How to use

rst2pdf

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 (rST) 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.

1.1 Related Reading

As well as the rst2pdf-specific features described in this manual, you many also find it useful to refer to the ReStructuredText manual and information about its directives:

- A ReStructureText Primer: https://docutils.sourceforge.io/docs/user/rst/quickstart.html
- Quick ReStructuredText: https://docutils.sourceforge.io/docs/user/rst/quickref.html
- ReStructuredText Specification: https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html
- ReStructuredText Directives: https://docutils.sourceforge.io/docs/ref/rst/directives.html

2   Command line options

Use the following options to control the output of *rst2pdf* on the command line.

2.1   General Options

Option	Description
`-h, --help`	Show the help message and exit.
`--version`	Print the version number and exit.
`-q, --quiet`	Print less information.
`-v, --verbose`	Print debug information.
`--very-verbose`	Print even more debug information.

2.2   File and Configuration

Option	Description
`--config=FILE`	Config file to use. Default: `~/.rst2pdf/config`.
`-o FILE, --output=FILE`	Write the PDF to `FILE`.
`--record-dependencies=FILE`	Write output file dependencies to `FILE`.

2.3   Styling Options

Option	Description
`-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: `""`.
`--print-stylesheet`	Print the default stylesheet and exit.
`--font-path=FOLDERLIST`	A colon-separated list of folders to search for fonts. Default: `""`.

2.4   PDF Options

Option	Description
`-c, --compressed`	Create a compressed PDF. Default: `False`.
`--baseurl=URL`	The base URL for relative URLs.
`--header=HEADER`	Page header if not specified in the document.
`--footer=FOOTER`	Page footer if not specified in the document.
`--first-page-on-right`	When using double-sided pages, the first page will start on the right-hand side (Book Style).
`--blank-first-page`	Add a blank page at the beginning of the document.
`--custom-cover=FILE`	Template file used for the cover page. Default: `cover.tmpl`.

2.5   Formatting Options

Option	Description
`--section-header-depth=N`	Sections up to this depth will be used in the header and footer's replacement of `###Section###`. Default: `2`.
`--smart-quotes=VALUE`	Convert ASCII quotes, ellipses, and dashes to typographically correct equivalents. Default: `0`. Accepted values: • `0`: Suppress all transformations. • `1`: Default transformations for quotes, em-dashes, and ellipses. • `2`: Use typewriter shorthand for dashes. • `3`: Invert shorthand for dashes.
`--fit-literal-mode=MODE`	Handle literals that are too wide. Options: `error`, `overflow`, `shrink`, `truncate`. Default: `shrink`.
`--fit-background-mode=MODE`	Fit the background image to the page. Options: `scale`, `scale_width`, `center`. Default: `center`.

2.6   Miscellaneous Options

Option	Description
`-e EXTENSIONS, --extension-module=EXTENSIONS`	Add a helper extension module (must end in `.py` and be on the Python path).
`--inline-links`	Show targets in parentheses instead of active links.
`--repeat-table-rows`	Repeat the header row for each split table.
`--raw-html`	Support embedding raw HTML. Default: `False`.
`--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 (useful for debugging). Default: `False`.
`--disable-splittables`	Disable splittable flowables in some elements. Useful if a document cannot otherwise be processed.
`--break-side=VALUE`	Section break behavior. Options: `even`, `odd`, `any`.

3 Configuration File

The configuration file uses an **INI-style** format with sections and key-value pairs. Comments are prefixed with #.

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]

3.1 Configuration Options

The table below provides detailed descriptions of the available configuration options.

Option	Description	Default Value
`stylesheets`	Comma-separated list of custom stylesheets.	`""`
`compressed`	Generate a compressed PDF. Use `true`/`false` or `1`/`0`.	`false`
`font_path`	Colon-separated list of folders to search for fonts.	`""`
`stylesheet_path`	Colon-separated list of folders to search for stylesheets.	`""`
`language`	Language for hyphenation and localization.	`en_US`
`header`	Default page header. Use `null` for no header.	`null`
`footer`	Default page footer. Use `null` for no footer.	`null`
`fit_mode`	Handle oversized literal blocks. Options: `shrink`, `truncate`, `overflow`.	`shrink`
`fit_background_mode`	Adjust background images. Options: `scale`, `center`.	`center`
`break_level`	Maximum heading level that starts on a new page.	`0`
`break_side`	Section break alignment. Options: `even`, `odd`, `any`.	`any`
`blank_first_page`	Add a blank page at the start of the document.	`false`
`first_page_even`	Treat the first page as even.	`false`

`smartquotes`	Configure smart quotes transformation. Accepted values: • `0`: Suppress all transformations. • `1`: Default transformations for quotes, em-dashes, and ellipses. • `2`: Use typewriter shorthand for dashes. • `3`: Invert shorthand for dashes.	`0`
`footnote_backlinks`	Enable footnote backlinks.	`true`
`inline_footnotes`	Show footnotes inline.	`false`
`custom_cover`	Template file for the cover page.	`cover.tmpl`
`floating_images`	Enable floating images for alignment.	`false`
`raw_html`	Enable support for the `..raw:: html` directive.	`false`

3.2 Example Configuration File

Here's an example configuration file showing the expected format:

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

[general]
stylesheets="fruity.json,a4paper.json,verasans.json"

# Folders to search for stylesheets.
stylesheet_path="~/styles:/usr/share/styles"

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

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 output is set to dash (-), output goes to `stdout`:

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

5   Images

5.1   Inline

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

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

.. |biohazard| image:: assets/biohazard.png
```

This ☣ means you have to run.

5.2   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 SVG support, you need to install svglib.

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

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

> Note
>
> Image URLs
>
> Attempting to be more compatible with rst2html, rst2pdf will try to handle images specified as HTTP or FTP URLs by downloading them to a temporary file and including them in the PDF.
>
> This is probably not a good idea unless you are **really** sure the image won't go away.

5.3   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).

6   Styling ReStructuredText

For well-formatted and consistent PDFs, the best starting point is well-formatted and consistent markup. There are some excellent references for ReStructuredText which we won't reproduce here but they are highly recommended as a starting point for working with rst2pdf.

In general, applying a stylesheet to a structured document will output a decent PDF with minimum fuss. That said, there are plenty of customisation and styling options available so read on if that sounds interesting.

6.1   Applying Styles

rst2pdf applies a default set of styles to the document. This default set can be viewed using `rst2pdf --print-stylesheet` which prints outh `rst2pdf/styles/styles.yaml`.

Each subsequent style within each style sheet file specified the `--stylesheets` CLI parameter is then registered in the the list of known styles known to rst2pdf. If the name of the style is already known, then the attributes specified in the style are applied "on top" of the already registered style.

rst2pdf will then resolve the `parent` style, which is why the order of inclusion matters per-style-name, not globally. That is, if you set the color of `bodytext` first in a file and then set the color of `normal` in a subsequent file, then the color you have set for `bodytext` will be the color used for paragraphs (unless overridden by a `class` directive. Further information on cereating stylesheet files is available in Creating Stylesheets.

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

```
.. class:: special

This paragraph is special.

This one is not.
```

Multiple styles can be listed and are applied in order where properties in the right hand styles override those to the left:

```
.. class:: special bluetext redtext

    This paragraph is special and is red.

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 rST docs, and for style information check the section in this manual on inline styles.

6.2   Headers and Footers

rST 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 current section title

###SectNum###

Replaced by the current 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

6.3 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   Customizing PDF Output

Stylesheets are used to control many aspects of the PDF output.

- General look and feel, colours, fonts, templates
- Page size
- Syntax highlighting for code

The stylesheets use a YAML format (JSON is also supported). Older versions of this tool used an RSON format; this is also still supported but we recommend you check the section on *migrating to yaml stylesheets* and update them (it's painless!)

7.1   Using Stylesheets

Specify a stylesheet to use with -s:

```
rst2pdf mydoc.rst -s mystyles
```

Often it makes sense to specify multiple stylesheets, for example to set the page size, the main styles, and some syntax highlighting. In that case, use comma-separated values:

```
rst2pdf mydoc.rst -s a4,mystyles,murphy
```

Order does matter: rst2pdf applies its own stylesheet first and then the list in given in order, so the last stylesheet in the list will take precedence over the ones that went before.

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.

7.2   Included StyleSheets

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

Font styles

These stylesheets modify 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

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
```

7.3 Default Stylesheet

You can make rst2pdf print the default stylesheet:

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

This makes an excellent starting point for creating a stylesheet. The default one is always included by default, so only the values that should be changed need to be included in the new stylesheet.

7.4 Migrating Stylesheet Format

Historically, (version 0.98 and earlier) rst2pdf had support for JSON and RSON stylesheets. Those stylesheets should still work if you are still using them but a warning will be produced:

```
[WARNING] styles.py:617 Stylesheet "./example.style" in outdated format, recommend converting to YAML
```

To update your stylesheet, use the `rst2pdf.style2yaml` utility:

```
python3 -m rst2pdf.style2yaml example.style
```

The command also accepts a list of paths, or wildcards, and by default will output the new stylesheet(s) to stdout. To write them to files instead, use the `--save` flag with the command above.

7.5 Migrating to the New Default Stylesheet

Historically (version 0.98 and earlier), rst2pdf used a different default style sheet. The updated default style file provide a more modern look to rst2pdf documents. To do this, it updates various spacing, margins and fonts. It also updates page template and font alias names and so you will need to make adjustments to derived style files.

Until you make these adjustments, you can use the historical default style sheet using by adding the `rst2pdf-0-9` style using the `-s` command line switch. For example:

```
rst2pdf mydoc.rst -s rst2pdf-0-9,mystyle.yaml
```

7.5.1 Updated Font Alias Names

The font aliases used for the standard fonts have changed from those used in the historical default style sheeet. As such, you will need to update to the new names in any derivative style files.

This table shows the old name and the equivalent new name:

Historical	Current
stdFont	fontSerif
stdSerif	fontSerif
stdBold	fontSerifBold
stdBoldItalic	fontSerifBoldItalic
stdItalic	fontSerifItalic
stdMono	fontMono
stdMonoBold	fontMonoBold
stdMonoBoldItalic	fontMonoBoldItalic
stdMonoItalic	fontMonoItalic
stdSans	fontSans
stdSansBold	fontSansBold
stdSansBoldItalic	fontSansBoldItalic
stdSansItalic	fontSansItalic

7.5.2   Updated Pate Template Names

The page template names used in the new default style sheet are different from the historical default style sheeet. As such, you will need to update to the new names in any derivative style files.

This table shows the old name and the equivalent new name:

Historical	Current
–	mainPage
cutePage	decoratedPage
emptyPage	emptyPage
oneColumn	oneColumn
twoColumn	Move to separate twocolumn template file
threeColumn	–

Note that firstTemplate is now mainPage. Historically, it was oneColumn.

8   Creating Stylesheets

The stylesheets are YAML-formatted and give control over many aspects of how the PDF is rendered. The main aspects are the styles of the elements, the page setup and templates, and the fonts to use . These are described in the following sections.

Only the settings that you want to change need to be included so for example, this would be a valid stylesheet:

```yaml
pageSetup:
  size: A5
fontsAlias:
  fontSerif: Times-Roman
styles:
  normal:
    fontSize: 14
```

8.1   Styles in Detail

At the top level there is a bit of an outlier: `linkColor`. You can specify any color name or a hex value:

```yaml
linkColor: #330099
```

Most of the other elements for colours and formatting are in the *styles* section.

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

Here's an example, the `twelvepoint` stylesheet:

```yaml
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:

```yaml
styles:
 - bodytext:
      alignment: TA_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.

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.

8.2   Style Elements

Within the `styles` element, it is possible to configure each element type. The following section lays out the known options and examples of how to use them. (This list is known to be incomplete, we're working on it and accept any additions you have).

parent

Each style property can inherit from another, for example the `code` style inherits from the `literal` style which sets the font used for fixed-width text throughout the document.

Example:

```
code:
   parent: literal
```

fontName

The name of the font to use for this type of element. It can be either the name of a font on your system, or one of the aliased fonts. The default is Helvetica as shown in the example here.

Example:

```
fontName: Helvetica
```

See also:

- Font Alias
- Fonts

fontSize

Use either a number (meaning point size) or a percentage. The default size for bodytext is 10.

Example:

```
fontSize: 150%
```

leftIndent and **rightIndent**

Example:

```
leftIndent: 0
rightIndent: 0
```

firstLineIndent

Example:

```
firstLineIndent: 0
```

alignment

The paragraph justification of the text. The values `TA_LEFT` and `TA_RIGHT` can be used.

Example:

```
alignment: TA_LEFT
```

spaceBefore and **spaceAfter**

The amount of vertical space included before or after an element. Especially useful when working with `bullet-list` and `bullet-list-item` elements.

Example:

```
spaceBefore: 4
spaceAfter: 8
```

bullet -related styles

The bullets can be complex to style, but there are some tricks that might help. The vertical space before and after the list and item elements are controlled by the `spaceBefore` and `spaceAfter` properties. Also these lists are *tables* so those styles also apply.

Example:

```
bulletFontName: Helvetica
bulletFontSize: 10
bulletText: "\u2022"
bulletIndent: 0
```

See also:

- Table Styles

textColor

Use either a color name, or a hex value including the # character at the start.

Example:

```
textColor: black
```

backColor

Use either the value `None`, a color name, or a hex value including the # character at the start. Sets the background color of the element.

Example:

```
backColor: beige
```

wordWrap

Can be set to `None`.

Example:

```
wordWrap: None
```

border -related styles

Setting and styling the border for an element. The example is from the default code block style.

Example:

```
borderColor: darkgray
borderPadding: 6
```

```
borderWidth: 0.5
borderRadius: None
```

allowWidows and **allowOrphans**

These directives are passed to ReportLab if they are present. Currently only implemented for paragraph styles.

Example:

```
allowWidows: 5
allowOrphans: 4
```

See also:

- Widows and Orphans

margin -related styles

This sets the margins of the element. On the `pageSetup` itself, you can use `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 opposing pages. 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.

Example:

```
margin-top: 2cm
margin-bottom: 2cm
margin-left: 2cm
margin-right: 2cm
margin-gutter: 0cm
```

8.2.1   Inline Styles

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

- `fontName`
- `fontSize`
- `textColor`
- `backColor`

8.2.2   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:

```
styles:
  item-list
      parent: bodytext
      spaceBefore: 0
      commands:
      - - VALIGN: [[0, 0], [-1, -1]]
          - TOP
      - - RIGHTPADDING: [[0, 0], [1, -1], 0]
      colWidths:
```

```
       - 20pt
 - bullet-list
     parent: bodytext
     spaceBefore: 0
     commands:
     - - VALIGN: [[0, 0], [-1, -1]]
         - TOP
     - - RIGHTPADDING: [[0, 0], [1, -1], 0]
     colWidths:
     - '20'
```

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.

8.3   Page Layout

There are some layouts available as standard stylesheets, but it is likely that you will also want to describe your own templates.

8.3.1   Page Setup

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

Here's the default stylesheet's element:

```
pageSetup:
   size: A4
   width:
   height:
   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). For example, a slide deck in a 16:9 ratio can be created as a document with width 32cm and height 18cm:

```
pageSetup:
    size: null
```

```
    width: 32cm
    height: 18cm
```

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

8.3.2   Page Templates

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 three 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, bottom position, width, height, left padding, bottom padding, right padding, top padding]
```

All the padding values are optional and default to 6 points.

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

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

And this means "the top 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 centered in your page or stretched to match your page size, depending on the `--fit-background-mode` option, so use with caution.

8.4   Font Alias

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

```
fontsAlias:
  fontSerif: Helvetica
  fontSerifBold: Helvetica-Bold
  fontSerifItalic: Helvetica-Oblique
  fontSerifBoldItalic: Helvetica-BoldOblique
  fontMono: 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 have to change it in each style. So, I suggest you use aliases.

More information in the dedicated Fonts section.

8.5   Widows and Orphans

Widow
    A paragraph-ending line that falls at the beginning of the following page/column, thus separated from the remainder of the text.

Orphan
    A paragraph-opening line that appears by itself at the bottom of a page/column.

rst2pdf has *some* widow/orphan control. Specifically, here's what's currently implemented:

On ordinary paragraphs, `allowWidows` and `allowOrphans` is passed to reportlab, which is supposed to do something about it if they are non-zero. In practice, it doesn't seem to have much effect.

The plan is to change the semantics of those settings, so that they mean the minimum number of lines that can be left alone at the beginning of a page (widows) or at the end (orphans).

Currently, these semantics only work for literal blocks and code blocks.

```
A literal block::

    This is a literal block.

A code block:

.. code-block:: python

    def x(y):
        print y**2
```

In future versions this may extend to ordinary paragraphs.

8.6 Table Styles

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 line width 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 auto-calculated. 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.

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" ]
]
```

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
```

```python
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:

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

You can use the `hl_lines` option to emphasize certain lines by dimming the other lines. This parameter takes a space separated list of line numbers. The other lines are then styled with the class `pygments_diml` that defaults to gray. For example, to highlight `print "line a"` and `print "line b"`:

```python
def myFun(x,y):
    print "line a"
    print "line b"
    print "line c"
```

rst2pdf includes several stylesheets for highlighting code:

- `abap`
- `algol_nu`
- `algol`
- `arduino`
- `autumn`
- `borland`
- `bw`
- `colorful`
- `default`
- `emacs`
- `friendly`
- `fruity`
- `igor`
- `lovelace`
- `manni`

- monokai
- murphy
- native
- paraiso-dark
- paraiso-light
- pastie
- perldoc
- rainbow_dash
- rrt
- sas
- solarized-dark
- solarized-light
- sphinx
- stata-dark
- stata-light
- stata
- styles
- tango
- trac
- vim
- vs
- xcode

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 overall look of a code box is controlled by the "code" style or by a class you apply to it using the `.. class::` directive. Additionally, 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
```

The code-block directive supports many options, that mirror Pygments':

```
FIXME: fix this to really explain them all. This is a placeholder.

                          'stripnl' : string_bool,
                          'stripall': string_bool,
                          'ensurenl': string_bool,
                          'tabsize' : directives.positive_int,
                          'encoding': directives.encoding,
                          # Lua
                          'func_name_hightlighting':string_bool,
                          'disabled_modules': string_list,
                          # Python Console
                          'python3': string_bool,
                          # Delphi
                          'turbopascal':string_bool,
                          'delphi' :string_bool,
                          'freepascal': string_bool,
                          'units': string_list,
                          # Modula2
                          'pim'   : string_bool,
                          'iso'   : string_bool,
                          'objm2' : string_bool,
                          'gm2ext': string_bool,
                          # CSharp
                          'unicodelevel' : csharp_unicodelevel,
                          # Literate haskell
                          'litstyle' : lhs_litstyle,
                          # Raw
                          'compress': raw_compress,
                          # Rst
                          'handlecodeblocks': string_bool,
                          # Php
                          'startinline': string_bool,
                          'funcnamehighlighting': string_bool,
                          'disabledmodules': string_list,
```

You can find more information about them in the pygments manual.

9.1 File inclusion

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

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

This will give a warning if `my_script.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 occurrence of string, string **included**

`:start-after: string`

    will include file beginning at the first occurrence of string, string **excluded**

`:end-before: string`

    will include file up to the first occurrence of string, string **excluded**

`:end-at: string`

    will include file up to the first occurrence of string, string **included**

9.1.2 Options

`linenos`

    Display line numbers along the code

`linenos_offset`

    If you include a file and are skipping the beginning, using the `linenos_offset` makes the line count start from the real line number, instead of 1.

10   Fonts

Working with fonts on many different platforms is a challenge. Here you will find the best information we have, but questions and updates are always welcome.

10.1   Standard PDF Fonts

The standard PDF fonts are always available, here is the list:

- `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`

10.2   Font Embedding

There are thousands of excellent 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.

10.2.1   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?

10.2.1.1   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`.

10.2.1.2 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.

10.2.2 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: []
```

The *embeddedFonts* element is a list of the font files that you want to embed into your PDF document. For each font, you provide the filenames of the four variants of the file (normal, bold, italic, bold italic).

For example, 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`, etc..)

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 you list them in the correct 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:
  fontSans: Tuffy
  fontSansBoldfontSansBold: Tuffy_Bold
  fontSansItalic: Tuffy_Italic
  fontSansBoldItalic: Tuffy_Bold_Italic
  fontMono: Courier
```

If, on the other hand, you only want a specific style to use the Tuffy font, don't change the `fontAlias` but rather 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.

11   Raw Directive

11.1   Raw PDF

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 background=images/background.jpg fit-background-mode=scale

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.

11.2   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.

11.3   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 optional argument which is the name of the next page template. For example:

```
PageBreak twoColumn
```

In addition, two additional attributes are supported: `background` and `fit-background-mode`. These allow setting the background image for this page and how to fit it (One of scale, scale_width or center). For example:

```
PageBreak mainPage background="images/background.jpg"
```

or:

```
PageBreak background=images/background.jpg fit-background-mode=scale
```

11.4   Frame Breaks

If you want to jump to the next frame in the page (or the next page if the current frame is the last), you can use the `FrameBreak` command. It takes an optional height in points, and then it only breaks the frame if there is less than that vertical space available.

For example, if you don't want a paragraph to begin if it's less than 50 points from the bottom of the frame:

```
.. raw:: pdf

   FrameBreak 50

This paragraph is so important that I don't want it at the very bottom of
the page...
```

11.5 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 degrees (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
```

11.6 Text Annotations

Text annotations are meta notes added to a page.

The syntax is this:

```
.. raw:: pdf

    TextAnnotation "text to add" [optional position]
```

The optional position is a set of 4 numbers for `x_begin`, `y_begin``, ``x_end` and `y_end`

11.7   Raw HTML

If you have a document that contains raw HTML, and have `xhtml2pdf` installed, `rst2pdf` will try to render that HTML inside your document. To enable this, use the `--raw-html` command line option.

12   The counter role

> **Note**
>
> The counter role only works in PDF, if you're reading the HTML version of the manual then this section is broken. Sorry :/

This is a nonstandard interpreted text role, which means it will only work with `rst2pdf`. It implements an unlimited number of counters you can use in your text. For example, you could use it to have numbered figures, or numbered tables.

The syntax is this:

```
Start a counter called seq1 that starts from 1: :counter:`seq1`
Now this should print 2: :counter:`seq1`

You can start counters from any number (this prints 12): :counter:`seq2:12`

And have any number of counters with any name: :counter:`figures`

So ``#seq1-2`` should link to `the number 2 above <#seq1-2>`_
```

The output is:

Start a counter called seq1 that starts from 1: 1 Now this should print 2: 2

You can start counters from any number (this prints 12): 12

And have any number of counters with any name: 1

Also, the counters create targets for links with this scheme: `#countername-number`.

So `#seq1-2` should link to the number 2 above

13   The version, revision roles

> Note
>
> These are non-standard roles, which means they will only work with rst2pdf and not with rst2html or any other docutils tools.

The `version` and `revision` roles can be used to get the version and revision of an installed Python package. For example:

```
Welcome to rst2pdf :version:`rst2pdf` (:revision:`rst2pdf`)!
```

> Important
>
> The package in question must be installed in the same environment that you are running rst2pdf in.

14   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.

15   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

Note that 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 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" at https://tobi.oetiker.ch/lshort/lshort.pdf

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

16 Hyphenation

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

To do it, you first need to install the `pyphen` python module.

Then, you need to specify the language in each style that you want hyphenation to work. To have hyphenation in the whole document, you can do it in the `base` style.

For example, for an English document, hyphenation can be turned on for the whole document with:

```
base:
   hyphenationLang: en-US
   embeddedHyphenation: 1
```

Notice the `embeddedHyphenation` option. It is optional, but it makes so that hyphenations will give preference to splitting words at embedded hyphens in the text.

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
   hyphenationLang: es-ES
   embeddedHyphenation: 1
```

And all paragraphs declared using the `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.
```

If you want to disable hyphenation in a style that inherits `hyphenationLang` from its parent, you can do so by setting `hyphenationLang` to `0`.

17   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 these 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.

18   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.pdfbuilder` to `extensions` in your `conf.py`. 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', 'MyProject', 'My Project', 'Author Name', {'pdf_compressed': True})
   #
   # would mean that specific document would be compressed
   # regardless of the global 'pdf_compressed' setting.

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

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

   # A list of folders to search for stylesheets. Example:
   pdf_style_path = ['.', '_styles']

   # 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

# Name of the cover page template to use
# pdf_cover_template = 'sphinxcover.tmpl'

# Label to use as a prefix for the subtitle on the cover page
# subtitle_prefix = 'version'

# 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
# pdf_extensions = []

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

# Show Table Of Contents at the beginning?
# pdf_use_toc = True

# How many levels deep should the table of contents be?
pdf_toc_depth = 9999
```

```
# Add section number to section references
pdf_use_numbered_links = False

# Background images fitting mode
pdf_fit_background_mode = 'scale'

# Repeat table header on tables that cross a page boundary?
pdf_repeat_table_rows = True

# Enable smart quotes (1, 2 or 3) or disable by setting to 0
pdf_smartquotes = 0
```

3. (Optional) Modify your `Makefile` or `make.bat` file

    For `Makefile` (on *nix systems)

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

    For `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.

19   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, and you can also develop extensions yourself. Find the included [extensions](#) by inspecting the codebase, each file includes some additional information about the extension.

Extensions include with rst2pdf:

- `dotted_toc` - a (very) experimental extension to add dots to the table of contents list between the titles and the page numbers.

- `fancy_titles` - an experimental extension to render headings with an SVG template.

- `plantuml_r2p` - basic PlantUML support.

- `preprocess` - preprocessing tool to make source file changes before handing it to docutils, can help keep compatibility between different output destinations.

20   Developers

To contribute to rst2pdf, visit the project on GitHub to get started.

21 Licenses

This is the license for rst2pdf:

```
Copyright (c) 2007-2020 Roberto Alsina and the contributors to the rst2pdf project

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.