pdf`)

sphinx

The ultiamate system for building and delivering reST documents. This is used to create full web sites and books from reST. (Ubuntu GNU/Linux: `sudo apt-get install python-sphinx`)

unoconv

Universal office converter. Converts many "wysiwyg" (what you see is what you get) formats. For example, it can take libreoffice impress (or powerpoint) documents and convert them to PDF slides or HTML web sites. (Ubuntu GNU/Linux: `sudo apt-get install unoconv`)

pandoc

Converts between many different types of documentation formats (Ubuntu GNU/Linux: `sudo apt-get install pandoc`)

Simplest example

Let us create a simple file. Put the following marked up text into a file called `rest-sample.rst`

```
============================
 A lightweight document title
============================

This is a very short document with *very* little markup in it.
```

Now let us convert *rest-sample.rst* to a few output formats.

html

```
rst2html rest-sample.rst > rest-sample.html
```

pdf

`rst2pdf rest-sample.rst` ## output goes into `rest-sample.pdf`

odt (libreoffice writer)

```
rst2odt rest-sample.rst > rest-sample.odt
```

latex (LaTeX)

```
rst2latex rest-sample.rst > rest-sample.tex
```

(you can then process rest-sample.tex with pdflatex, which might overwrite the rest-sample.pdf file which you created with rst2pdf!)

What we have just done is write a very simple markdown document and convert it to LaTeX, html, pdf and odt.

From LaTeX to markdown

The automatic conversion

As I mentioned above, I am interested in moving away from LaTeX for those documents that do not involve serious use of mathematical typesetting.

To see an example of converting a LaTeX document to reST look at the following:

```
## download a sample LaTeX document and its figure form the web
$ wget http://www.electronics.oulu.fi/latex/examples/example_1/example1.tex
$ wget http://www.electronics.oulu.fi/latex/examples/example_1/myfigure.pdf
```

```
## now typeset it with:
$ pdflatex example1.tex
## and view it by opening example1.pdf in your favorite pdf
## viewer, for example:
$ evince example1.pdf &
```

Now let us convert that document using pandoc:

```
$ pandoc -o example1.rst example1.tex
```

You now have a reST file called `example1.rst`

You could convert this new `example1.rst` file to HTML with:

```
$ rst2html example1.rst > example1.html
```

What gets missed in the automatic conversion

Now try looking at both example1.html (which comes through the path LaTeX ---pandoc---> reST ---docutils---> HTML) and example1.pdf (which comes from running `pdflatex` on the original document).

What you see is that it got much of the text, but:

- Title/author are not present. We'll have to fill them out ourselves.
- The equation looks poor in HTML. This can be fixed by using the MathJAX extension to reST.
- The figure did not get inserted. This can be fixed with a bit of hand editing.

The real problem is that some extensions to LaTeX are completely dropped. For example, LaTeX's `lstlisting` package is not supported and it *drops* all blocks of text which are in an lstlisting. This is a real problem.

A bigger project: using sphinx

We will now try to create a new reST project and set it up to be processed by sphinx, that very powerful tool that can make a whole web site out of your reST marked-up documents.

First install some of the sphinx tools with:

```
$ sudo apt-get install python-sphinx
$ sudo apt-get install fonts-mathjax libjs-mathjax
```

Then make a new diretory to work in

```
$ mkdir my-sphinx-project
$ cd my-sphinx-project
```

Then create a project in there with `sphinx-quickstart`

```
$ sphinx-quickstart
```

You can answer most of these questions with the defaults, but you might change the following:

- Separate source and build directories: y
- Project name: my-sphinx-project

- Author name(s): Your Name
- Version: 0.1
- Release: 0.1.0
- epub builder: y
- mathjax: y
- Create Windows command file: n

You now have a project! You will see that there is a file called `index.rst` which lets you include your own test. We will first build the project:

```
$ make html
```

Now load up the file `_build/html/index.html` in your web browser, and you can reload it whenever you make changes and type `make`

Now let us modify the `index.rst` file to include a file of our own:

```
Welcome to my-sphinx-project's documentation!
=============================================

.. toctree::
   :maxdepth: 2
   :caption: Contents:

   my-project.rst

Indices and tables
==================
```

and create a minimal file called `my-project.rst`

```
This is my project
==================

This is my project
```

From here on you put your text in `my-project.rst` and you can do the usual cycle of running `make` and reloading the page in your web browser to see how things are going.

References

- http://pandoc.org/demos.html
- http://zverovich.net/2016/06/16/rst-vs-markdown.html
- http://www.unexpected-vortices.com/doc-notes/markdown-and-rest-compared.html
- http://www.electronics.oulu.fi/latex/examples/example_1/