How to use rst2pdf

Author: Roberto Alsina <ralsina@netmanagers.com.ar>
Version: 0.14.1
Revision: 2030

Contents

1 Introduction

This document explains how to use rst2pdf. Here is the very short version:

```
rst2pdf.py mydocument.txt -o mydocument.pdf
```

That will, as long as mydocument.txt is a valid Restructured Text (ReST) document, produce a file called mydocument.pdf which is a PDF version of your document.

Of course, that means you just used default styles and settings. If it looks good enough for you, then you may stop reading this document, because you are done with it. If you are reading this in a PDF, it was generated using those default settings.

However, if you want to customize the output, or are just curious to see what can be done, let´s continue.

2 Command line options

`-h, --help`	Show this help message and exit
`--config=FILE`	Config file to use. Default=~/.rst2pdf/config
`-o FILE, --output=FILE`	Write the PDF to FILE
`-s STYLESHEETS, --stylesheets=STYLESHEETS`	A comma-separated list of custom stylesheets. Default=""
`--stylesheet-path=FOLDERLIST`	A colon-separated list of folders to search for stylesheets. Default=""
`-c, --compressed`	Create a compressed PDF. Default=False
`--print-stylesheet`	Print the default stylesheet and exit
`--font-folder=FOLDER`	Search this folder for fonts. (Deprecated)
`--font-path=FOLDERLIST`	A colon-separated list of folders to search for fonts. Default=""
`--baseurl=URL`	The base URL for relative URLs.
`-l LANG, --language=LANG`	Language to be used for hyphenation and docutils localization. Default=None
`--header=HEADER`	Page header if not specified in the document.
`--footer=FOOTER`	Page footer if not specified in the document.
`--smart-quotes=VALUE`	Try to convert ASCII quotes, ellipsis and dashes to the typographically correct equivalent. Default=0

The possible values are:

1. Suppress all transformations. (Do nothing.)
2. Performs default SmartyPants transformations: quotes (including backticks-style), em-dashes, and ellipses. "--" (dash dash) is used to signify an em-dash; there is no support for en-dashes.
3. Same as --smart-quotes=1, except that it uses the old-school typewriter shorthand for dashes: "--" (dash dash) for en-dashes, "---" (dash dash dash) for em-dashes.
4. Same as --smart-quotes=2, but inverts the shorthand for dashes: "--" (dash dash) for em-dashes, and "---" (dash dash dash) for en-dashes.

`--fit-literal-mode=MODE`	What to do when a literal is too wide. One of error,overflow,shrink,truncate. Default="shrink"
`--inline-links`	Shows target between parenthesis instead of active link
`--repeat-table-rows`	Repeats header row for each splitted table
`-q, --quiet`	Print less information.
`-v, --verbose`	Print debug information.
`--very-verbose`	Print even more debug information.
`--version`	Print version number and exit.
`--no-footnote-backlinks`	Disable footnote backlinks. Default: False
`--inline-footnotes`	Show footnotes inline. Default: True
`--default-dpi=NUMBER`	DPI for objects sized in pixels. Default=300
`--show-frame-boundary`	Show frame borders (only useful for debugging). Default=False

`--disable-splittables`	Don't use splittable flowables in some elements. Only try this if you can't process a document any other way.
`-b LEVEL, --break-level=LEVEL`	Maximum section level that starts in a new page. Default: 0
`--first-page-even`	Whether first page is odd (as in the screen on "facing pages"), or even (as in a book)
`--blank-first-page`	Add a blank page at the beginning of the document.
`--break-side=VALUE`	How section breaks work. Can be "even", and sections start in an even page,"odd", and sections start in odd pages, or "any" and sections start in the next page,be it even or odd. See also the -b option.
`--date-invariant`	Don't store the current date in the PDF. Useful mainly for the test suite, where we don't want the PDFs to change.
`-e EXTENSIONS`	Alias for --extension-module
`--extension-module=EXTENSIONS`	Add a helper extension module to this invocation of rst2pdf (module must end in .py and be on the python path)

3 Configuration File

Since version 0.8, rst2pdf will read (if it is available) configuration files in `/etc/rst2pdf.conf` and `~/.rst2pdf/config`.

The user's file at `~/.rst2pdf/config` will have priority over the system's at `/etc/rst2pdf.conf` [1]

Here's an example file showing some of the currently available options:

```
# This is an example config file. Modify and place in ~/.rst2pdf/config

[general]
# A comma-separated list of custom stylesheets. Example:
# stylesheets="fruity.json,a4paper.json,verasans.json"
stylesheets=""

# Create a compressed PDF
# Use true/false (lower case) or 1/0
# Example: compressed=true
compressed=false

# A colon-separated list of folders to search for fonts. Example:
# font_path="/usr/share/fonts:/usr/share/texmf-dist/fonts/"
font_path=""

# Language to be used for hyphenation support
language="en_US"

# Default page header and footer
header=None
footer=None

# What to do if a literal block is too large. Can be
# shrink/truncate/overflow
fit_mode="shrink"

# What is the maximum level of heading that starts in a new page.
# 0 means no level starts in a new page.
break_level=0
```

Included with rst2pdf is an example file with every option in it.

4 Pipe usage

If no input nor output are provided, stdin and stdout will be used respectively

You may want to use rst2pdf in a linux pipe as such:

```
cat readme.txt | rst2pdf | gzip -c > readme.pdf.gz
```

or:

```
curl http://docutils.sourceforge.net/docs/user/rst/quickstart.txt | rst2pdf > quickstart.pdf
```

If no input argument is provided, stdin will be used:

```
cat readme.txt | rst2pdf -o readme.pdf
```

If outpufile is set to dash '-', output goes to stdout:

```
rst2pdf -o - readme.txt > output.pdf
```

5 Headers and Footers

ReST supports headers and footers, using the header and footer directive:

```
.. header::

   This will be at the top of every page.
```

Often, you may want to put a page number there, or a section name.The following magic tokens will be replaced (More may be added as rst2pdf evolves):

###Page###

Replaced by the current page number.

###Title###

Replaced by the document title

###Section###

Replaced by the currect section title

###SectNum###

Replaced by the currect section number. **Important:** You must use the sectnum directive for this to work.

###Total###

Replaced by the total number of pages in the document. Keep in mind that this is the **real** number of pages, not the displayed number, so if you play with page counters this number will probably be wrong.

Headers and footers are visible by default but they can be disabled by specific Page Templates for example, cover pages. You can also set headers and footers via *command line options* or the configuration file.

If you want to do things like "put the page number on the *out* side of the page, check The oddeven directive

6 Footnotes

Currently rst2pdf doesn't support real footnotes, and converts them to endnotes. There is a real complicated technical reason for this: I can't figure out a clean way to do it right.

You can get the same behaviour as with rst2html by specifying --inline-footnotes, and then the footnotes will appear where you put them (in other words, not footnotes, but "in-the-middle-of-text-notes" or just plain notes.)

7 Images

7.1 Inline

You can insert images in the middle of your text like this:

```
This |biohazard| means you have to run.

.. |biohazard| image:: ../rst2pdf/tests/input/images/biohazard.png
```

This  means you have to run.

This only works correctly with reportlab 2.2 or later.

7.2 Supported Image Types

For raster images, rst2pdf supports anything PIL (The Python Imaging Library) supports. The exact list of supported formats varies according to your PIL version and system.

For vector image support, you need to install Dinu Gherman's svglib (http://pypi.python.org/pypi/svglib/) or Uniconvertor from http://sk1project.org version 1.1.3 or later.

It provides support for these formats:

- CorelDRAW ver.7-X3,X4 (CDR/CDT/CCX/CDRX/CMX)
- Adobe Illustrator up to 9 ver. (AI postscript based)
- Postscript (PS)
- Encapsulated Postscript (EPS)
- Computer Graphics Metafile (CGM)
- Windows Metafile (WMF)
- XFIG
- Scalable Vector Graphics (SVG)
- Skencil/Sketch/sK1 (SK and SK1)
- Acorn Draw (AFF)

Some features will not work when using these images.For example, gradients will not display, and text may cause problems.

To test suitability of your vector images for use with rst2pdf, try converting them to PDF using uniconvertor. The result should be exactly the way they will look when used in your documents.

If you can choose between raster and vectorial images, for non-photographic images, vector files are usually smaller and look better, specially when printed.

If you want to use PDF files as images, you need to install PythonMagick (http://www.imagemagick.org), which will be used to convert it to PNG and then inserted in your document. If the quality is not good enough, try something like `--default-dpi 1200`

This only works for one-page PDF files, and has several drawbacks, such as inability to copy text from the embedded image.

In the future, rst2pdf will support ReportLab's PageCatcher to properly embed PDFs. That is not implemented yet, though.

If there is any other image format you need supported, please report it as a feature request in rst2pdf's site.

7.3 Image Size

PDFs are meant to reflect paper. A PDF has a specific size in centimeters or inches.

Images usually are measured in pixels, which are meaningless in a PDF. To convert between pixels and inches or centimeters, we use a DPI (dots-per-inch) value.

For example, 300 pixels, with a 300DPI, are exactly one inch. 300 pixels at 100DPI are 3 inches.

For that reason, to achieve a nice layout of the page, it's usually a good idea to specify the size of your images in those units, or as a percentage of the available width and you can ignore all this DPI nonsense ;-)

The rst2pdf default is 300DPI, but you can change it using the --default-dpi option or the default_dpi setting in the config file.

Examples of images with specified sizes:

```
.. image:: home.png
   :width: 3in

.. image:: home.png
   :width: 80%

.. image:: home.png
   :width: 7cm
```

The valid units you can use are:

"em" "ex" "px" "in" "cm" "mm" "pt" "pc" "%" "".

- px: Pixels. If you specify the size using this unit, rst2pdf will convert it to inches using the default DPI explained above.
- No unit. If you just use a number, it will be considered as pixels. (**IMPORTANT:** this used to default to points. It was changed to be more compatible with rst2html)
- em: This is the same as your base style's font size. By default: 10 points.
- ex: rst2pdf will use the same broken definition as IE: em/2. In truth this should be the height of the lower-case x character in your base style.
- in: Inches (1 inch = 2.54 cm).
- cm: centimeters (1cm = 0.39 inches)
- mm: millimeters (10mm = 1cm)
- pt: 1/72 inch
- pc: 1/6 inch
- %: percentage of available width in the frame. Setting a percentage as a height does **not** work and probably never will.

If you don't specify a size at all, rst2pdf will do its best to figure out what it should do:

Since there is no specified size, rst2pdf will try to convert the image's pixel size to inches using the DPI information available in the image itself. You can set that value using most image editors. For example,

using Gimp, it's in the Image -> Print Size menu.

So, if your image is 6000 pixels wide, and is set to 1200DPI, it will be 5 inches wide.

If your image doesn't have a DPI property set, and doesn't have it's desired size specified, rst2pdf will arbitrarily decide it should use 300DPI (or whatever you choose with the --default-dpi option).

As of 0.10.1, images taller than the page will not work (rst2pdf will fail to run), and images wider than the page will be cropped.

8 Styles

You can style paragraphs with a style using the class directive:

```
.. class:: special

This paragraph is special.

This one is not.
```

Or inline styles using custom interpreted roles:

```
.. role:: redtext

I like color :redtext:`red`.
```

For more information about this, please check the ReST docs.

The only special thing about using rst2pdf here is the syntax of the stylesheet.

You can make rst2pdf print the default stylesheet:

```
rst2pdf --print-stylesheet
```

If you want to add styles, just create a stylesheet, (or take the standard stylesheet and modify it) and pass it with the -s option:

```
rst2pdf mydoc.txt -s mystyles.txt
```

Those styles will always be searched in these places, in order:

- What you specify using --stylesheet_path
- The option stylesheet_path in the config file
- The current folder
- ~/.rst2pdf/styles
- The styles folder within rst2pdf's installation folder.

You can use multiple -s options, or pass more than one stylesheet separated with commas. They are processed in the order you give them so the *last* one has priority.

8.1 Included StyleSheets

To make some of the more common adjustments easier, rst2pdf includes a collection of stylesheets you can use:

Font styles

These stylesheets modfy your font settings.

- `serif` uses the PDF serif font (Times) instead of the default Sans Serif (Arial)
- `freetype-sans` uses your system's default TrueType Sans Serif font
- `freetype-serif` uses your system's default TrueType Serif font
- `twelvepoint` makes the base font 12pt (default is 10pt)
- `tenpoint` makes the base font 10pt
- `eightpoint` makes the base font 8pt
- `kerning` switches to document to DejaVu Sans font and enables kerning.

Page layout styles

These stylesheets modify your page layout.

- `twocolumn` uses the twoColumn layout as the initial page layout.
- `double-sided` adds a gutter margin (margin at the "in side" of the pages)

Page size styles

Stylesheets that change the paper size.
The usual standard paper sizes are supported:

- A0
- A1
- A2
- A3
- A4 (default)
- A5
- A6
- B0
- B1
- B2
- B3
- B4
- B5
- B6
- Letter
- Legal
- 11x17

The name of the stylesheet is lowercase.

Code block styles

See Syntax Highlighting

So, if you want to have a two-column, legal size, serif document with code in murphy style:

```
rst2pdf mydoc.txt -s twocolumn,serif,murphy,legal
```

8.2 StyleSheet Syntax

It´s a JSON file with several elements in it.

8.3 Font Alias

This is the fontsAlias element. By default, it uses some of the standard PDF fonts:

```
"fontsAlias" : {
  "stdFont": "Helvetica",
  "stdBold": "Helvetica-Bold",
  "stdItalic": "Helvetica-Oblique",
  "stdBoldItalic": "Helvetica-BoldOblique",
  "stdMono": "Courier"
},
```

This defines the fonts used in the styles. You can use, for example, Helvetica directly in a style, but if later you want to use another font all through your document, you will haveto change it in each style. So, I suggest you use aliases.

The standard PDF fonts are these:

> Times_Roman Times-Bold Times-Italic Times-Bold-Italic Helvetica Helvetica_Bold Helvetica-Oblique Helvetica-Bold-Oblique Courier Courier-Bold Courier-Oblique Courier-Bold-Oblique Symbol Zapf-Dingbats

8.4 Style Definition

Then you have a 'styles' which is a list of [stylename, styleproperties]. For example:

```
["normal" , {
  "parent": "base"
}],
```

This means that the style called "normal" inherits style "base". So, each property not defined in the normal style will be taken from the base style.

I suggest you do not remove any style from the default stylesheet. Add or modify at will, though.

If your document requires a style that is not defined in your stylesheet, it will print a warning and use bodytext instead.

Also, the order of the styles is important: if styleA is the parent of styleB, styleA should be earlier in the stylesheet.

These are all the possible attributes for a style and their default values. Some of them, like alignment, apply only when used to paragraphs, and not on inline styles:

```
"fontName":"Helvetica",
"fontSize":10,
"leading":12,
"leftIndent":0,
```

```
"rightIndent":0,
"firstLineIndent":0,
"alignment":"left",
"spaceBefore":0,
"spaceAfter":0,
"bulletFontName":"Helvetica",
"bulletFontSize":10,
"bulletText": "\u2022",
"bulletIndent":0,
"textColor": black,
"backColor":None,
"wordWrap":None,
"borderWidth": 0,
"borderPadding": 0,
"borderColor": None,
"borderRadius": None,
"allowWidows": 1,
"allowOrphans": 0
```

The following are the only attributes that work on styles when used for interpreted roles (inline styles):

- fontName
- fontSize
- textColor
- backColor (if your reportlab is version 2.3 or newer)

8.5 Font Embedding

There are thousands of excelent free True Type and Type 1 fonts available on the web, and you can use many of them in your documents by declaring them in your stylesheet.

8.5.1 The Easy Way

Just use the font name in your style. For example, you can define this:

```
["normal" , {
  "fontName" : "fonty"
}]
```

And then it *may* work.

What would need to happen for this to work?

8.5.1.1 Fonty is a True Type font:

1. **You need to have it installed in your system, and have the fc-match**
   utility available (it's part of fontconfig). You can test if it is so by running this command:

   ```
   $ fc-match fonty
   fonty.ttf: "Fonty" "Normal"
   ```

   If you are in Windows, I need your help ;-) or you can use The Harder Way (True Type)

2. **The folder where fonty.ttf is located needs to be in your font path. You can set it**
   using the --font-path option. For example:

   ```
   rst2pdf mydoc.txt -s mystyle.style --font-path /usr/share/fonts
   ```

   You don't need to put the *exact* folder, just something that is above it. In my own case, fonty is in /usr/share/fonts/TTF

Whenever a font is embedded, you can refer to it in a style by its name, and to its variants by the aliases Name-Oblique, Name-Bold, Name-BoldOblique.

8.5.1.2 Fonty is a Type 1 font:

You need it installed, and the folders where its font metric (.afm) and binary (.pfb) files are located need to be in your font fath.

For example, the "URW Palladio L" font that came with my installation of TeX consists of the following files:

```
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplb8a.pfb
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplbi8a.pfb
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplr8a.pfb
/usr/share/texmf-dist/fonts/type1/urw/palatino/uplri8a.pfb
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplb8a.afm
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplbi8a.afm
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplr8a.afm
/usr/share/texmf-dist/fonts/afm/urw/palatino/uplri8a.afm
```

So, I can use it if I put `/usr/share/texmf-dist/fonts` in my font path:

```
rst2pdf mydoc.txt -s mystyle.style --font-path /usr/share/texmf-dist/fonts
```

And putting this in my stylesheet, for example:

```
[ "title", { "fontName" : "URWPalladioL-Bold" } ]
```

There are some standard aliases defined so you can use other names:

```
'ITC Bookman'            : 'URW Bookman L',
'ITC Avant Garde Gothic' : 'URW Gothic L',
'Palatino'               : 'URW Palladio L',
'New Century Schoolbook' : 'Century Schoolbook L',
'ITC Zapf Chancery'      : 'URW Chancery L'
```

So, for example, you can use `Palatino` or `New Century SchoolBook-Oblique` And it will mean `URWPalladioL` or `CenturySchL-Ital`, respectively.

Whenever a font is embedded, you can refer to it in a style by its name, and to its variants by the aliases Name-Oblique, Name-Bold, Name-BoldOblique.

8.5.2 The Harder Way (True Type)

The stylesheet has an element is "embeddedFonts" that handles embedding True Type fonts in your PDF.

Usually, it's empty, because with the default styles you are not using any font beyond the standard PDF fonts:

```
"embeddedFonts" : [ ],
```

You can put there the name of the font, and rst2pdf will try to embed it as described above. Example:

```
"embeddedFonts" : [ "Tuffy" ],
```

Or you can be explicit and tell rst2pdf the files that contain each variant of the font.

Suppose you want to use the nice public domain Tuffy font, then you need to give the filenames of all variants:

```
"embeddedFonts" : [ ["Tuffy.ttf","Tuffy_Bold.ttf","Tuffy_Italic.ttf","Tuffy_Bold_Italic.ttf"] ],
```

This will provide your styles with fonts called "Tuffy" "Tuffy_Bold" and so on. They will be available with the names based on the filenames (Tuffy_Bold) and also by standard aliases similar to those of the standard PDF fonts (Tuffy-Bold/Tuffy-Oblique/Tuffy-BoldOblique).

Now, if you use *italics* in a paragraph whose style uses the Tuffy font, it will use Tuffy_Italic. That's why it's better if you use fonts that provide the four variants, and you should put them in **that** order. If your font lacks a variant, use the "normal" variant instead.

For example, if you only had Tuffy.ttf:

```
"embeddedFonts" : [ ["Tuffy.ttf","Tuffy.ttf","Tuffy.ttf","Tuffy.ttf"] ],
```

However, that means that italics and bold in styles using Tuffy will not work correctly (they will display as regular text).

If you want to use this as the base font for your document, you should change the fontsAlias section accordingly. For example:

```
"fontsAlias" : {
  "stdFont": "Tuffy",
  "stdBold": "Tuffy_Bold",
  "stdItalic": "Tuffy_Italic",
  "stdBoldItalic": "Tuffy_Bold_Italic",
  "stdMono": "Courier"
},
```

If, on the other hand, you only want a specific style to use the Tuffy font, don't change the fontAlias, and set the fontName properties for that style. For example:

```
["heading1" , {
  "parent": "normal",
  "fontName": "Tuffy_Bold",
  "fontSize": 18,
  "keepWithNext": true,
  "spaceAfter": 6
}],
```

By default, rst2pdf will search for the fonts in its fonts folder and in the current folder. You can make it search another folder by passing the --font-folder option, or you can use absolute paths in your stylesheet.

8.5.3 The Harder Way (Type1)

To be written (and implemented and tested)

8.6 Page Size and Margins

In your stylesheet, the pageSetup element controls your page layout.

Here's the default stylesheet's:

```
"pageSetup" : {
  "size": "A4",
  "width": null,
  "height": null,
  "margin-top": "2cm",
  "margin-bottom": "2cm",
  "margin-left": "2cm",
  "margin-right": "2cm",
  "spacing-header": "5mm",
  "spacing-footer": "5mm",
  "margin-gutter": "0cm"
},
```

Size is one of the standard paper sizes, like A4 or LETTER.

Here's a list: A0, A1, A2, A3, A4, A5, A6, B0, B1, B2, B3, B4, B5, B6, LETTER, LEGAL, ELEVENSEVENTEEN.

If you want a non-standard size, set size to null and use width and height.

When specifying width, height or margins, you need to use units, like inch (inches) or cm (centimeters).

When both width/height and size are specified, size will be used, and width/height ignored.

All margins should be self-explanatory, except for margin-gutter. That's the margin in the center of a two-page spread.

This value is added to the left margin of odd pages and the right margin of even pages, adding (or removing, if it's negative) space "in the middle" of opposingpages.

If you intend to bound a printed copy, you may need extra space there. OTOH, if you will display it on-screen on a two-page format (common in many PDF readers, nice for ebooks), a negative value may be pleasant.

8.7 Advanced: table styles

This is new in 0.12.

These are a few extra options in styles that are only used when the style is applied to a table. This happens in two cases:

1. You are using the class directive on a table:

```
.. class:: thick

+-------+---------+
|   A   |   B     |
+-----------------+
```

2. It's a style that automatically applies to something that is *drawn* using a table. Currently these include:

    - Footnotes / endnotes (endnote style)
    - Lists (item_list, bullet_list option_list and field_list styles)

The options are as follows:

Commands

For a full reference of these, please check the Reportlab User Guide specifically the TableStyle Commands section (section 7.4 in the manual for version 2.3)
Here, however, is a list of the possible commands:

```
BOX (or OUTLINE)
FONT
FONTNAME (or FACE)
FONTSIZE (or SIZE)
GRID
INNERGRID
LEADING
LINEBELOW
LINEABOVE
LINEBEFORE
LINEAFTER
TEXTCOLOR
ALIGNMENT (or ALIGN)
LEFTPADDING
RIGHTPADDING
BOTTOMPADDING
TOPPADDING
BACKGROUND
ROWBACKGROUNDS
COLBACKGROUNDS
VALIGN
```

Each takes as argument a couple of coordinates, where (0,0) is top-left, and (-1,-1) is bottom-right, and 0 or more extra arguments.
For example, INNERGRID takes a linewidth and a color:

```
[ "INNERGRID", [ 0, 0 ], [ -1, -1 ], 0.25, "black" ],
```

That would mean "draw all lines inside the table with .25pt black"

colWidths

A list of the column widths you want, in the unit you prefer (default unit is pt).
Example:

```
"colWidths": ["3cm",null]
```

If your colWidths has fewer values than columns in your table, the rest are autocalculated. A column width of null means "guess".
If you don't specify column widths, the table will try to look proportional to the restructured text source.

8.8 Multiple Stylesheets

When you use a custom stylesheet, you don't need to define *everything* in it. Whatever you don't define will be taken from the default stylesheet. For example, if you only want to change page size, default font and font size, this would be enough:

```
{
    "pageSetup" : {
        "size": "A5",
    },
    "fontsAlias" : {
        "stdFont": "Times-Roman",
    },
    "styles" : [
        ["normal" , {
        "fontSize": 14
        }]
    ]
}
```

Note

The `command` option used for table styles is not kept across stylesheets. For example, the default stylesheet defines endnote with this command list:

```
"commands": [ [ "VALIGN", [ 0, 0 ], [ -1, -1 ], "TOP" ] ]
```

If you redefine endnote in another stylesheet and use this to create a vertical line between the endnote's columns:

```
"commands": [ [ "LINEAFTER", [ 0, 0 ], [ 1, -1 ], .25, "black" ] ]
```

Then the footnotes will **not** have VALIGN TOP!

To do that, you **MUST** use all commands in your stylesheet:

```
"commands": [
    [ "VALIGN", [ 0, 0 ], [ -1, -1 ], "TOP" ],
    [ "LINEAFTER", [ 0, 0 ], [ 1, -1 ], .25, "black" ]
]
```

8.9 Styling Your Document

Which styles you need to modify to achieve your desired result is not obvious. In this section, you will see some hints and pointers to that effect.

8.9.1 The Base Styles

There are three styles which have great effect, they are `base`, `normal` and `bodytext`.

Here's an example, the twelvepoint stylesheet:

```
{"styles": [["base", {"fontSize": 12}]]}
```

Since all other styles inherit `base`, changing the fontSize changes the fontSize for everything in your document.

The `normal` style is meant for most elements, so usually it's the same as changing `base`.

The `bodytext` style is for elements that form paragraphs. So, for example, you can set your document to be left-aligned like this:

```
{"styles": [["bodytext", {"alignment": "left"}]]}
```

There are elements, however, that don't inherit from bodytext, for example headings and the styles used in the table of contents. Those are elements that are not real paragraphs, so they should not follow the indentation and spacing you use for your document's main content.

The `heading` style is inherited by all sorts of titles: section titles, topic titles, admonition titles, etc.

8.9.2 Lists

Styling lists is mostly a matter of spacing and indentation.

The space before and after a list is taken from the `item_list` and `bullet_list` styles:

```
["item_list", {
  "parent": "bodytext",
  "spaceBefore": 0,
  "commands": [
        [ "VALIGN", [ 0, 0 ], [ -1, -1 ], "TOP" ],
        [ "RIGHTPADDING", [ 0, 0 ], [ 1, -1 ], 0 ]
    ],
    "colWidths": ["20pt",null]
}]

["bullet_list", {
  "parent": "bodytext",
  "spaceBefore": 0,
  "commands": [
        [ "VALIGN", [ 0, 0 ], [ -1, -1 ], "TOP" ],
        [ "RIGHTPADDING", [ 0, 0 ], [ 1, -1 ], 0 ]
    ],
    "colWidths": ["20",null]
}],
```

Yes, these are table styles, because they are implemented as tables. The RIGHTPADDING command and the colWidths option can be used to adjust the position of the bullet/item number.

To control the separation between items, you use the item_list_item and `bullet_list_item` styles' spaceBefore and spaceAfter options, for example:

```
["bullet_list_item" , {
  "parent": "bodytext",
  "spaceBefore": 20
}]
```

Remember that this is only used **between items** and not before the first or after the last items.

9 Syntax Highlighting

Rst2pdf adds a non-standard directive, called code-block, which produces syntax highlighted for many languages using Pygments.

For example, if you want to include a python fragment:

```
.. code-block:: python

    def myFun(x,y):
        print x+y
```

```
def myFun(x,y):
    print x+y
```

Notice that you need to declare the language of the fragment. Here's a list of the currently supported.

You can use the `linenos` option to display line numbers:

```
1 def myFun(x,y):
2     print x+y
```

Rst2pdf includes several stylesheets for highlighting code:

- autumn
- borland
- bw
- colorful
- emacs
- friendly
- fruity
- manni
- murphy
- native
- pastie
- perldoc
- trac
- vs

You can use any of them instead of the default by adding, for example, a `-s murphy` to the command line.

If you already are using a custom stylesheet, use both:

```
rst2pdf mydoc.rst -o mydoc.pdf -s mystyle.json,murphy
```

The default is the same as "emacs".

There is an online demo of pygments showing these styles:

http://pygments.org/demo/1817/

The overal look of a code box is controlled by the "code" style or by a class you apply to it using the `.. class::` directive. Aditionally, if you want to change some properties when using different languages, you can define styles with the name of the language. For example, a `python` style will be applied to code blocks created with `.. code-block:: python`.

The look of the line numbers is controlled by the "linenumbers" style.

As rst2pdf is written in python let's see some examples and variations around python.

Python in console

```
>>> my_string="python is great"
>>> my_string.find('great')
10
>>> my_string.startswith('py')
True
```

Python traceback

```
Traceback (most recent call last):
    File "error.py", line 9, in ?
    main()
    File "error.py", line 6, in main
    print call_error()
    File "error.py", line 2, in call_error
    r = 1/0
ZeroDivisionError: integer division or modulo by zero
Exit 1
```

9.1 File inclusion

Also, you can use the code-block directive with an external file, using the :include: option:

```
.. code-block:: python
   :include: setup.py
```

This will give a warning if setup.py doesn't exist or can't be opened.

9.1.1 Include with Boundaries

you can add selectors to limit the inclusion to a portion of the file. the options are:

start-at:	string will include file beginning at the first occurence of string, string **included**
start-after:	string will include file beginning at the first occurence of string, string **excluded**
end-before:	string will include file up to the first occurence of string, string **excluded**
end-at:	string will include file up to the first occurence of string, string **included**

Let's display a class from rst2pdf:

```
.. code-block:: python
   :include: ../rst2pdf/flowables.py
   :start-at: class Separation(Flowable):
   :end-before: class Reference(Flowable):
```

this command gives

```python
class Separation(Flowable):
    """A simple <hr>-like flowable"""

    def wrap(self, w, h):
        self.w = w
        return w, 1*cm

    def draw(self):
        self.canv.line(0, 0.5*cm, self.w, 0.5*cm)
```

10 Raw Directive

Rst2pdf has a very limited mechanism to pass commands to reportlab, the PDF generation library. You can use the raw directive to insert pagebreaks and spacers (other reportlab flowables may be added if there's interest), and set page transitions.

The syntax is shell-like, here's an example:

```
One page

.. raw:: pdf

    PageBreak

Another page. Now some space:

.. raw:: pdf

    Spacer 0,200
    Spacer 0 200

And another paragraph.
```

The unit used by the spacer by default is points, and using a space or a comma is the same thing in all cases.

10.1 Page Counters

In some documents, you may not want your page counter to start in the first page.

For example, if the first pages are a coverpage and a table of contents, you want page 1 to be where your first section starts.

To do that, you have to use the SetPageCounter command.

Here is a syntax example:

```
.. raw:: pdf

    SetPageCounter 0 lowerroman
```

This sets the counter to 0, and makes it display in lower roman characters (i, ii, iii, etc) which is a style often used for the pages before the document proper (for example, TOCs and abstracts).

It can take zero or two arguments.

SetPageCounter

When used with no arguments, it sets the counter to 0, and the style to arabic numerals.

SetPageCounter number style

When used with two arguments, the first argument must be a number, it sets the page counter to that number.
The second number is a style of counter. Valid values are:

- lowerroman: i, ii, iii, iv, v ...
- roman: I, II, III, IV, V ...
- arabic: 1, 2, 3, 4, 5 ...
- loweralpha: a, b, c, d, e ... [Don't use for numbers above 26]
- alpha: A, B, C, D, E ... [Don't use for numbers above 26]

> ***Note***
>
> Page counter changes take effect on the **current** page.

10.2 Page Breaks

There are three kinds of page breaks:

PageBreak

Break to the next page

EvenPageBreak

Break to the next **even** numbered page

OddPageBreak

Break to the next **odd** numbered page

Each of them can take an additional number which is the name of the next page template.

For example:

```
PageBreak twoColumn
```

10.3 Page Transitions

Page transitions are effects used when you change pages in *Presentation* or *Full Screen* mode (depends on the viewer). You can use it when creating a presentation using PDF files.

The syntax is this:

```
.. raw:: pdf

   Transition effect duration [optional arguments]
```

The optional arguments are:

direction:

Can be 0,90,180 or 270 (top,right,bottom,left)

dimension:

Can be H or V

motion:

Can be I or O (Inside or Outside)

The effects with their arguments are:

- Split duration direction motion
- Blinds duration dimension
- Box duration motion
- Wipe duration direction
- Dissolve duration
- Glitter duration direction

For example:

```
.. raw:: pdf

   Transition Glitter 3 90
```

Uses the Glitter effect, for 3 seconds, at direction 90 degress (from the right?)

Keep in mind that Transition sets the transition *from this page to the next* so the natural thing is to use it before a PageBreak:

```
.. raw:: pdf

   Transition Dissolve 1
   PageBreak
```

11 The oddeven directive

This is a nonstandard directive, which means it will only work with rst2pdf, and not with rst2html or any other docutils tool.

The contents of oddeven should consist of **exactly** two things (in this case, two paragraphs). The first will be used on odd pages, and the second one on even pages.

If you want to use more complex content, you should wrap it with containers, like in this example:

```
.. oddeven::

    .. container::

        This will appear on odd pages.

        Both paragraphs in the container are for odd pages.

    This will appear on even pages. It's a single paragraph, so no need for containers.
```

This directive has several limitations.

- I intentionally have disabled splitting into pages for this, because I have no idea how that could make sense. That means that if its content is larger than a frame, you **will** make rst2pdf barf with one of those ugly errors.
- It will reserve the space of the larger of the two sets of contents. So if one is small and the other large, it **will** look wrong. I may be able to fix this though.
- If you try to generate HTML (or anything other than a PDF via rst2pdf) from a file containing this, it will not do what you want.

12 Mathematics

If you have Matplotlib installed, rst2pdf supports a math role and a math directive. You can use them to insert formulae and mathematical notation in your documents using a subset of LaTeX syntax, but doesn't require you have LaTeX installed.

For example, here's how you use the math directive:

```
.. math::

   \frac{2 \pm \sqrt{7}}{3}
```

And here's the result:

$$\frac{2 \pm \sqrt{7}}{3}$$

If you want to insert mathematical notation in your text like this: π that is the job of the math *role*:

```
This is :math:`\pi`
```

Produces: This is π

Currently, the math role is slightly buggy, and in some cases will produce misaligned and generally broken output. Also, while the math directive embeds fonts and draws your formula as text, the math role embeds an image. That means:

- You can't copy the text of inline math
- Inline math will look worse when printed, or make your file larger.

So, use it only in emergencies ;-)

You can also use an inline substitution of the math directive for things you use often, which is the same as using the math role:

```
This is the square of x: |xsq|

.. |xsq| math:: x^2
```

This is the square of x: x^2

You don't need to worry about fonts, the correct math fonts will be used and embedded in your PDF automatically (they are included with matplotlib).

For an introduction to LaTeX syntax, see the "Typesetting Mathematical Formulae" chapter in "The Not So Short Introduction to LaTeX 2e":

http://www.tex.ac.uk/tex-archive/info/lshort/english/lshort.pdf

Basically, the inline form `$a^2$` is similar to the math role, and the display form is similar to the math directive.

Rst2pdf doesn't support numbering equations yet.

13 Hyphenation

If you want good looking documents, you want to enable hyphenation.

To do it, you need to install Wordaxe [2].

If after installing it you get the letter "s" or a black square instead of a hyphen, that means you need to replace the rl_codecs.py file from reportlab with the one from wordaxe.

For more information, see this issue in rst2pdf's bug tracker.

Also, you may need to set hyphenation to true in one or more styles, and the language for hyphenation via the command line or paragraph styles.

For english, this should be enough:

```
["bodytext" , {
  "alignment": "justify",
  "hyphenation": true
}],
```

If you are not an english speaker, you need to change the language using the -l or --language option.

Since Wordaxe version 0.2.6, it can use the PyHyphen library if it's available. PyHyphen can use any OpenOffice dictionary, and can even download them automatically. [3]

For example, this will enable german hyphenation globally:

```
rst2pdf -l de_DE mydocument.txt
```

If you are creating a multilingual document, you can declare styles with specific languages. For example, you could inherit bodytext for spanish:

```
["bodytext_es" , {
  "parent": "bodytext",
  "alignment": "justify",
  "hyphenation": true,
  "language": "es_ES"
}],
```

And all paragraps declared of bodytext_es style would have spanish hyphenation:

```
.. class:: bodytext_es

Debo a la conjunción de un espejo y de una enciclopedia el descubrimiento de Uqbar.
El espejo inquietaba el fondo de un corredor en una quinta de la calle Gaona,
en Ramos Mejía; la enciclopedia falazmente se llama *The Anglo-American Cyclopaedía*
(New York, 1917) y es una reimpresión literal, pero también morosa, de la
*Encyclopaedia Britannica* de 1902.
```

Here is the result (made thinner to force hyphenation):

Debo a la conjunción de un espejo y de una enciclopedia el descubrimiento de Uqbar. El espejo inquietaba el fondo de un corredor en una quinta de la calle Gaona, en Ramos Mejía; la enciclopedia falazmente se llama *The Anglo-American Cyclopaedía* (New York, 1917) y es una reimpresión literal, pero también morosa, de la *Encyclopaedia Britannica* de 1902.

BTW: That's the beginning of "Tlön, Uqbar, Orbis Tertius", read it, it's cool.

If you explicitly configure a language in a paragraph style and also pass a language in the command line, the style has priority, so remember:

> ***Important***
>
> If you configure the bodytext style to have a language, your document is supposed to be in that language, regardless of what the command line says.
>
> If this is too confusing, let me know, I will try to figure out a simpler way.

14 Page Layout

By default, your document will have a single column of text covering the space between the margins. You can change that, though, in fact you can do so even in the middle of your document!

To do it, you need to define *Page Templates* in your stylesheet. The default stylesheet already has 3 of them:

```
"pageTemplates" : {
  "coverPage": {
      "frames": [
          ["0cm", "0cm", "100%", "100%"]
      ],
      "showHeader" : false,
      "showFooter" : false
  },
  "oneColumn": {
      "frames": [
          ["0cm", "0cm", "100%", "100%"]
      ]
  },
  "twoColumn": {
      "frames": [
          ["0cm", "0cm", "49%", "100%"],
          ["51%", "0cm", "49%", "100%"]
      ]
  }
}
```

A page template has a name (oneColumn, twoColumn), some options, and a list of frames. A frame is a list containing this:

```
[ left position, top position, width, height ]
```

For example, this defines a frame "at the very left", "at the very top", "a bit less than half a page wide" and "as tall as possible":

```
["0cm", "0cm", "49%", "100%"]
```

And this means "the bottom third of the page":

```
["0cm", "66.66%", "100%", "33.34%"]
```

You can use all the usual units, cm, mm, inch, and % which means "percentage of the page (excluding margins and headers or footers)". Using % is probably the smartest for columns and gives you a fluid layout, while the other units are better for more "fixed" elements.

Since we can have more than one template, there is a way to specify which one we want to use, and a way to change from one to another.

To specify the first template, do it in your stylesheet, in pageSetup (oneColumn is the default):

```
"pageSetup" : {
  "firstTemplate": "oneColumn"
}
```

Then, to change to another template, in your document use this syntax (will change soon, though):

```
.. raw:: pdf

   PageBreak twoColumn
```

That will trigger a page break, and the new page will use the twoColumn template.

You can see an example of this in the *Montecristo* folder in the source package.

The supported page template options and their defaults are:

- showHeader : True
- defaultHeader : None

  Has the same effect as the header directive in the document.
- showFooter : True
- defaultFooter : None

  Has the same effect as the footer directive in the document.
- background: None

  The background should be an image, which will be stretched to match your page, so use with caution.

15 Smart Quotes

Quoted from the smartypants documentation:

> This feature can perform the following transformations:
>
> Straight quotes (" and ') into "curly" quote HTML entities Backticks-style quotes (``like this'') into "curly" quote HTML entities Dashes (-- and ---) into en- and em-dash entities Three consecutive dots (... or . . .) into an ellipsis entity This means you can write, edit, and save your posts using plain old ASCII straight quotes, plain dashes, and plain dots, but your published posts (and final PDF output) will appear with smart quotes, em-dashes, and proper ellipses.

You can enable this by passing the `--smart-quotes` option in the command line. By default, it's disabled. Here are the different values you can use (again, from the smartypants docs):

0

Suppress all transformations. (Do nothing.)

1

Performs default SmartyPants transformations: quotes (including ``backticks'' -style), em-dashes, and ellipses. "--" (dash dash) is used to signify an em-dash; there is no support for en-dashes.

2

Same as smarty_pants="1", except that it uses the old-school typewriter shorthand for dashes: "--" (dash dash) for en-dashes, "---" (dash dash dash) for em-dashes.

3

Same as smarty_pants="2", but inverts the shorthand for dashes: "--" (dash dash) for em-dashes, and "---" (dash dash dash) for en-dashes.

Currently, even if you enable it, this transformation will only take place in regular paragraphs, titles, headers, footers and block quotes.

16 Kerning

Kerning is the process of adjusting letter spacing. It is usually accepted that kerning makes your text look better.

For example, if you are using proper kerning, the As and Ws in AWAWA will overlap slightly.

If you want kerning in your PDFs, you need to do the following:

- Use wordaxe at least 1.0.0
- Use a TrueType font
- Set kerning to true in your style. For example, if you want **all** text to be kerned, you can set it in the "base" style.

For convenience, a stylesheet that uses DejaVu fonts with kerning is provided as kerning.json, so you can copy and adapt to your needs, or just use it with the `-s` option.

17 Sphinx

Sphinx is a very popular tool. This is the description from its website:

> Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.
>
> It was originally created to translate the new Python documentation, and it has excellent support for the documentation of Python projects, but other documents can be written with it too.

Rst2pdf includes an experimental PDF extension for sphinx.

To use it in your existing sphinx project you need to do the following:

1. Add rst2pdf.createpdf in your conf.py's extensions. For example:

```
extensions = ['sphinx.ext.autodoc','rst2pdf.pdfbuilder']
```

2. Add the PDF options at the end of conf.py, adapted to your project:

```
# -- Options for PDF output --------------------------------------------------

# Grouping the document tree into PDF files. List of tuples
# (source start file, target name, title, author, options).
#
# If there is more than one author, separate them with \\.
# For example: r'Guido van Rossum\\Fred L. Drake, Jr., editor'
#
# The options element is a dictionary that lets you override
# this config per-document.
# For example,
# ('index', u'MyProject', u'My Project', u'Author Name',
#  dict(pdf_compressed = True))
# would mean that specific document would be compressed
# regardless of the global pdf_compressed setting.

pdf_documents = [
    ('index', u'MyProject', u'My Project', u'Author Name'),
]

# A comma-separated list of custom stylesheets. Example:
pdf_stylesheets = ['sphinx','kerning','a4']

# Create a compressed PDF
# Use True/False or 1/0
# Example: compressed=True
#pdf_compressed = False

# A colon-separated list of folders to search for fonts. Example:
# pdf_font_path = ['/usr/share/fonts', '/usr/share/texmf-dist/fonts/']

# Language to be used for hyphenation support
#pdf_language = "en_US"

# Mode for literal blocks wider than the frame. Can be
```

```
# overflow, shrink or truncate
#pdf_fit_mode = "shrink"

# Section level that forces a break page.
# For example: 1 means top-level sections start in a new page
# 0 means disabled
#pdf_break_level = 0

# When a section starts in a new page, force it to be 'even', 'odd',
# or just use 'any'
#pdf_breakside = 'any'

# Insert footnotes where they are defined instead of
# at the end.
#pdf_inline_footnotes = True

# verbosity level. 0 1 or 2
#pdf_verbosity = 0

# If false, no index is generated.
#pdf_use_index = True

# If false, no modindex is generated.
#pdf_use_modindex = True

# If false, no coverpage is generated.
#pdf_use_coverpage = True

# Documents to append as an appendix to all manuals.
#pdf_appendices = []

# Enable experimental feature to split table cells. Use it
# if you get "DelayedTable too big" errors
#pdf_splittables = False

# Set the default DPI for images
#pdf_default_dpi = 72

# Enable rst2pdf extension modules (default is empty list)
# you need vectorpdf for better sphinx's graphviz support
#pdf_extensions = ['vectorpdf']

# Page template name for "regular" pages
#pdf_page_template = 'cutePage'
```

3. (Maybe) add this in your Makefile (on unix-like systems):

```
pdf:
    $(SPHINXBUILD) -b pdf $(ALLSPHINXOPTS) _build/pdf
    @echo
    @echo "Build finished. The PDF files are in _build/pdf."
```

4. (Maybe) add this to your make.bat (on windows):

```
if "%1" == "pdf" (
    %SPHINXBUILD% -b pdf %ALLSPHINXOPTS% %BUILDDIR%/pdf
    echo.
    echo.Build finished. The PDF files are in %BUILDDIR%/pdf
    goto end
)
```

Then you can run `make pdf` or `sphinx-build -b pdf ...` similar to how you did it before.

18 Extensions

Rst2pdf can get new features from *extensions*. Extensions are python modules that can be enabled with the -e option.

Several are included with rst2pdf.

18.1 Preprocess (`-e preprocess`)

preprocess is a rst2pdf extension module (invoked by -e preprocess on the rst2pdf command line.

It preprocesses the source text file before handing it to docutils.

This module serves two purposes:

1. It demonstrates the technique and can be a starting point for similar user-written processing modules; and
2. It provides a simplified syntax for documents which are targeted only at rst2pdf, rather than docutils in general.

The design goal of "base rst2pdf" is to be completely compatible with docutils, such that a file which works as a PDF can also work as HTML, etc.

Unfortunately, base docutils is a slow-moving target, and does not make this easy. For example, SVG images do not work properly with the HTML backend unless you install a patch, and docutils has no concept of page breaks or additional vertical space (other than the <hr>).

So, while it would be nice to have documents that render perfectly with any backend, this goal is hard to achieve for some documents, and once you are restricted to a particular transformation type, then you might as well have a slightly nicer syntax for your source document.

Preprocessor extensions:

All current extensions except style occupy a single line in the source file.

`.. include::`

Processes the include file as well.

`.. page::`

Is translated into a raw PageBreak

`.. space::`

Is translated into a raw Spacer. If only one number given, is used for vertical

`.. style::`

Allows you to create in-line stylesheets. (If you wish them to be in YAML format, you must give -e yaml before -e preprocess on the command line.)

`.. widths::`

creates a new table style (based on table or the first non-numeric token) and creates a class using that style for the next table. Allows you to set the widths for the table, using percentages.

`SingleWordAtLeftColumn`

If this is surrounded by blank lines, the singleword style is applied to the word. This is a workaround for the broken interaction between docutils subtitles and bibliographic metadata. Who wants a subtitle inside the TOC?

Preprocessor operation:

The preprocessor generates a file that has the same name as the source file, with .build_temp. embedded in the name, and then passes that file to the restructured text parser.

This file is left on the disk after operation, because any error messages from docutils will refer to line numbers in it, rather than in the original source.

18.2 Inkscape (`-e inkscape`)

inkscape.py is an rst2pdf extension (e.g. rst2pdf -e inkscape xxx xxxx) which uses the inkscape program to convert an svg to a PDF, then uses the vectorpdf code to process the PDF.

> ***Note***
>
> The initial version is a proof of concept; uses subprocess in a naive way, and doesn't check return from inkscape for errors.

18.3 Dotted_TOC (`-e dotted_toc`)

All I did was take the wrap() method from the stock reportlab TOC generator, and make the minimal changes to make it work on MY documents in rst2pdf.

18.3.1 History:

The reportlab TOC generator adds nice dots between the text and the page number. The rst2pdf one does not.

A closer examination reveals that the rst2pdf one probably deliberately stripped this code, because the reportlab implementation only allowed a single TOC, and this is unacceptable for at least some rst2pdf users.

There are other differences in the rst2pdf one I don't understand. This module is a hack to add back dots between the lines. Maybe at some point we can figure out if this is right, or how to support dots in the TOC in the main code.

Mind you, the original RL implementation is a complete hack in any case:

- **It uses a callback to a nested function which doesn't even bother to**

  assume the original enclosing scope is available at callback time. This leads it to do crazy things like eval()

- **It uses a single name in the canvas for the callback function**

  (this is what kills multiple TOC capability) when it would be extremely easy to generate a unique name.

19 Developers

19.1 Guidelines

In rst2pdf we want many things. We want ponies and icecream. But most of all, we want rst2pdf to kick ass. The best way to achieve that is making rst2pdf work right. The best way to do *that* is through testing and documenting.

So, if you want to do something inside rst2pdf, you are welcome, but...

- Create an Issue for the task. That's easy, just go to http://rst2pdf.googlecode.com and do it.
- If you intend to fix a bug:
  - Create a **minimal** test case that shows the bug.
  - Put it inside rst2pdf/tests/input like the others:
    - mytest.txt is the test itself
    - mytest.cli is any needed command line arguments (if needed)
    - mytest.style is a custom stylesheet (if needed)
  - Run the test suite on it:

    ```
    cd rst2pdf/tests
    ./autotest.py input/mytest.txt
    ```

  - Check the output:

    less output/mytest.log acroread output/mytest.pdf

  - If it's really a bug, mark the test as *bad* and save everything in SVN:

    ```
    setmd5 bad input/mytest.txt
    svn add input/mytest.*
    svn add md5/mytest.json
    svn commit -m "Test case for Issue X"
    ```

- Always, when committing something, check for regressions running the full test suite, it takes only a minute or two. Keep in mind that regressions can be trivial!

  For example, if you change the spacing of definition lists, 3 or 4 tests will regress.
- Keep your Issues updated. If you are working on frobnozzing the gargles, then by all means post it in the issue. There's no issue about it? You were meant to create one, remember? ;-)
- If you added a command line option, document it in doc/rst2pdf.txt. That will make it appear in the manual and in the man page.

  Maybe it should also be available for sphinx users, let me know about it.
- If you implemented a new feature, please document it in manual.txt (or in a separate file and add an include in manual.txt)
- If you implement an extension, make the docstring valid restructured text and link it to the manual like the others.

Why should you bother with all this?

It's important that you do it this way because it means that the rest of us know what you are doing. It also means you don't break rst2pdf.

19.2 Continuous Integration

rst2pdf has a semi-public CI server running Hudson. What's CI? It's a server running the test suite all the time. That means that if a commit breaks something, we can find out about it.

And yes, you are supposed to check for regressions yourself, but will you test against ReportLab 2.3? How about Python 2.4? Hudson allows us to have all those builds (and more) running every day, so we know it doesn't break for anyone else.

Or at least, we hope so, this is a work in progress ;-)

19.3 Running tests

19.3.1 first run

while in project:

```
python bootstrap.py
./bin/buildout
./bin/nosetests -i regulartest -i sphinxtest
```

19.3.2 next runs

while in project:

```
./bin/nosetests -i regulartest -i sphinxtest
```

19.4 Getting commit rights

Just ask in the mailing list.

20 Licenses

This is the license for rst2pdf:

```
Copyright (c) 2007,2008,2009 Roberto Alsina
Nicolas Laurance, Christoph Zwerschke, Yasushi Masuda, Josh VanderLinden,
Patrick Maupin.


Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

Some fragments of rstpdf are copied from ReportLab under the following license:

```
Copyright (c) 2000-2008, ReportLab Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

        *       Redistributions of source code must retain the above copyright notice,
                this list of conditions and the following disclaimer.
        *       Redistributions in binary form must reproduce the above copyright notice,
                this list of conditions and the following disclaimer in the documentation
                and/or other materials provided with the distribution.
        *       Neither the name of the company nor the names of its contributors may be
                used to endorse or promote products derived from this software without
                specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE OFFICERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

---

1 The `/etc/rst2pdf.conf` location makes sense for Linux and linux-like systems. if you are using rst2pdf in other systems, please contact me and tell me where the system-wide config file should be.

2 You can get Wordaxe from http://deco-cow.sf.net. Version 1.0.0 or later is recommended.

3 For more information, please check the PyHyphen website at http://pyhyphen.googlecode.com