PlainDoc
PlainDoc
PlainDoc
PlainDoc
Sampo
sampo"@iki.fi

PlainDoc Document Production System

Sampo Kellomäki (sampo@iki.fi)

February 23, 2014

Abstract

PlainDoc ist ein Dokument-Produktionssystem, das auf einfachen Textdateien beruht. Es hält den Großteil des Dokuments in für Menschen lesbarer Form - die PlainDoc - Quelle selbst dient als einfache Textversion des Dokuments. Es handhabt EPS, Gnuplot, Dia-Diagramme, Tabellen und wörtlichen Text, verwendet LaTeX für die Produktion von PDF und kann nativ monolithisches oder in Seiten aufgeteiltes HTML und DocBook erzeugen. Es unterstützt Include-Dateien. Das Dateiformat ist CVS-freundlich und leicht mit diff vergleichbar. Es ist geeignet für Software-Handbücher und Dokumentation, technische Publikationen, wissenschaftliche Arbeiten, Bücher, Dokumente mit rechtlicher Verbindlichkeit und Präsentationsfolien.

1 Introduction to PlainDoc

Plaindoc web site: `http://zxid.org/plaindoc/pd.html`

PlainDoc is a document production system based on plain text files. It tries to keep most of the document in human readable form with the intent that the PlainDoc source code itself will serve as the plain text version of the document.

PlainDoc system was developed by Sampo Kellomäki (`sampo@iki.fi`) from around 2002 onwards with the aim of solving document editing problems for writing:

- IT specifications documents
- Software product manuals and documentation
- Scientific and research papers
- Legal documents

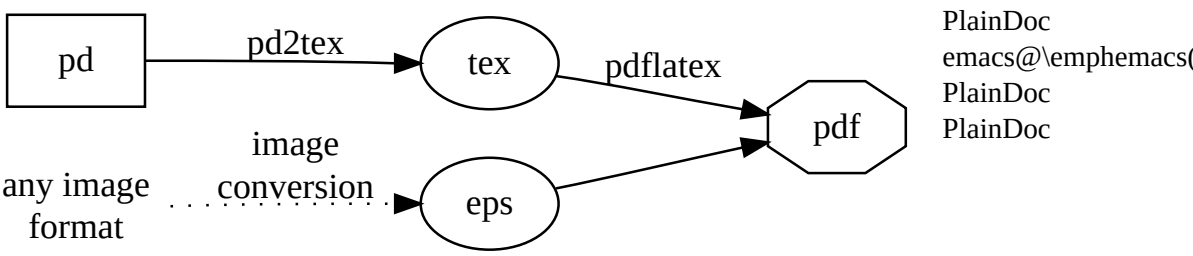


Figure 1: Generation of pdf from sources (simplified)

- Presentation slides

Some of the goals were

- Document source is the plain text representation, no separate conversion needed
- Documents are intuitive to write and understand
- Getting a neophyte to a reasonable level of productivity and achievement should be easy. A college freshman should be able to use PlainDoc after 1 hour training, provided that all the tool chains have already been installed
- It must be very difficult to fatally corrupt a document; fixing corruption should be as simple as editing the file
- It must be possible to do diffs between versions of the document
- Using cvs (or git) should be well supported (helps to avoid fatal loss of document)
- Enable use of plain text productivity environments like *emacs(1)*
- The PlainDoc system MUST be serious enough to produce most any type of document and thus end the need to use any other system
- Typeset quality output in paper and web formats

PlainDoc has now (Sept, 2007) been around for more than five years and it has been successfully used to produce

- Major IT specifications conforming to formatting rules (120 page range)
- Research papers and thesis conforming to formatting rules (200 page range)
- Product manuals (700 page range)
- Legal documents and contracts conforming to formatting rules

PlainDoc
LaTeX
WYSIWYG
PlainDoc
formatter
LaTeX
PlainDoc

PlainDoc acknowledges its LaTeX legacy and does not aim at WYSIWYG (except in plain text document production, of course :-) however we are not totally against visual formatting either. Thus many hooks for accessing the underlying document formatter's capabilities have been made available, such as

- Direct entry of TeX code (allows setting margins, etc.)
- Direct entry of DocBook code
- Direct entry of HTML code
- Support explicit line and page breaks
- Support for raw image placement (i.e. NOT using floats)

These should allow you to get your job done without the system philosophy standing too much in the way, while for most part leveraging the automatic formatting of standard constructs.

1.1 Tool chains

The PlainDoc system is actually composed of multiple programs. Most important of them is the *pd2tex* formatter (which despite of its name actually produces other formats too), but no meaningful output, other than HTML, can be obtained without a properly configured backend formatting tool chain, such as LaTeX system or DocBook tool chain. Some more frontend tools may be helpful if you need to add diagrams or images to your documents. Tables on next page explain some of the tools.

1.2 Data flow

PlainDoc system is best understood as a process rather than an application. Understanding of complex documents is easier if you think about which files are the

make@\emphmake()
LaTeX

Table 1: Backend Tools used in a PlainDoc environment

Tool	Purpose
pd2tex	The main PlainDoc processor itself
LaTeX (teTeX)	Typesetting system, PostScript and PDF backends
gs (GhostScript)	Rendering back-end
make	Automate document generation and maintenance
cvs, svn, git	Version control and collaboration (optional)
perl	Tools are written in perl, but use few modules
gcc	For compiling the tools (optional)

Table 2: Frontend Tools used in a PlainDoc environment (all optional)

Tool	Purpose
GraphViz / dot	Draw graphs (vectorial) from textual input
gnuplot	Draw graphs (vectorial) from statistical data
dia	Vectorial diagram (hand drawn) support
gimp	Bitmap graphics and photography support
ImageMagick	Automated processing of bitmap graphics
gv (GhostView)	Previewing tool for postscript and pdf
acroread	Previewing tool for pdf
xpdf	Another previewing tool for pdf
emacs	Edit text, GUI for invoking commands

sources, how data flows from them to intermediate files, and finally gets assembled to the document, and possibly converted to target format. Programmers will recognize that pd2tex behaves very much like *make(1)*, checking which source files, like images, changed, and runs the commands necessary to convert them to pdf[1] and then triggers the LaTeX system to produce the final document.

[1]PDF is the most preferred form to import images to PlainDoc or LaTeX documents. Everything else gets internally converted to PDF.

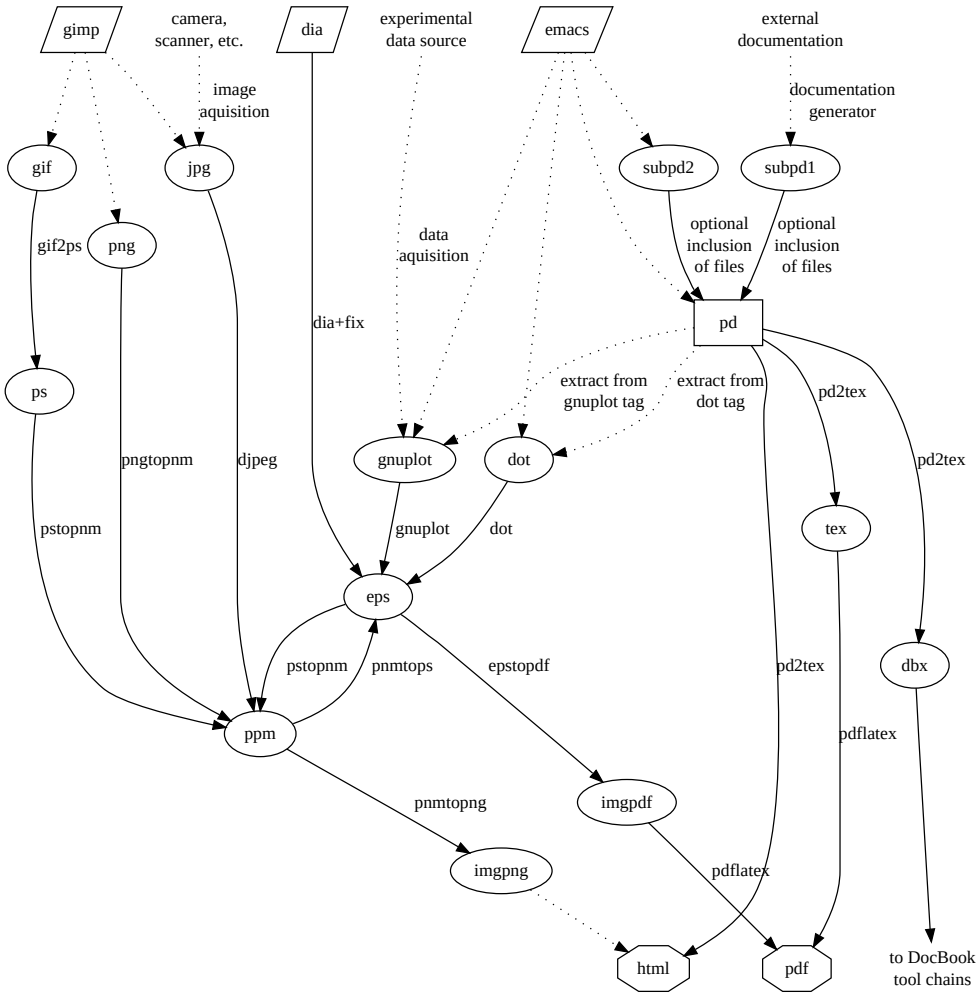


Figure 2: Data flow and image conversions

2 Invocation

acroread@\emphacro
xpdf@\emphxpdf()

First time, run `pd2tex -init` to generate directory hierarchy.

Then, usually all you need to do is

```
pd2tex your-doc.pd
```

This will generate a `tex/your-doc.pdf` file that you can view with *acroread(1)* or *xpdf(1)*. It also generates the `html/your-doc.html`, `tex/your-doc.dbx`, `tex/your-doc.rtf`, and `tex/your-doc.nonl` versions of the document. If the document contains images, automatic steps are taken to convert them to `.pdf` and `.png` formats as needed by the documents.

For full option listing, please try `pd2tex -h` which produces (you should still run it to see what options *your* copy of pd2tex supports):

```
Usage: pd2tex mydoc.pd  # Generate mydoc.tex, mydoc.pdf, mydoc.html, etc.
       pd2tex -acroread mydoc.pd  # Regenerate document and preview it
       pd2tex <mydox.pd >mydoc.tex       # filter mode
       pd2tex -dbx <mydoc.pd >mydoc.dbx  # filter mode for DocBook
Options:
  -dbx         Invokes DocBook filter mode
  -html        Invokes HTML filter mode (must make subdirectory html)
  -gensafe     Convert images from ps, eps, dot, or dia to pdf only if no pdf
  -gendep      Convert from ps, eps, dot, or dia to pdf based on time stamps
  -genforce    Force conversion of images from ps, eps, dot, or dia to pdf
  -nogen       Prevent conversion of images from ps, eps, dot, or dia to pdf
  -notex       Prevent .tex output in normal mode. Also prevents .pdf output.
  -nopdf       Prevent .pdf output in normal mode (.tex is still generated).
  -nodbx       Prevent .dbx output in normal mode
  -nohtml      Prevent .html output in normal mode
  -pdfonly     Only generate .pdf output. Do not attempt to generate HTML.
  -p           Shorter synonym of the above (only generate .pdf output)
  -fn          Omit footnotes.
  -FN          Force footnotes even on dbx
  -n           Dry run. Do not alter files on disk.
  -acroread    Automatically launch acroread after processing the document
  -d DIR       Change current working directory to DIR
  -o path      Specify output path different from input
  -DMACRO=VAL Define a macro to have a value
  -init        Create typical directory hierarchy used by pd2tex (tex, ...)
```

3 Syntax

PlainDoc

I recommend you just start writing as if you were writing a plain text email. Then come back here and see you how can apply some formatting. Best way really is to learn by doing (running pd2tex a million times in the process). Trying to learn the system before you start writing will just lead to frustration. To learn by example, view the source code of this document, available from
`http://zxid.org/plaindoc/sampo-plaindoc.pd`
About the only important thing you should remember up front is

> Paragraph break is created by putting an empty line between paragraphs, i.e. single newline will not break paragraph - you need two.

3.1 Section structure

PlainDoc uses underlined titles to indicate section headers. Different types of underlining indicate different levels. Generally you should make the underlining same length as the section title text, but *pd2tex* actually allows for some slop so do not get overly worried about this.

```
Doc Title Underlining (in the beginning of the document)
########################################################

1 Major section or Chapter underlining
======================================

1.1 Minor section underlining
-----------------------------

1.1.1 Teeny section underlining
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1.1.1.1 Subsubsubsection
^^^^^^^^^^^^^^^^^^^^^^^^
```

Usually you will use section numbers in front of sections, but underlying document formatting system will assign the numbers sequentially anyway, ignoring your numbers. This means that any numbers in the `.pd` file are only for benefit of those who read or edit the `.pd` file. This also means that there is no particularly urgent need to renumber if you happen to add new sections or change order - the

PDF output will have the numbers sequential irrespective of whether you make them sequential in the `.pd`. LaTeX LaTeX

If you would like *pd2tex* NOT number sections automatically, then you should add near beginning of your document

```
<<pdflags: secnum=0>>
```

you may also find

```
<<pdflags: stripsecnum=0>>
```

useful as this allows you to control the section numbering manually.

The underlining scheme only works if the underline is at least four characters long and there is an empty line before the title. In some exceptional cases you need section titles shorter than that - or *pd2tex* gets confused for some other reason. In these situations you can use the following special forms

```
<<sec: Section title>>
<<subsec: Section title>>
<<subsubsec: Section title>>
<<subsubsubsec: Section title>>
```

N.B. Although the above look like tags, there is no closing tag. The section simply ends when another section of the same level starts.

N.B. The fourth layer (`1.1.1.1` Subsubsubsection) is only available for documents of style "book". For other document styles you may get LaTeX errors about `subsubsubsection` not being supported.[2]

The sectioning markers actually take a couple of optional extra arguments

```
<<sec:id:short title: Section title>>
```

The ID argument is used for internal references, such as *see* specifications and paginated HTML file names. By default the ID is formed from the text of the Section title by squashing certain special characters. You may want to choose the ID explicitly if you anticipate changing the section title and need a stable ID for your *see* references. Another reason to pick an ID is that your ID can be much shorter that the automatically made one.

The *short title* argument allows you to specify an alternate shortend section title that is used on the footers and headers as well as in the table of contents. This only works with LaTeX / PDF backend. You may want to pick a shorter title so the headers will format nicer.

[2] For books the ^^^ maps to subsubsection.

3.2 Paragraphs and text emphasis

formatter
LaTeX
formatter

A new paragraph is started by an empty line (or a paragraph ends in an empty line if you like). There is no special marker for this. A mere newline does not start a new paragraph: you need two newlines in sequence. This allows paragraph body text to be wrapped with simple newlines.[3] Note that the formatter will not respect the simple line breaks, it will still format the paragraph as a whole.

You can introduce some emphasis[4] formatting using special characters

```
*bold*
+italic+
~computeroutput~
[REF]
```

Sometimes your document is so hairy that *pd2tex* gets confused in detecting whether star or plus really means emphasis (they could mean mathematical formula or even bulleted list). In these cases you can use following forms to disambiguate. One particular case where this is necessary is when you want to simply make just *one* character italic or computer output.

```
<<bold: your bold text>>
<<italic: your italic text>>
<<tt: your computer text>>
```

If you are aiming only at using the LaTeX based formatter, you can also access the TeX math mode using dollar signs:

```
Einstein's famous formula, $E=mc^2$, is very simple...
```

3.2.1 Verbatim text

If you want to create a bigger block of verbatim text, just indent it by two spaces more than surrounding document (this technique is used to generate most of the inset monospaced (Courier) blocks in this document, such as the one that follows).

[3] The Unix or emacs tradition is to explicitly wrap the paragraphs by inserting single newlines to keep lines about 70 characters long. However, pd2tex does not require this: you can keep entire paragraph as one line, like Mac or Word users would, as long as there are two newlines between paragraphs.

[4] Some document formatting systems and typographers are very dogmatic about what is "emphasis". pd2tex tries to subvert them as best as it can to make sure star gives bold (whether it's considered emphasis or not) and plus gives italic (whether that is emphasis or not). Usually one or the other will map to the underlying system's notion of emphasis and the other is created through explicit manipulation of fonts.

```
And the listing follows

  function foo(bar) {
    a = bar;
    return a+3;
  }

As can be seen, the code is trivial.
```

For formal specification writing you may want to use special tag `schema`

```
<<schema:
<xs:element name="TITLE">
  <xs:complexType mixed="true">
    <xs:attributeGroup ref="cb:typeAttributes"/>
  </xs:complexType>
</xs:element>
>>
```

Usually this produces just verbatim output, but may allow some automated processing on the schema.

Similar `code`, `ccode`, `console`, `diffout`, `email`, and `logoutput` exist for illustrating program code, c code (also php, perl, C++, Java) in particular, console output, diffs, and logs respectively. All these forms of verbatim output may eventually evolve to support some form of syntax highlighting.

3.2.2 Block quotes

To create an indented block quote, you start each line of the quote by a greater than symbol, in a manner to quoting in email or Usenet (news) posting.

```
> Block quote example
> second line

> Second paragraph.
```

Would render as

> Block quote example second line
>
> Second paragraph.

As can be seen, the specific positions of single newlines within block quote are ignored: all of it is formatted as indented paragraph. If you want to create paragraph breaks in a block quote just follow the two newline rule.

3.2.3 Footnotes

sampo"@iki.fi

Footnotes are created using `footnote` tag, which may wrap to several lines.[5]

```
<<footnote: Example footnote>>
```

There are no special formatting requirements for the text of the footnote, except that you have to be careful about not confusing *pd2tex* about where the footnote ends.

3.3 URLs, email addresses, paths, and function names

Some constructs used by programming and web documentation have distinctive syntactical structure that is fairly easy to recognize and therefore is formatted specially.

Email addresses are recognized by at character (@). For example

```
sampo@iki.fi
```

introduces an email address which is formatted using teletype font like this: `sampo@iki.fi`.

URL formatting is recognized by `://` somewhere near beginning of a string, e.g:

```
http://foo.bar/goo.htm?123
www.foo.bar/goo.html?123
foo.com/goo.html?123
iki.fi/goo.html?123
```

introduces an URL which is formatted using teletype font like this: `http://foo.bar/goo.htm?123` or like this `www.foo.bar` or like this for com-net-org domains `foo.com`, `bar.net`, `wee.org`, or like this for two letter country domains: `iki.fi`.

More examples: `www.foo.bar/goo.html?123` or like this for com-net-org domains `foo.com/goo.html?123`, `bar.net/goo.html?123`, `wee.org/goo.html?123`, or like this for two letter country domains: `iki.fi/goo.html?123`.

More examples: `www.foo.bar/goo.html` or like this for com-net-org domains `foo.com/goo.html`, `bar.net/goo.html`, `wee.org/goo.html`, or like this for two letter country domains: `iki.fi/goo.html`.

However, some well known file extensions are recognized separately. For example `foo.pl` is not a URL in Poland, but rather a file with extension `.pl` (as in

[5] Example footnote

perl(1) script). Similar exceptions apply to `foo.cc` and `foo.hh` which are common extensions for C++ source code.

perl@\emphperl()
function@\emphfunct
fork@\emphfork()
strlen@\emphstrlen()
proce_dure@\emphp

Presence of slash anywhere in a string or presence of dot in middle of a string cause the string to be considered a filesystem path and to be formatted using teletype font. Examples:

```
foo.ext
/foo
foo/bar
foo/bar.ext
foo/wee/bar
foo/wee/bar.ext
foo/
.ext
```

would format as `foo.ext` or `/foo` or `foo/bar` or `foo/bar.ext` or `foo/wee/bar` or `foo/wee/bar.ext` or `foo/` or `.ext`.

Dotted quad format IP addresses are recognized. There are some provisions for wildcarding or indicating the netmask. Following should work

```
192.168.1.*
192.168.1.0/24
192.168.1.1
```

and format as `192.168.1.*`, `192.168.1.0/24`, or `192.168.1.1`.

Uniform resource names are recognized, if they start by `urn` and colon, like `urn:liberty:foo`

For benefit of documenting XML, structures like `<tag>` are recognized and rendered as computer output.

Following an old Unix convention of suffixing function names and manual page entries with parentheses, like this

```
function()
fork(2)
strlen(3)
proce_dure(a,b,c)
```

would format as *function()* or *fork(2)* or *strlen(3)* or *proce_dure(a,b,c)*.

The PlainDoc formatter recognizes these structures and formats them using *italic* font. In this context the underscore character looses its special meaning (i.e. LaTeX math mode subscript command). PlainDoc formatter LaTeX formatter PlainDoc

You can prevent the automatic formatting from happening by wrapping the text in `e`-tag, like:

```
<<e: and/or>>
```

If you do not want automatic formatting to happen under any circumstances, you can specify:

```
<<pdflags: autoformat=0>>
```

3.4 Bulleted and numbered lists

Bulleted lists are started by including on left edge a bullet character and a space and then providing the text for the list item. If text wraps to two or more lines, you need to indent the subsequent lines by as much as the beginning of the text on the bullet line. Top level list can only start after an empty line (this is to avoid misdetection of bullet characters appearing as first character of a line in an ordinary paragraph).

Numbered lists work similar to bulleted lists: you simply start the line with a number and a dot and a space and follow the text for the list item, indenting correctly if it wraps. Instead of arabic numerals, you can also use letters. The actual numbering of the ordinal list items is done automatically by the underlying formatter, so the numbers that you provide do not matter (but you must provide a number for *pd2tex* to understand that you are creating an ordered list), they are only for your own reference - or reference of those who want to view your document in the plain text format.

Description lists are introduced with a double colon. The text before the double colon is the description title and the text that follows is the description body. The body can be wrapped to multiple lines, but you need to indent the subsequent lines by four spaces.

PlainDoc supports arbitrary nesting of lists of different types. Also verbatim code and certain other constructs can be nested in lists.[6]

[6] You are not supposed to type | or :, they are only used to illustrate alignment of indentation.

```
Lists and indent illustration  (| = current indent, : = parent's indent;
                                lesser indent terminates construct)

1.: parent list
  :a.|same level first (starts sublist)
  :b.|same level second
  :  |* subsublist first
  :  |* subsublist second
  :c.|same level third (terminates above subsublist)
  :  |* new subsublist
2.: next parent item (terminates above sublist)
```

PlainDoc

Lists and indent illustration (| = current indent, : = parent's indent; lesser indent terminates construct)

1. parent list
   a. same level first (starts sublist)
   b. same level second
      - subsublist first
      - subsublist second
   c. same level third (terminates above subsublist)
      - new subsublist
2. next parent item (terminates above sublist)

3.5 Tables

PlainDoc tables are formatted by having column headers underlined with equals signs and then supplying the table data in the columns. Use space characters for alignment and formatting.

```
<<table: example caption
Header1   Header2  Header3
========= ======== =========
row1col1  row1col2 row1col3
row2col1  row2col2 row2col3 last col overflowing
row3col1  row3col2 row3col3
```

```
row4col1 n.b. empty line starts "row mode" table where each line
row4col2 represents a cell and the amount of text in each cell
row4col3 can exceed the width of the column (wraps to multiple lines)

row5col1
row5col2
row5col3
:
row6col1  row6col2 row6col3
>>
```

This renders as (may appear on separate page due to underlying formatter's float placement algorithm): see table 3.

Table 3: example caption

Header1	Header2	Header3
row1col1	row1col2	row1col3
row2col1	row2col2	row2col3 last col overflowing
row3col1	row3col2	row3col3
row4col1 n.b. empty line starts "row mode" table where each line	row4col2 represents a cell and the amount of text in each cell	row4col3 can exceed the width of the column (wraps to multiple lines)
row5col1	row5col2	row5col3
row6col1	row6col2	row6col3

Also `longtable` keyword can be used. That will cause the table to be split across several pages (if it's long enough).

`minitable` keyword causes the table, which should not be big, to be placed inset in the text, i.e. the text will wrap around the table.

Column widths are controlled by the number of equals signs under the table header. They are NOT computed automatically. You can tweak the table by adding

formatter
LaTeX

or deleting equals signs. The amount of space per equals sign is controlled by `$tex_col_wid_factor` and `$dbx_col_wid_factor` in *pd2tex* source code. Rather than tweaking these factors, you are encouraged to experiment and iterate the number of equals signs in your document until you are happy. Eventually you will gain insight as to what is a good number of equals signs.

When composing a table, you usually horizontally align the columns. This means that the text MUST fit under the column header. However, sometimes it would be better if the text wrapped to multiple lines instead of forcing the column very wide. For the last column of the table this is accomplished simply by letting the text run off the right edge. However, for the other columns, you need a different trick:

Col1	Col2
Abc	This is minitable row 1
Def	This is 2nd row

Table 4: Minitable caption

> If an empty line is encountered in a table definition, the next row is described by having one column per line. The number of lines you supply must match exactly the number of columns in the table. Otherwise *pd2tex* will get confused and misformat your table - and quite often most of the rest of the document.

The table facility is not fully flexible,[7] but gets the job done for most simple and medium cases. If you really need a complex table, you will need to use `tex` or `dbx` tag to insert directly your formatter dependent code.

If the line immediately following the equals signs, has keyword `WIDTHS:` followed by comma separated list of numbers, then these numbers are used for table column widths. An empty specification leaves the column width as specified by the equals signs. A plain number specifies the width as absolute millimeters. A number prefixed by plus or minus sign makes the column that much wider or narrower, respectively.

If line immediately following the equals signs has keyword `OPTIONS:` then the rest of the line is parsed for table options. The first option specifies the reference tag for the table (e.g. for use in a `see` specification).

If a table does not fit on one page, consider using `longtable` keyword, which requires long table support at LaTeX level as well. If you do not want table to float, you can use `rawtable` keyword.

3.5.1 Comma Separated Values tables

Many spreadsheet programs allow exporting or saving the spreadsheet in `.csv` format, i.e. as comma separated values. Such file can be imported to Plaindoc

[7] This is by design to keep tables reasonably simple and easy to use for common cases.

document as a table using `csv` tag:

```
<<csv: file,topleft,botright,options: Legenda>>
```

where *topleft* and *botright* are specifications of cell in letter-number syntax customary to spreadsheets. For example, given following `demo.csv`

```
"Link","Worst 47M","Wishful 47M","Sum","Notes"
"====","=========","==========","===","====="
1,123,456,579,"foo"
2,222,333,555,"bar"
3,66,77,88,"rof"
```

specifying

```
<<csv: demo,B1,D4: Example>>
```

should result in

Table 5: Example

Worst 47M	Wishful 47M	Sum
123	456	579
222	333	555

and actually results in

The first row is always assumed to contain column titles. The equals signs row is necessary and is used to determine column widths in the output. Any further rows are considered to be normal data.

The default separator is comma. Thus comma should not exist anywhere in the values. The double quotes are not sufficient to protect commas in values. If you need to use comma in the values, then you need to use some other character as separator. Currently only other separator available is the pipey symbol "|". To use it, you need to specify pipeysep as option, e.g:

```
<<csv: file,B1,D2,pipeysep: Example>>
```

3.6 Multicolumn Blocks

LaTeX
LaTeX
LaTeX

Sometimes you need several columns, but a table is not the right choice or the LaTeX document type can not be twisted to do this automatically. A typical example would be two address labels side by side. For these situations you can use:

```
<<multicolstart: 40mm>>
Col1 text that is supposed to wrap around.

More col1 text in a new paragraph.
<<multicolnext: 40mm>>
Col2 text that is supposed to wrap around.
<<multicolnext: 40mm>>
Col3 text that is supposed to wrap around.

More col3 text in a new paragraph.
<<multicolend: >>
```

Which will render as

Col1 text that is supposed to wrap around. More col1 text in a new paragraph.	Col2 text that is supposed to wrap around.	Col3 text that is supposed to wrap around. More col3 text in a new paragraph.

The argument to the `multicolstart` and `multicolnext` is the width of the column that follows. As you can see, the plain text version has the columns linearily one after another, but the LaTeX / PDF and HTML renderings will place the columns side by side.

Multicolumn can not span a page break. If you need true multicolumned article format, you should investigate appropriate LaTeX styles and packages.

3.7 Images

You can include any general image using the following constructs. The image will be converted to `.pdf` (with `.eps` intermediary, unless it's already in one of these formats).

```
<<img: file: Legenda>>
<<img: file,posspec: Legenda>>
<<img: file,posspec,sizespec: Legenda>>
<<img: file,posspec,sizespec,trimspec: Legenda>>
```

where *posspec* is a LaTeX position spec. The *file* parameter specifies the file name *without any extension*. The extension is not relevant because *pd2tex* will automatically attempt conversion from a variety of file formats. If the automatic conversion fails, you may need to manually convert the image to `.pdf` format and place it in `tex/` subdirectory (where it would have been placed by the automatic conversion). LaTeX

Table 6: LaTeX position specs (with extensions)

Spec	Meaning
!	Try harder
H	Here, forces image here
h	here (if only spec, forces image here)
b	botton
t	top
p	floats page
!hp	Try hard here or floats page
Www	Wrap text around figure. Figure width ww cm.
R	Raw. Do not use float. Must leave caption empty.
*	Causes figure* to be used, as may be needed in twocolumn documents.

sizespec can come in two variants: either as symbolic or as hard coordinates. LaTeX

Table 7: Size specs

Spec	Meaning
wXh	Hard absolute width by height (both can have units)
2cmX3cm	2 by 3 cm
th	LaTeX Text Height (can also be used as unit)
tw	LaTeX Text Width (can also be used as unit)
1twX1thS	S is the stretch flag
n	Natural, size taken from image itself (no forced resize)
1	The default, corresponds to 1twX1th
15	1.5, 67% size
2	Half size (50%)
3	Third size (33%)
4	Quarter size (25%)
8	Eigth size (12.5%)
80	0% size

trimspec permits image to be cropped. It has format

```
L1B2R3T4
```

where first number specifies number of points to trim from left, second number specifies the points to trim from bottom, the third number specifies the points to trim on right, and the fourth number specifies how much to trim from top. Use this option for cropping badly behaving eps images (e.g. if original image is missing bounding box and ends up occupying a whole page).

If you are frustrated with LaTeX floats going all over the place, try

```
<<img: foo.png,R,n: >>
```

This causes Raw positioning (without float) and uses "natural" image size, i.e. whatever the original size of the image is, without any attempt to squeeze or stretch the image. Note that if you use R, you MUST NOT supply caption. If you use this approach and are not happy with image size, you should edit your image in your favorite image editor (this exercise may make you eventually appreciate the built-in scaling features).

3.7.1 Dia diagrams with layers

dia@\emphdia()
PlainDoc
dia@\emphdia()

Often its convenient to prepare a diagram with multiple overlays to illustrate multiple aspects of the same topic. In *dia(1)* this is usually done by creating the overlays as layers and then controlling the visibility of the layers when exporting the image.

To make this task easier, PlainDoc supports specification of the layers using special tag:

```
<<dia: file,posspec,sizespec,trimspec:layer1,layer2: Legenda>>
```

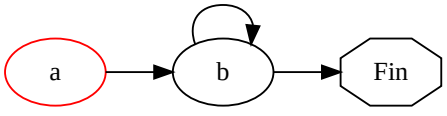
This is almost the same tag as the `img`, however with the twist that layers are specified between first and second colon. Use comma to separate layer names if you have multiple. See the above section on images for description of other specs.

3.7.2 Double images

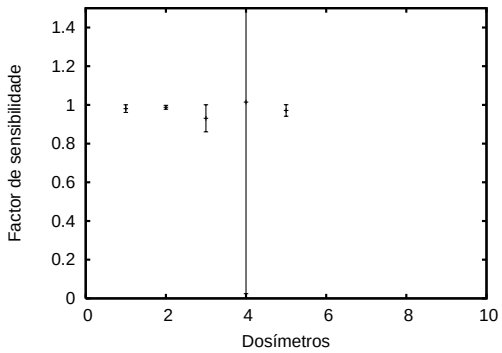
You can create two side-by-side images with

```
<<doubleimg: ref-tag,posspec: Text for legend
image-file1: Sublegend for image 1 (will be labelled a)
image-file2: Sublegend for image 2 (will be labelled b)
>>
```

For example, using our graph and diagram we could produce Fig-3



(a) Sublegend for image 1



(b) Sublegend for image 2

Figure 3: Text for legend

For *dia(1)* diagrams you can use

```
<<doubledia: ref-tag,posspec: Text for legend
image-file1:layer,other: Sublegend for image 1 (will be labelled a)
image-file2:layer,another: Sublegend for image 2 (will be labelled b)
>>
```

gnuplot@\emphgnupl

3.7.3 gnuplot diagrams

You can create *gnuplot* diagrams as normal images. *pd2tex* has support to automatically invoke *gnuplot* if there is a file whose name corresponds to missing image and ends in the extension `.gnuplot`. The file must contain *gnuplot* commands, but due to *gnuplot*'s ability process inline data (file name `'-'` in plot command), can also contain the data itself.

Another way to create a gnuplot diagram is using gnuplot directive and include the gnuplot commands and data inline in your `.pd` file. For example:

```
<<gnuplot: name-for-diag,,2: Legend for gnuplot diagram.
set terminal postscript eps lw 3.0 24
set nokey
set xlabel "Dosímetros"
set ylabel "Factor de sensibilidade"
plot [0:10] [0:1.5] '-' using 1:($2/14.57) with errorbars
# media 14.57, desvio padrao 0.98
# num   Valor   Normalizado (mal, com f=med/N, deve ser f=N/med)
1 14.29 .02
2 14.39 .01
3 13.56 .07
4 14.78 .99
5 14.15 .03
e
>>
```

Note how `'-'` was specified to include the data inline and last line is `e` to indicate the end of the data. Your data SHOULD start with `set terminal postscript eps` stanza[8]. If this line is missing, it will be supplied with one using default arguments. If you do not want to use Latin 1 (ISO-8889-1) encoding, you should specify the desired encoding on the first line. See *gnuplot(1)* documentation for further information. The above would create output in Fig-4.

[8]Optional additional arguments that may control font size and line thickness. You may want to specify these if you plan to reduce the image size significantly.

dot@\emphdot()
dot@\emphdot()
dot@\emphdot()
dot@\emphdot()

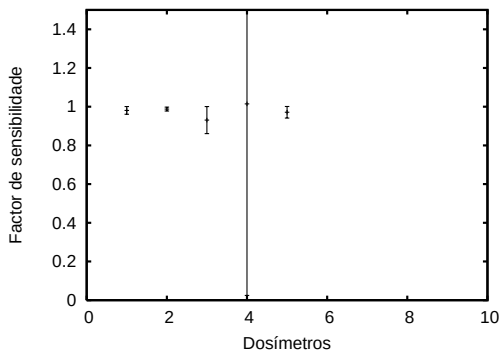


Figure 4: Legend for gnuplot diagram.

3.7.4 GraphViz or dot graphs

You can create *dot(1)* diagrams as normal images. *pd2tex* has support to automatically invoke *dot(1)* if there is a file whose name corresponds to missing image and ends in the extension `.dot`. The file must contain a description of a graph in *dot(1)* format.

Another way to create a dot diagram is using dot directive and include the dot graph inline in your `.pd` file. For example:

```
<<dot: name-for-graph,,2: Legend for dot graph.
digraph states {
  rankdir=LR;
  a -> b -> c;
  b -> b [style=dotted];
  a [color=red];
  c [shape=octagon,style=filled,fillcolor=blue,label="Fin"]
}
>>
```

See *dot(1)* documentation or `http://www.graphviz.org/` for further information. The above would create output in Fig-5.

3.8 Bibliographies

You make bibliographical references using square brackets:

```
...as described in [RFC2739].
```

perl@\emphperl()
PlainDoc

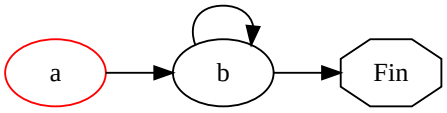


Figure 5: Legend for dot graph.

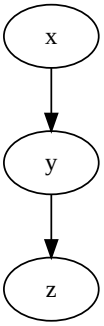


Figure 6: Vertical graph

In the end of the document you create the bibliography section with `references` tag:

```
<<references: Reference section title
[RFC2739] T. Small, D. Hennessy, F. Dawson: "Calendar Attributes
          for vCard and LDAP", RFC2739, IETF, 2000.

[vCard21] Internet Mail Consortium, "vCard - The Electronic
          Business Card Version 2.1",
          http://www.imc.org/pdi/vcard-21.txt, September 18, 1996.
>>
```

In the references section you describe the references. You start a reference by the bracketed tag that was used in the text to refer to it and follow that by description of the reference. No special structure exists for the description.

If you want to use structured database to keep and format your descriptions, you can write a *perl(1)* program to generate the references in the format you like from the database and use the PlainDoc inclusion facilities to bring them into your document.

It is possible to have more than one bibliography, simply use different title for them, e.g. "Normative" vs. "Informative". If you do not supply any title, the default title of the underlying formatting system is used.

3.9 Referencing Sections, Tables and Figures

Its fairly common for a document to reference a figure, e.g. "see Fig-1.2". However, since sections, tables, and figures are automatically renumbered as needed, you can't safely just hard code a number in the document. Instead you should use the `see` construct

```
<<see: Section_title>>
<<see: fig:figure_name>>
<<see: table:table_name>>
```

The identifier for a section is derived from the section title by substituting all problematic characters with an underscore. For example, see 3.2 or see Syntax section 3.2.

The identifier for a figure is derived from the figure file name by substituting all problematic characters with an underscore. Figure identifier is always prefixed by `fig:` prefix.

The identifier for a table is derived from OPTIONS specification within the table - if there was no OPTIONS spec, then the table is unreferenceable. The table identifier is always prefixed by `table:` prefix.

3.10 Creating an Index

To enable index, you must include somewhere in your document

```
< <makeindex: 1>>
```

This triggers index generation and will insert a section containing the index.

Creating index involves marking the words to be indexed with `ix` construct, like this:

```
<<ix: Dickens>> said that...
```

All bibliographical references, function names, path names, URLs, and email addresses are automatically included in the index. You can also specify words, concepts, and people indexes as follows

```
< <wordix:
word or phrase
word
word
>>
< <conceptix:
concept 1
concept 2
>>
< <peopleix:
John Q. Public
AD Brown
>>
```

LaTeX
PlainDoc

In general all of the above accept one indexable phrase per line and then make great effort to detect occurrences of said phrase in text of the document. This in general will avoid cluttering most of the text with `ix` declarations, but has the disadvantage that even the irrelevant mention of the phrase will get indexed. Also, there is no easy way of indicating the most relevant index entry.

Indexing currently only works with LaTeX backend.

3.11 Document Preamble

Usually you start PlainDoc documents with a preamble that controls formatting template and provides metadata like revision control and authorship information. All these tags are optional and have reasonable defaults. (In the following, the two starting angle brakets are separated by space to prevent interpretation when rendering this document. In your own document you would omit the space.)

```
#./pd2tex    # -*-pd-*-
Document Title
##############
< <class: article_or_book!options!language!header_title!after_page!moreopt>>
< <cvsid: $Id: sampo-plaindoc.pd,v 1.32 2009-11-10 22:44:17 sampo Exp $>>
< <version: 1.0-05>>
< <author: doc author>>
< <credit: Credit title
John Public, Acme Corporation
Joe Doe, Sample, Inc. >>
< <history:0: revision history title
```

```
08:: 17.5.2004,  Sampo Kellomäki (sampo@symlabs.com)
    * changed this
    * edited that
09:: 20.8.2004,  Sampo Kellomäki (sampo@symlabs.com)
    * more edits
>>
< <abstract: ...>>
```

PlainDoc
class
LaTeX
class
LaTeX
class
LaTeX
class
class

The first line that starts with the hash character is an optional comment that identifies the file as PlainDoc file. If you have emacs `pd-mode` installed, it will automatically be switched on.

class The `class` tag takes as an argument a string which can be divided into up to 6 parts separated by exclamation marks. The first part is the LaTeX document class name.

The second part is for optional arguments to LaTeX document class. This is typically used to specify paper size and point size of main font.

The third part are optional arguments to pass to LaTeX *babel* package that deals with language specifics. Usually you would pass the ISO language code (e.g. "pt" for portugues). The default is english.

The fourth part is an optional string to be included in footer or header of your document. Usually it would be abbreviated identification of the document, or perhaps your name. The exact way how this gets used will depend on the format template.

The fifth part is also optional. Some format templates display it after page number, thus permitting you to create effects like "page 5 of 37".

The sixth parameter, which is optional, can supply additional options. Currently defined options include

lineno Turns on line numbering (at least in `tex/pdf` output)

In absence of `class` tag, the default document class is `article`.

cvsid Intended to hold revision control identifier, usually used for CVS Id tag.

version Allows version of the document to be formally declared. Typically this is the externally visible version designation and most of the time this has nothing to do with `cvsid`.

inclusion facilty PlainDoc

author Indicates document author, and often email, too. The author information is used to generate the title page. There is no special formatting for author information, but if you include an email address, you may want to put it in parentheses rather than the customary angle brackets to avoid confusion about where the tag ends.

credit Indicates other (minor) authors or people who should be given credit for the work. The string on the tag line will be used as title of the credits section. All subsequent lines describe the worthy contributors, one per line. It is customary to separate the company name by a comma.

history Change log of the document. The string on the tag line specifies the title of the change log and rest of the tag is formatted as description list with bulleted sub lists. Usually the description title (the part before double colon (::)) is the revision number of the document. This is followed, on the same line, by date and editor, separated by a comma. All subsequent lines should be formatted as single level bulleted list, one list item per line (i.e. wrapping lines does not work). The bulleted items must be indented by exactly four spaces because it is a sublist of the description list (see list below).

You may have a change log in CVS. If you want to use that, I suggest you write a perl script that extracts it from cvs and formats it according to the conventions of the `history` tag and then just use the file inclusion facility to bring it in. I.e. we do not support this very well yet, patches welcome.

abstract Used for short description about the document, usually abstract of a scientific paper. No special formatting requirements.

See also `moretexpreamble`, `texpreamble`, `dbxpreamble`, `additionalarticleinfodbx`, and `htmlpreamble`.

3.12 Including other files into document

File inclusion facilty of PlainDoc is a very powerful way to assemble large documents from smaller bits and pieces. Typically you would have one `.pd` file for each chapter and then a *master document* that pulls them all together.

To include a file you simply enclose its name in double angle brackets (n.b. we had to insert a space between the angle brackets to prevent their special interpretation here).

```
< <path.ext> >
< <includerange: path.ext: start-end> >
```

The `includerange` tag allows you to include only selected lines from the other file. Line numbers are zero based (i.e. first line is 0) and both must be specified, however it's ok for the end to be out of range, e.g. use 9999 to include everything until the end of the file.

Generally all includes are processed in a special preprocessing step before other tags and formatting are processed.

3.13 Other special formatting

```
(*** TODO items)
< <ignore: comments out a block> >
```

Todo items - expressed as opening parentheses, three stars, some text and a closing parentheses - do not appear in formatted document. They allow editor to add notes and comments where she needs to revisit something.

The `ignore` tag allows you to "comment out" sections of the document. Ignore blocks do not appear in the formatted output - this is a bit difficult to illustrate. For commenting out really large sections, it may be easier to use `<<if: 0>>` blocks, see below in "Conditional processing" section.

3.13.1 Passing comments to backend

```
< <comment: Your comment here >>
```

will produce in HTML and DBX output

```
<!-- Your comment here -->
```

and in TEX output

```
; Your comment here
```

The difference between `ignore` and `comment` is that the former prevents the text from reaching the backend at all while the latter will pass the text to the backend, but use the backend's comment syntax to escape it (so it will typically not render even if it is in the file).

> N.B. If you want to pass comments only to a specific type of backend, you can use the backend specific tag, such as `< <html: <!- HTML only comment ->` » `< <tex: ; TEX only comment >>`

3.14 Special support for grammars

You can include fragments from a schema grammar file as figures with

```
<<sgfrag:sgfile:yoursection:xsdfile.xsd: Caption>>
```

The *sgfile* specifies the name of the file without the `.sg` extension.
The *yoursection* looks for

```
#sec(yoursection)
foo
#endsec(yoursection)
```

inside the schema grammar file and extracts the content (*foo* in this case).
The *xsdfile.xsd* specifies optional xsd file (see below).
The *Caption* is the caption for the resulting figure.
If you want to render schema grammar fragments as underlying xsd, you can specify

```
<<pdflags: showsgasxsd=0>    Display schema grammar as schema grammar. The defa
<<pdflags: showsgasxsd=1>    Includes the XSD file using DocBook or XML include
<<pdflags: showsgasxsd=2>    Inlines the contents of the XSD file
```

3.15 Outputting verbatim blocks as files

Sometimes you want to keep some schema fragments inline in document, but would like to output them as files for other mechanized processing as well. For this you should use `schema`, `code`, or `logoutput` tag with optional file argument as follows:

```
< <schema:filepath: verbatim data
more data
>>
```

3.16 DocBook only

```
< <dbxpreamble: > >
< <additionalarticleinfodbx: > >
< <dbx: > >
```

N.B. This section may be illegible in some output formats. Please consult the original `sampo-plaindoc.pd`

3.17 HTML only

```
< <htmlpreamble: > >
< <html: > >
```

N.B. This section may be illegible in some output formats. Please consult the original `sampo-plaindoc.pd`

You can also create hyper links using,

```
<<link:url: Text>>
```

For example: ZXID. The URL itself may contain colon (e.g. as in http://...), only colon followed by a space starts the text. If no text is supplied, the URL itself is used as text. For example symlabs.com. There can not be space after first colon and there MUST be a space after second colon.

3.17.1 Multipage HTML

Multipage HTML allows each section, subsection, etc., to become a file by itself. The file name is generally formed from document base name and the section label that corresponds to the file.

The HTML headers and footers for the files can be specified with

```
< <htmlpreamble2: > >
< <htmlpostamble2: > >
```

The pre and postambles can be customized by using bang bang (!!) macros

TITLE Page title, composed of section number and title of the section

BASE The document base name

PREV Link to page of previous section in navigation order

NEXT Link to page of next section in navigation order.[9]

[9]Currently NEXT does not work in preamble. This bug will be fixed some time in future.

3.17.2 HTML Info Boxes

HTML infobox is a HTML table that can be visualized or hidden using JavaScript. It is convenient means of saving real estate on page, while still including text in an easily accessible form.

```
< <infobox:id:link:tableargs: Content> >
```

id Is HTML object ID that is used in JavaScript manipulations to refer to the box

link The link text for visualizing the box

tableargs Additional arguments for the `<table>` tag, usually used to control width, alignment, and style.

Content Any content to be displayed. Raw HTML.

Example

```
<<infobox:blognav:Show Navigation:align=right width=300:
*Navigation*

* <<link: >>
* Administrative issues
  - <<link: >>
>>
```

3.18 TeX only

```
< <texpreamble: > >
< <moretexpreamble: > >
< <tex: > >
< <eqn: > >
< <1stpage: > >
```

3.19 Conditional processing

Plaindoc supports conditional processing using

PlainDoc

```
< <if: MACRO> >
foo
< <else: > >
bar
< <fi: > >
```

where the MACRO is a defined either with < <define1st: MACRO!VAL> > construct or is passed as -DMACRO=val command line flag (n.b. the usual !! in front of MACRO name is not used). The *else* block is mandatory (but can be empty). Macros defined using < <define: MACRO!VAL> > construct can only be processed after first expansion of includes and conditional processing - since this is so confusing, you should use define1st or -D method.

3.20 Seals to protect against tampering

A common problem in signing legal documents is the possibility that they might be tampered with afterwards to make it seem like something was signed where as in reality the document did not have such clause at the time.

PlainDoc provides a document *checksum* in form of pdseal construct. Place near the end of the document

```
< <pdseal: > >
```

which will produce in the output something like

```
PDSEAL1VJgudYOTsJWgnWhGQK495wBthTcN
```

This checksum is computed over "visible" characters of the document. Validity of paper document can be checked against pdseal by entering the visible characters in sequence following a formatting convention. Legal documents should be elaborated such that the visible character sequence of pdseal conveys the true meaning and intent of the document. Then run

```
pd2tex -verify <visible-chars-of-document
```

The normalization used for pdseal will collapse whitespace. Remove single newlines (double newlines signify paragraph breaks and are preserved, but multiple empty lines are collapsed to just one). Normalization will remove any indentation, which may make interpretting nested lists a bit more complicated - we suggest using different item numbering schemes to distinguish the levels of the list. You

have to enter list items with an empty line between each item, because otherwise `pdseal` will consider all items just as lines of a paragraph and run them together. Normalization will ignore bold, italic, and font size, so be sure the legal meaning of your document does not depend on these. LaTeX LaTeX

`pdseal` can not take in account several types of visible text

- Text inside special LaTeX constructs (`tex:` blocks)
- Text in headers and footers
- Text in footnotes (including footnote index numbers in main text)
- Text in images
- Text in table of content, list of figures, index, etc.
- Cross references using `see:` construct (in legal text it is safest to make cross references explicit)

Sometimes you need to edit the visible characters to account for these. Also, if title page is not rendered, (`maketitle: 0`), then the tags for title page items, like `author` should not have values.

It is unfortunate that such editing may needed and this may be seen as diminishing the credibility of the `pdseal`. Generally the line of thinking should be that first by editing and experimenting find the text that matches the `pdseal` and then argue about the differences in the text that had to be edited. If no matching text can be found by editing, you can challenge the opponent to show what edits would make the text match `pdseal` and argue that if they find none, the text has been tampered with. Unfortunately opponent may also argue that the `pdseal` is buggy or too imprecisely defined for them to be able to guess what trivial formatting might be causing it to fail. Of course if you have the original unaltered text (even if not signed by opponent), it becomes easy to know what the differences are and then proceed to show that the original text matches the `pdseal` that opponent signed, rather than the version that opponent claims to be the document.

It has been noticed that when generating PDFs LaTeX may alter some characters to visually more pleasing form. These may include dashes and typographers quotes. If the seal is not verifying, you may need to edit the pasted text for these possibilities. Accented characters should use same character encoding as when document was prepared, i.e. if you wrote it in Unicode UTF-8, then the pasted text should be in UTF-8 as well. For european languages we recommend using ISO Latin1 as this is the encoding we test with.

The `cvssig` feature provides a way to connect the document to the version control checkout of the document sources. This is highly useful, but distinct from the checksum that `pdseal` provides.

PlainDoc
PlainDoc
PlainDoc
LaTeX

3.21 Summary of Special Characters and Their Meaning

PlainDoc works by giving some punctuation and special characters a special meaning. Usually these characters work in the normal way unless used in a special context. Generally you should not worry about them too much when editing documents, but if output shows that PlainDoc has indeed confused a punctuation character used in plain meaning with the special meaning, you may need to take some steps to disambiguate the meaning. Often this involves adding whitespace or some rearrangement, but in extreme cases you may need to recourse to some special PlainDoc syntax or LaTeX syntax.

```
!  -- No special meaning, reserved for punctuation in content
! ! -- Macro variable expansion (bangbang) (see <<define: var: value>>)
"just textual quoting"  -- no special meaning, but LaTeX will apply typographer
#  -- doc title underline, often comment in programming
$\gamma$                -- TeX math mode
%  -- TeX comment character
&  -- Special in HTML as it starts entity escape. Not special in other backend.
'  -- (single quote) No special meaning, reserved for punctuation in content.
(  -- causes preceding word (without space) to be considered a function name
)  -- No special meaning, reserved for punctuation in content.
*emph*   -- Bold emphasis
* bullet -- On left edge introduces a bulleted list item
+italic+ -- Italic emphasis
+ bullet -- On left edge introduces a bulleted list item
~teletype~  -- Teletype emphasis, use for "computer text" like
               variable names, etc. Also subsection underline.
,  -- No special meaning, used for punctuation in content.
- bullet -- On left edge introduces a bulleted list item, section underline
.  -- No special meaning, used for punctuation in content.
/  -- Causes detection of file path or URL
term:: definition  -- ("Four dots") Introduce definition list items
;  -- No special meaning, used for punctuation in content.
<< -- Starts PlainDoc tag
<  -- Starts highlighting text as XML tag. Usually this means computer output
=  -- Chapter title underline
```

```
-  -- section underline
~  -- subsection underline, also teletype emphasis
^  -- subsubsection underline
>  -- Ends XML tag highlight
>> -- Ends PlainDoc tag
?  -- No special meaning, reserved for punctuation in content.
@  -- No special meaning, but often indicates an email address
[Reference]  -- Also used in TeX macros for optional args
\  -- Invoke TeX macro, e.g. \newpage or \foo[optarg]{arg1}{arg2}
\\ -- Line break (<br> in HTML)
^  -- TeX math superscript, e.g. $E=mc^2$; subsubsection underline
_  -- TeX math subscript,   e.g. $H_2O$ or $H_{ref}$
`  -- No special meaning
{arg} -- TeX macro argument grouping
|
```

4 Producing Slides (presentations, overheads, transparencies, "powerpoints")

Generally your slide set (slidestack, slide stack, slidedeck, slide deck) will start with something like

```
My Presentation
###############
< <class: slide!12pt! !CUR-DAL Id Mgmt>>
< <author: Sampo Kellomäki (sampo@symlabs.com)>>
< <maketitle: 1>>

< <moretexpreamble:
\usepackage{pdfslide} \overlay{background.pdf}

\fancyhead{}
\fancyfoot{}
\fancyhead[R]{\tiny{\thepage}}

\fancyfoot[L]{\raisebox{-5mm}{\includegraphics[height=6mm,keepaspectratio]{logo
\fancyfoot[C]{\raisebox{-5mm}{\includegraphics[height=8mm,keepaspectratio]{logo
\fancyfoot[R]{\raisebox{-5mm}{\includegraphics[height=8mm,keepaspectratio]{logo
```

```
%\setlength{\footskip}{2\baselineskip}

> >
```

LaTeX
LaTeX
mpage@\emphmpage

This enables special page size and margins that are useful for creating slides. It also creates a page break after each section (there may be other page breaks if you have more material than will fit on one slide). Of course you can always add more page breaks by using

```
<<newpage: >>
```

construct.

The *moretexpreamble* stuff is direct LaTeX code that allows you fine control over headers, footers, and the background of your slides. Especially the overlay feature is great for getting the "corporate look" to your slides. If you do not understand what it does, you need to ask some LaTeX expert. One caveat: the `.pdf` files that you might use in *includegraphics* are relative to the `tex/` directory.

If you need to get just one or two more lines on page, you may find

```
<<tex: \enlargethispage*{2\baselineskip}>>
```

useful.

In slide mode, the sections and subsections are not numbered. If you want numbering, you should simply add the numbers manually.

You can include images and figures in your slides in a normal way. However, at times it may be useful to omit the legend from the figures. You can do this by supplying "0" (zero) as the legend.

To print the slides, reorder pages (mpage -j flags are buggy)

```
pstops 4:2,3,0,1 /tmp/foo.ps /tmp/0.ps
mpage -4 /tmp/0.ps | nc printer-ip-address 9100
```

The tricky part is getting the landscape slides ordered so they read naturally while most 4-up printing software (like *mpage(1)*) are geared towards portrait printing. If you print one, or even two, slides per page, this is not likely to be a problem. "Natural" two sided printing is left as an exercise to the reader.

5 Installing the tool chains

PlainDoc
perl@\emphperl()
perl@\emphperl()
LaTeX
gcc@\emphgcc()
binutils@\emphbinuti
glibc@\emphglibc()
pd2tex@\emphpd2tex

It's easiest if you get your PlainDoc system already compiled and installed by someone, but if you are familiar with building open source software, building all of your own tool chains is certainly feasible. The *pd2tex* itself is a *perl(1)* program so it does not need any compilation, but it depends on many other programs so you need to have them in order to have a "tool chain". In this chapter I explain how I built mine and try to give some tips.

In the very minimum you will need *perl(1)*. Generally perl comes with just about any Linux distribution and with most other Unixes so this is not a major obstacle. With perl only, you will be able to generate HTML output as well as `.dbx` and `.tex` intermediate files. To further process the latter two, you will need to install additional tools.

teTeX or texlive variant of LaTeX usually ships with Linux distributions and is easily obtained and installed for other Unixes. For Windows MikTeX is the best alternative. DocBook toolchains are not explained any further here: refer to your favorite web search.

Since a lot of information here depends on the particular versions of the software packages and is always in flux, you should expect some discrepancies when you actually build your own system. If my recipe does not work for you, please study the documentation (usually `INSTALL` and/or `README` files in the top directory of each software package's source code tree) and try to build it the way they recommend.

These recipes were created around Sept. 2004. You can expect that these instructions will be updated from time to time.[10]

N.B. *gcc(1)*, *binutils(1)*, and *glibc(3)* are probably only worth worrying about if you plan to build everything from sources.

The perl dependency is not very sensitive, because *pd2tex(1)* does not use any perl modules (except the ones that distribute as standard). While the development work happens currently (Apr 2006) on perl-5.8.4 system, no exotic features are used, so it should work with perl-5.6 and may even work with perl-5.003. I'm interested in patches to ensure backwards compatibility.

5.1 Preliminaries

Most of these preliminaries are likely to have already been satisfied by your linux distribution.

[10] Some updates have been made as of 20130411 based on Ubuntu.

Table 8: Software versions

Ware & Version	Web	How to check
perl-5.6.x	perl.org (perl-5.8.x also works)	which perl && perl –version
gnuplot-4.0.0	ftp://ftp.gnuplot.info/pub/gnuplot/ http://gnuplot.sourceforge.net	which gnuplot && gnuplot –version
graphviz-1.16	www.graphviz.org	which dot && dot -V
gs-8.53	www.ghostrscript.com	which gs && gs –version
dia-0.94sampo	http://www.gnome.org/projects/dia/	(version 0.96.1 also works)
gcc-3.4.2	gcc.gnu.org	which gcc && gcc –version
binutils-2.15.91.0.2	ftp.gnu.org	which ld && ld –version
glibc-2.3.3	ftp.gnu.org	ls -al /lib/libc-*.so

5.1.1 zlib-1.2.1

Nearly all Linux and Unix platforms ship with zlib, so usually this requirement is trivially satisfied. But if you need to compile it:
http://www.gzip.org/

```
./configure --prefix=/apps
make test
make install
```

5.2 gnuplot-4.0.0

Installing gnuplot is optional, unless you have data in gnuplot format or you wish to create some.

```
sudo apt-get install gnuplot    # Works on Ubuntu
```

- ftp://ftp.gnuplot.info/pub/gnuplot/
- http://gnuplot.sourceforge.net/

Needs zlib (see CPPFLAGS and LDFLAGS)

Gnuplot can be built with all sorts of options, but we really only need the Postscript/EPS output. Thus you should not worry about png, gif, or pdf libraries and their license entanglements.

First apply following patch (which has been submitted to the gnuplot team)
— datafile.c.orig 2005-01-20 04:28:09.051477624 -0500 +++ datafile.c 2005-01-20 04:32:09.821874960 -0500 @@ -570,6 +569,7 @@ /* now allocated dynamically */ int i; int name$_t$*oken*; +*staticlonginline*$_t$*ell*; /*∗*remember*fileposition*from*′* −*′* *to*′ =′ 200 50119*sampo@iki.fi* ∗/
TBOOLEAN duplication = FALSE; TBOOLEAN set$_i$*ndex* = *FALSE*, *set*$_e$*very* = *FALSE*, *set*$_t$*hru* = *FALSE*; @@ −729,6 +729,14 @@ *data*$_f$*p* = *lf*$_t$*op*(); *if*(!*data*$_f$*p*)*data*$_f$*p* = *stdin*; +*inline*$_t$*ell* = *ftell*(*data*$_f$*p*); /∗*remember*position*for*′ =′ 20050119 *sampo@iki.fi* ∗ / +*mixed*$_d$*ata*$_f$*p* = *TRUE*; /∗*don*′*tclosecommand*file∗ / +*elseif*(∗*df*$_f$*ilename* ==′ =′ *strlen*(*df*$_f$*ilename*) == 1) +*plotted*$_d$*ata*$_f$*rom*$_s$*tdin* = *TRUE*; +*data*$_f$*p* = *lf*$_t$*op*(); +*if*(!*data*$_f$*p*) + *data*
This patch is request id 1105717, submitted on 20.1.2005, into gnuplot patch tracking, `https://sourceforge.net/tracker/index.php`.
Optimization must be turned off due to bug in gnuplot mxtics feature when using time series data.

```
CPPFLAGS=-I/apps/include LDFLAGS=-L/apps/lib ./configure --prefix=/apps/gnuplot

./prepare   # does autoreconf && aclocal && autoconf && automake
CPPFLAGS=-I/apps/include CFLAGS=-g LDFLAGS=-L/apps/lib ./configure --prefix=/ap
make
make install
```

If you get error like

/apps/lib/libpng.so: undefined reference to ‘deflate’

you need to add `-lpng -z` as last options on the linking line (`cd src` and cut and paste the failed command, adding the flag).

5.3 dia-0.94patch

Installing dia (`http://www.gnome.org/projects/dia/`) is optional, unless you have diagrams in dia format or you wish to create some.

```
sudo apt-get install dia    # Install on Ubuntu (seems recent packages have my
```

Recent dia (dia-0.96-pre1 and newer) seem to have fixed the bugs, below.
For older versions of dia, please see on `http://bugzilla.gnome.org/` bugs

```
153606  Add --show-layers=LAYER,LAYER flag for automated export
153607  Pango fonts are crappy in Acroread, Latin 1 fonts are goo...
153609  Wrong (too small) text size in multiline text using PANGO...
```

The bug #153606 is most relevant for enabling automated exports. Bug #153607 may be relevant for european language uses. Bug #153609 contains an important patch to work around the problem (disabling font cache).

LaTeX LaTeX **LaTeX** LaTeX class LaTeX

5.4 teTeX or other LaTeX

You will need some sort of LaTeX system to generate PDFs. The teTeX-2.0.2 that ships with nearly every Linux distribution (as of 2005) is adequate. More recent Linuxes have texlive, which is good.

```
sudo apt-get install texlive-full          # Works on Ubuntu 12
```

Windows users should get MikTeX.

5.4.1 Additional LaTeX packages

Installing additional LaTeX packages is optional for most situations.

floatflt already included in teTeX-2.0.2, but sometimes missing on Ubuntu, see below.

lineno only needed if you want line numbers, needs installation and adding to preamble (`\usepackage{lineno} \linenumbers]`) or specifying `lineno` as *moreopts* in class.

longtable only needed for long table support

textpos only needed if you need arbitrary placement of text and graphics (needs install)

everyshi Required by textpos (already included in teTeX-2.0.2)

enumitem Control list spacing (optional)

listing Special formatting of program listings (think code tag)

Usually you install additional LaTeX packages (you can download them from `ctan.org`) as follows

```
cd /apps/teTeX/2.0.2/share/texmf/tex/latex
tar xvzf /t/textpos.tar.gz
```

The package directory should appear as immediate subdirectory of the `share/texmf/tex/latex` directory. LaTeX

```
mv tex-archive/macros/latex/contrib/textpos .
```

Sometimes you need to run installation script (see README, if any)

```
cd textpos
latex textpos.ins
```

Finally rebuild `ls-R` so that LaTeX will find the new packages:

```
cd /apps/teTeX/2.0.2/share/texmf
../../bin/i686-pc-linux-gnu/texhash
ls -alF /apps/teTeX/2.0.2/share/texmf/ls-R  # double check
```

5.4.2 Installing Myriad as main document font, `pmy.zip` + MyriadPro route

Installing additional fonts is optional and only needed in special circumstances.

Instructions given in `http://www.tug.org/tex-archive/fonts/metrics/w-a-schmidt/pmy.txt` work fine. You need to get `http://www.tug.org/tex-archive/fonts/metrics/w-a-schmidt/pmy`

The only problem is where to get the actual `.pfb` (and `.afm`) files. Presumably you would have to buy them from Adobe. I found MyriadPro from the net and did

```
cd /apps/teTeX/2.0.2/share/texmf/fonts/type1/adobe/myriad/
tar xvzf myriad-pro-pmy.pfb.tgz
```

The tar ball should expand to following files `pmyr8a.pfb`, `pmyri8a.pfb`, `pmyb8a.pfb`, `pmybi8a.pfb`, `pmyrd8a.pfb`, `pmyr8ac.pfb`, `pmyri8ac.pfb`, `pmys8ac.pfb`, `pmysi8ac.pfb`, `pmyb8ac.pfb`, and `pmybi8ac.pfb`.

Unfortunately MyriadPro was not supplied with `.afm` files so I just wholly omitted them and things seemed to work anyway.[11]

```
cd /apps/teTeX/2.0.2/share/texmf
unzip /t/pmy.zip
../../bin/i686-pc-linux-gnu/texhash
updmap --enable Map pmy.map
../../bin/i686-pc-linux-gnu/texhash
```

[11] Using lcdf-typetools it might be possible to generate the `.afm` file, but I have not investigated this yet.

After this just add to TeX preamble LaTeXCompanion

```
\usepackage[T1]{fontenc}
\renewcommand{\rmdefault}{pmy}
```

Voila, it works. See [LaTeXCompanion], p.339 for further ideas.
A way to autodetect this?

```
< <moretexpreamble:
  \IfFileExists{t1pmy.fd}{
       \renewcommand*{\sfdefault}{pmy}
  }{
       \renewcommand*{\sfdefault}{phv}
  }
  \AtBeginDocument{\sffamily}
> >
```

For further font investigations see lcdf-typetools-2.38 at `http://www.lcdf.org/type/`

5.4.3 Installing New Century School Book via fouriernc package

```
wget http://www.ctan.org/get/fonts/fouriernc.zip
cd /apps/teTeX/std/share/texmf/fonts/tfm/public
unzip  /t/fouriernc.zip
cd  ../../../vf/public/
ln -s ../../tfm/public/fouriernc
cd ../../../tex/latex/
ln -s ../../../texmf/fonts/tfm/public/fouriernc
mktexlsr
```

But unfortunately this depends on more packages: Fourier-GUTenberg

```
wget http://www.ctan.org/get/fonts/fourier-GUT.zip
cd /apps/teTeX/std/share/texmf
unzip  /t/fouriernc.zip
ln -s ../../fourier-GUT/tex/latex/fourier tex/latex/
ln -s ../../../fourier-GUT/fonts/tfm/public/fourier fonts/tfm/public
ln -s ../../../fourier-GUT/fonts/afm/public/fourier fonts/afm/public
ln -s ../../../fourier-GUT/fonts/map/public/fourier fonts/map/public
ln -s ../../../fourier-GUT/fonts/type1/public/fourier fonts/type1/public
ln -s ../../../fourier-GUT/fonts/vf/public/fourier fonts/vf/public
updmap --enable Map fourier.map
mktexlsr
```

Finally in packages

```
\RequirePackage{fouriernc}
\renewcommand{\familydefault}{\rmdefault}
```

or in your document

```
\usepackage{fouriernc}
```

5.4.4 Changing Document Main Font

```
\renewcommand{\familydefault}{\sfdefault}
```

5.4.5 Finnish hyphenation

Just specifying the *finnish* language in the document preamble is not enough. You actually need to install the hyphenation patterns as well.
Get `fi8hyph.tex` from `http://tug.ctan.org/tex-archive/language/hyphenation/fihyph/`
Or check that it already is in `/apps/teTeX/std/share/texmf/tex/generic/hyphen/fi8hyph.tex`

```
\usepackage[finnish]{babel}
```

5.5 emacs pd-mode

Installing emacs pd-mode is optional.
To install, just add following to your `.emacs` file and restart

```
(setq auto-mode-alist (cons (cons "\\.pd"   'pd-mode) auto-mode-alist))

;; pd-mode
;;
;; Copyright (C) 1996, 1997 Free Software Foundation, Inc.
;; Derived from m4-mode.el by Andrew Csillag <drew_csillag@geocities.com>
;; as distributed with emacs-21, which see.
;; 28.2.2003, hacked by Sampo Kellomaki <sampo@symlabs.com>
;;
;; Either paste this in your .emacs or arrange it to be loaded.
;; Include -*-pd-*- on first line of your files.
```

```
(defgroup pd nil
  "Major mode for editing PlainDoc documents"
  :prefix "pd-"
  :group 'languages)

(defvar pd-font-lock-keywords
  `(
    ("^[0-9]+.+\n===+$"  . font-lock-string-face)
    ("^[0-9]+.+\n---+$"  . font-lock-string-face)
    ("^[0-9]+.+\n~~~+$"  . font-lock-string-face)
    ("<<\\w+[^>]*>>"     . font-lock-doc-string-face)
    ("\\[\\w+\\]"         . font-lock-type-face)
    ("(\\*\\*\\*[^)]*)"  . font-lock-function-name-face)
    ("\\*\\w[^*]*\\w\\*" . font-lock-type-face)
    ("\\^\\w[^^]*\\w\\^" . font-lock-type-face)
    ("^\\w+[^:]*::"      . font-lock-type-face)
    ("\\~\\w[^~]*\\w\\~" . font-lock-keyword-face)
    ("\\+\\w[^+]*\\w\\+" . font-lock-keyword-face)
    ("\\!\\w[^!]*\\w\\!" . font-lock-keyword-face)
    "Default font-lock-keywords for pd mode.")
)

(defvar pd-mode-syntax-table nil
  "syntax table used in pd mode")
(setq pd-mode-syntax-table (make-syntax-table))
(modify-syntax-entry ?# "<\n" pd-mode-syntax-table)
(modify-syntax-entry ?\n ">#" pd-mode-syntax-table)

(defcustom pd-mode-hook nil
  "*Hook called by `pd-mode'."
  :type 'hook
  :group 'pd)

(defvar pd-mode-map
  (let ((map (make-sparse-keymap)))
    (define-key map "\C-c\C-c" 'comment-region)
    map))

(defvar pd-mode-abbrev-table nil
  "Abbrev table used while in pd mode.")
```

PlainDoc

```
(unless pd-mode-abbrev-table
  (define-abbrev-table 'pd-mode-abbrev-table ()))

;;;###autoload
(defun pd-mode ()
  "A major mode to edit pd files"
  (interactive)
  (kill-all-local-variables)
  (use-local-map pd-mode-map)

  (make-local-variable 'comment-start)
  (setq comment-start "#")
  (make-local-variable 'comment-end)
  (setq comment-end "")
  (make-local-variable 'parse-sexp-ignore-comments)
  (setq parse-sexp-ignore-comments t)
  (setq local-abbrev-table pd-mode-abbrev-table)

  (make-local-variable 'font-lock-defaults)
  (setq major-mode 'pd-mode
        mode-name "pd"
        font-lock-defaults '(pd-font-lock-keywords nil)
        )
  (set-syntax-table pd-mode-syntax-table)
  (run-hooks 'pd-mode-hook))

(provide 'pd-mode)

;; end of pd mode
```

If your document extension is not `.pd`, you can always say

```
M-x pd-mode
```

to get it started.

5.6 GraphViz-2.0

GraphViz is a neat tool for generating diagrammatic graphs from textual input files. The syntax of the graphing language is very natural and easy to learn. Further more, PlainDoc system integrates full support for GraphViz, and specifically

dot@\emphdot()
PlainDoc
gs@\emphgs()

dot(1) tool. You can find more about GraphViz from `graphviz.org`, including how to download and install this great tool.

However, if you do not wish to draw graphs using GraphViz, there is no need to install it.

```
sudo apt-get install graphviz  # works on Ubuntu
```

5.7 GhostScript (gs-8.53)

GhostScript is the real workhorse behind PlainDoc. Many image conversions of pd2tex rely heavily on GhostScript and it is used by visualization software like gv, GSview, gpdf, and xpdf, so life without GhostScript is nearly impossible. Good news is that *pd2tex* is not very sensitive to the version of GhostScript and most *gs(1)* binaries in the mainstream Linux distributions work fine. GhostScript web site: `www.ghostscript.com`

5.8 Other Image Processing Tools

- pnmtools (pngtopnm, pnmtopng, pnmcrop, pnmdepth, pnmscale, pnmtops)
  - usually part of Linux distribution
- djpeg
- gif2ps

5.9 Missing epstopdf (Ubuntu)

```
sudo apt-get install texlive-full          # get everything
sudo apt-get install texlive-extra-utils   # more specific
```

5.10 Missing floatflt (Ubuntu)

```
tlmgr install floatflt
tlmgr update --self --all for a complete update
```

CTAN: `http://www.ctan.org/tex-archive/macros/latex/contrib/floatflt`
Just download the DTX and INS file and run

```
latex floatflt.ins
```

From http://intertwingly.net/blog/2010/04/22/Restoring-floatflt-sty

```
mkdir -p /usr/share/texmf-texlive/tex/latex/floatflt
cd /usr/share/texmf-texlive/tex/latex/floatflt
rm -f floatflt.* float*.tex
wget http://mirror.ctan.org/macros/latex/contrib/floatflt/floatflt.ins
wget http://mirror.ctan.org/macros/latex/contrib/floatflt/floatflt.dtx
latex floatflt.ins
texhash /usr/share/texmf-texlive
```

6 FAQ

The following sections answer some of the common questions. But if your question is not answered, please feel free to contact us – with a well studied question, of course.

6.1 Tips for Cramming a Lot of Legalese in Small Space

In legalese the typographic conventions of readability may be undesirable as the purpose is to have the victim sign without reading. Therefore some tips to cram a lot of fine print:

1. Adjust fontsize:[12]

```
< <class: clean!a4paper,10pt>>
```

2. Adjust margins

```
#< <papersize: fancy!custom!dummy!WIDTH!HEIGHT!LM!TM!RM!BM!HEAD-HEIGHT!HEA
< <papersize: fancy!custom!dummy!210mm!297mm!22mm!7mm!18mm!10mm!2mm!3mm!5m
< <papersize: fancy!custom!dummy!210mm!297mm!35mm!20mm!25mm!15mm!12pt!11mm
```

3. Adjust line and paragraph spacing (see also http://www.terminally-incoherent.com/blog/200

[12] If your legalese is to be faxed, you probably want to stick to 12pt font.

```
<<linespace: !0pt!\medskipamount>>

\setlength{\parindent}{0pt}

\setlength{\parskip}{0pt}
\setlength{\parsep}{0pt}
\setlength{\headsep}{0pt}
\setlength{\topskip}{0pt}
\setlength{\topmargin}{0pt}
\setlength{\topsep}{0pt}
\setlength{\partopsep}{0pt}

\usepackage[compact]{titlesec}
\titlespacing{\section}{0pt}{*0}{*0}
\titlespacing{\subsection}{0pt}{*0}{*0}
\titlespacing{\subsubsection}{0pt}{*0}{*0}
```

4. Adjust list spacing (see also enumitem, http://www.wand.net.nz/ stj2/blog/?p=626)

   ```
   \usepackage{mdwlist} and \begin{itemize*} or \begin{enumerate*}
   ```

   Some other method is supposed to exist.

5. Adjust list spacing

   ```
   \usepackage{enumitem}
   \setitemize{nolistsep,leftmargin=*}
   ```

6. Rewrite text to be more compact

   - Combine paragraphs
   - Inline lists into paragraphs

7. Stretch page to fit one more line

   ```
   <<tex: \enlargethispage*{\baselineskip}>>
   ```

8. Control how rigorously latex splits

   ```
   \sloppy
   ```

9. Even - Odd pages

   ```
   \setlength{\evensidemargin}{15pt}
   \setlength{\oddsidemargin}{45pt}
   ```

6.2 PlainDoc vs. other formats

PlainDoc
PlainDoc
LaTeX
LaTeX
LaTeX
LaTeX
WYSIWYG
Word
Word
Word
Word
Word
Word
Word

1. What about perl pod? Perl pod (Plain Old Documentation) is a pretty good system and, in hindsight, I guess I could simply have improved it, but at the time (2002) it did not seem high enough calibre for serious technical document production (its apparent main focus is on generating software documentation). POD appeals only a little to the neophyte audience.

2. PlainDoc looks like Wiki, why invent another format? Wikis have some "plain text" merits, but the formatting of bulleted lists or section titles does not really follow the usenet news / email convention or culture. Besides, the Wikis have not managed to agree in any common markup. If there ever is common Wiki markup, we will probably support importing and exporting it.

3. Why not just edit directly LaTeX? Pure LaTeX is not human readable and format conversions from LaTeX to, say, DocBook or HTML were at the time (2002) much less than perfect. LaTeX does not appeal to neophyte audience.

4. Why not just edit directly DocBook? Pure DocBook is not human readable and the syntax (as most XML syntax) is too baroque for human editing. Sure you can edit it using emacs, but you will soon start to think "there's gotta be a better way". If you use some GUI/structured editor like OpenOffice to edit DocBook, you will not be able to meaningfully diff the files. DocBook does not appeal to neophyte audience.

5. What about Lyx? Lyx is a GUI. I do not want a GUI. Lyx output is quite texish, thus not very human readable and thus the Lyx document can not be used as the plain text document. Back in 2002 LyX plain text output left much to desire. Sure, LyX does appeal to certain category of neophyte user, but I think it does not help to wean people off the GUI and WYSIWYG model (despite the claims to contrary by LyX team). LyX documents can not be easily diffed since the gui is liable to reformat the entire underlying file any time you do any change.

6. Word will do the job! No. Word is a GUI. Word is not plain text format and word documents are very prone to corruption. Word plain text output leaves much to desire. Word does not run on all platforms. Word documents can not be diffed using simple tools.

7. OpenOffice? Mainly same gripes as with Word. OpenOffice XML file format (or DocBook format) still suffers from the GUI capital crime: any change to the document and the entire XML is liable to be reformatted. This makes diffing them hell (it also does not play nice with cvs, but this is minor point).

6.3 LaTeX tips

LaTeX
TeXpert
LaTeX

Unfortunately its possible that you will during the `pdflatex` command run to TeX related errors and the process stops (`pdflatex` will print a lot of scary looking messages, but unless it stops you can ignore them without much harm done). First, do not panic. You can get out of `pdflatex` by typing `X` and Enter. This will abort the TeX process.[13]

When an error happens, you should understand why. First task is finding where in the document it is happening. The line numbers reported by TeX refer to the `.tex` intermediate file corresponding to your `.pd`. You may examine this file and try to understand the cause, or you may just try searching in the `.pd` source for the text that appears to be causing trouble.

Unless the cause is trivial, or you are a TeXpert, the chances are you are stuck. At this point, either try to get TeX help (read a book, try Google) or try trial and error to see which part of the document is causing the indigestion. You can eliminate parts of document by enclosing them in `ignore` clauses, or just by deleting them entirely. Often this is an iterative process of trying a fix, regenerating, and previewing. Do not give up.

Be suspicious of special characters in complex constructs getting misinterpretted.

Beware that sometimes a structure that does not close, may cause weird errors far down the line. A very common case of this is when you use the empty line hack to introduce wide table columns one per line and you get out of sync.

To cram little more on page use

```
< <tex: \enlargethispage*{\baselineskip}> >
```

6.4 Some common LaTeX errors

Too deeply nested Apparently this really means what it says. Maybe something not closing?

Float too large Picture or table is too large to fit in available space on page. Ignore.

Overfull vbox Means that something didn't really fit. May cause misformatting and ugliness. Ignore, it's only a warning.

[13] By colossal error in user interface design, Control-C is captured so it does not permit you to get rid of the program. You can also try Control-Z and then kill it with *kill(1)* command.

Missing $ inserted Automatic switch to math mode: char (e.g. underscore) only allowed in math mode was seen and LaTeX "helpfully" switches to math mode. Generally fixed either by eliminating the suspect character, enclosing text in `< <tt: ...> >` block, or some other form of escaping. LaTeX LaTeX LaTeX

Twocolumn format: put twocolumn option to article or use multicol mode.

direct tex like

```
just a \hspace{\fill} word
```

N.B. This example only renders decently on PDF (generated using the LaTeX backend).

For accurate freeform layout and positioning, try textpos placement macros (http://purl.org/nxg/dist/textpos).

6.5 Booklet printing

For best results you will want to enable two sided printing (left and right hand side papers have different margins) at LaTeX level:

```
Title
#####
< <class: book!a4paper,12pt!portuges!Zita Lopes, LIP>>
< <linespace: 1.5!!\medskipamount>>
< <author: Zita Maria Oliveira Lopes Kellomäki>>
< <moretexpreamble:
\setlength{\evensidemargin}{15pt}
\setlength{\oddsidemargin}{45pt}
\hyphenation{GEANT Sam-po Kel-lo-mä-ki com-ple-men-tam e-xac-ta}
>>
```

You can print A5 booklets with the following recipe:

```
pd2tex file.pd
pdftops tex/file.pdf
psbook tex/file.ps tex/file-book.ps    # omit -s for best result
mpage -o -2 -j1%2 tex/file-book.ps     # odd sheets
# HP4100: rotate output by 180 degrees and put in input tray with image up (p.
mpage -o -2 -j2%2 tex/file-book.ps     # even sheets
# invert order of output, fold, and staple in middle
```

booklet
PlainDoc
function_names@\er
TeXpert
LaTeX

Provided that you did not screw up with mental gymnastics regarding geometry and transformations that relate to inserting the papers in the right orientation for the second printing pass, you should now have a stack of double side printed A4s that you can fold in middle and staple in the center to make your booklet. Folding will often produce uneven right edge of papers. The best fix is to simply use a good guillotine to even it out.

6.6 Revision Control and Changebars

- Use cvs or svn for revision history
- `http://www.elvis.ac.nz/brain?LaTex/RevisionControl`
- `http://tug.ctan.org/tex-archive/support/latexdiff/`

```
latexdiff-fast ../bak/arch-v04.tex ../arch.tex >arch-v04-v05-diff.tex
```

- Seems to work well enough except around graphics

6.7 Communities and Links

Following projects have their own local instructions (and improvements) for PlainDoc

- http://web.it.kth.se/ fjon/edda/index.html (by Fredrik Jonsson)

Useful links

- `http://www.artofproblemsolving.com/LaTeX/AoPS_L_GuideCommands.php`

6.8 Known bugs

1. Use of underscore outside math mode will confuse TeX. The right fix is to escape the underscore. Unfortunately this is not done automatically, so you have to do it manually. Underscore works right in verbatim blocks and *function_names()*. Similar problem exists for caret.
2. I am not a LaTeX- or TeXpert. I wrote this software to avoid learning LaTeX :-) thus there are probably better ways of doing things if you are in the know.

3. If rendered document starts by "<1sp" after you added

```
\usepackage{lineno}
\linenumbers
```

   clause, then this is due to ordering dependency between packages. It appears that *lineno* package needs to appear before *longtable*, and possibly before *fancyhdr*. Solution: Use 'lineno' as moreopt parameter of class. Otherwise, you will have to hand construct a texpreamble.

class
sampo-plaindoc"@iki.fi
LaTeX
PlainDoc
PlainDoc
Makefile

6.9 Reporting bugs

1. Currently there is no bug tracking or mailing list. If you are willing to set up such things, please let me know. Until then, mail all bug reports, fixes, and feature requests to `sampo-plaindoc@iki.fi` (this alias will help me sort my mail).

2. I do not have resources or time to provide much end user support and specially LaTeX error debugging support. Please make serious effort to investigate and work around the problem before mailing me. If you must include your document or command output, please trim it to a minimal test case that will reproduce your problem.

3. No confidentiality treatment is available for any communication you have with me regarding PlainDoc support. If you must have such treatment, you must pay for it.

4. Please use common sense when reporting bugs. If I see version numbers missing or stupid mistakes I will not reply.

5. I am a plain text person and a laggard in mail technologies. Some of the surest ways of getting your mail ignored are to use attachments, use HTML content, quote entire message without trimming away irrelevancies, fail to put your comments inline, or sending any content that looks like spam. Say what you have to say directly in the message body, including any code listings or command output. *Do not use attachments!*

7 Legal, Copyright, GPLv2 License

PlainDoc System, *pd2tex* processor, `xsd2sg.pl`, Makefile, and Documentation,

Sampo
sampo"@iki.fi
PlainDoc
PlainDoc
PlainDoc

8 Appendix A: High Level Call Graph of pd2tex

Revision history

15 11.4.2013 Sampo

- Added NONL support

14 8.2.2013 Sampo

- Documented RTF support

13 27.3.2009 Sampo

- Changebars and latexdiff

12 12.3.2009 Sampo

- Added doubledia 11:: ?.?.2008, Sampo

10 1.7.2007 Sampo

- Considered EDDA contribution (http://web.it.kth.se/ fjon/edda/index.html) from Fredrik Jonsson <fjon@kth.se>

09 9.9.2006 Sampo

- Merged intelligent parts of Felix's patch
- Improved handling of slide mode
- Prominently documented the raw and natural mode for images

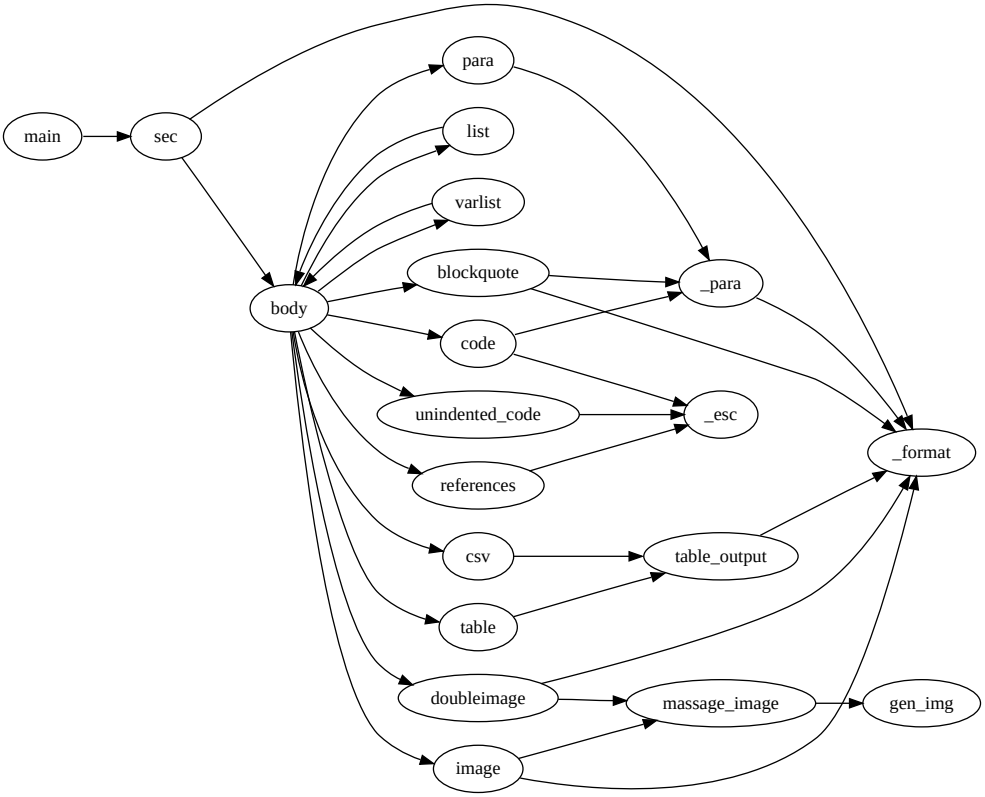


Figure 7: pd2tex call graph (simplified)

- Random updates and improvements to this doc

08 9.7.2006 Sampo

- added rawdbx et al.
- documented showsgasxsd flag

07 22.6.2006 Sampo

- Documented autodetect flag
- Empty line before top level list requirement

06 13.4.2006 Sampo

- IP address detection, link construct
- Improved slides and booklet or two side printing documentation

05 13.1.2006 Sampo

- Separated temp files to tex and html subdirectories
- Added schema, code, and logoutput tags
- Various Liberty derived improvements

04 14.6.2005 Sampo

- Documented dia feature
- Added discussion of slide production

03 29.12.2004 Sampo

- added direct gnuplot and dot support
- added ix and see tags for index and cross referencing

02 18.12.2004 Sampo

- added function, URL, email, and path name recognition and formatting
- improved recognition of formatting in titles and list titles
- ability to specify table column widths explicitly
- ability to specify table ref tags

01 23.10.2004 Sampo Kellomäki (sampo@iki.fi)

- documented nearly all syntax
- new notes for hand over

00 22.9.2004 Sampo Kellomäki (sampo@iki.fi)

- started this document on

References

[LaTeXCompanion] Michel Goossens, Frank Mittelbach, and Alexander Samarin: "The LaTeX Companion". Addison-Wesley, Reading, Massachusetts, 1994.

PDSEAL1F8MNpl99toJ6oW3j.GOhxDdyk68N

Contents

List of Figures

List of Tables

Index