How to use rst2pdf

Author: Roberto Alsina <ralsina@netmanagers.com.ar>

Version: 0.6

Revision: 293

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
`-o FILE, --output=FILE`	Write the PDF to FILE
`-s STYLESHEETS, --stylesheets=STYLESHEETS`	A comma-separated list of custom stylesheets
`-c, --compressed`	Create a compressed PDF
`--print-stylesheet`	Print the default stylesheet and exit
`--font-folder=FOLDER`	Search this folder for fonts.
`-l LANG, --language=LANG`	Language to be used for hyphenation.
`--fit-literal-mode=MODE`	What todo when a literal is too wide. One of error,overflow,shrink,truncate. Defaults to truncate.
`-b LEVEL, --break-level=LEVEL`	Maximum section level that starts in a new page. Default: 1
`-q, --quiet`	Print less information.
`-v, --verbose`	Print debug information.
`--very-verbose`	Print even more debug information.

3 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 will 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.

Headers and footers are visible by default but they can be disabled by specific Page Templates for example, cover pages.

4 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.

I could go the HTML path and just render them where you put them, but then they wouldn't be footnotes either, would they? They would be "in-the-middle-of-text-notes" or just plain notes.

5 Inline Images

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

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

.. |biohazard| image:: biohazard.png
```

This means you have to run.

However, although I think rst2pdf generates the right reportlab code, the image will not be visible.

It will work, however, if you use a development version of reportlab. In which case wordaxe won't work, and you will have no hyphenation. Your choice ;-)

6 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 take the standard stylesheet, modify it and pass it with the -s option:

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

6.1 StyleSheet Syntax

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

6.2 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

6.3 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 styleheet, 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":"Times-Roman",
"fontSize":10,
"leading":12,
"leftIndent":0,
"rightIndent":0,
"firstLineIndent":0,
"alignment":TA_LEFT,
"spaceBefore":0,
"spaceAfter":0,
"bulletFontName":"Times-Roman",
"bulletFontSize":10,
"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

Notice that backColor is **not** in that list.

6.4 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.

6.4.1 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?

a. 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 ;-)

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

b. 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, regardless of whether it's True Type or Type 1.

6.4.2 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 (included in rst2pdf's source distribution).

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 that is the order in which you should put them. 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",
  "bulletFontName": "Tuffy_Bold",
  "fontSize": 18,
  "bulletFontSize": 18,
  "leading": 22,
  "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.

6.4.3 The Harder Way (Type1)

To be written (and implemented and tested)

6.5 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",
  "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.

6.6 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 chage page size, default font and font size, this would be enough:

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

7 Syntax Highlighting

7.1 Inline

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.

Rst2pdf includes several stylesheets for highlighting code:

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

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

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

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

The default is the same as "emacs".

7.1.1 Examples

As rst2pdf is 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
```

7.2 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.

7.2.1 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

```
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)
```

8 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).

The syntax is shell-like, here's an example that shows all the syntax:

```
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 is points, and using a space or a comma is the same thing in all cases.

9 Hyphenation

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

To do it, you need to install Wordaxe version 0.2.5 or later from http://deco-cow.sf.net.

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" , {
  "parent": "normal",
  "spaceBefore":6,
  "alignment": "TA_JUSTIFY",
  "hyphenation": true
}],
```

If you are not an english speaker, you need to change the language.

You can use the -l or --language option. The currently available dictionaries for wordaxe are:

- de_DE
- da
- en_GB
- en_US
- ru

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 german:

```
["bodytext_de" , {
  "parent": "bodytext",
  "alignment": "TA_JUSTIFY",
  "hyphenation": true,
  "language": "de_DE"
}],
```

And all paragraps declared of bodytext_de style would have german hyphenation:

```
.. class:: bodytext_de

Ein Vierteljahrhundert hindurch hatte ich Kopf, Herz, Hand und--Füße der
Schilderung der Alpenwelt und ihrer Bewohner gewidmet mit dem
erfreulichen Erfolg, daß die deutsche Leserwelt es gewöhnt geworden war,
beim Anblick meines Namens auf Büchern sofort an die--Alpen zu denken.
```

BTW: I have no idea what that says, I just copied it from project Gutenberg. Hopefully it's not offensive :-)

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:

> 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.

10 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
- showFooter : True