Simplex – Simple LaTeX (v0.3.7)

Julian Fleischer

May 6, 2013

Abstract

Writing LaTeX can be a laborious task. Often the syntax is cumbersome and error-prone. Simplex is an attempt to fix this. A solution for creating documents using LaTeX which is as easy as writing Markdown, without sacrificing the power of LaTeX.

Contents

List of Figures

1 Basic Syntax

The key idea of simplex is that control commands and actual text should go separated. Thus: Everything in simplex is indented, except for control commands.

Let's see how that looks:

```
command
    text
    more text
```

The basic distinction between groups of text is the notion of a paragraph.

```
    this belongs to the first
    paragraph.

    this is yet another paragraph.
    this belongs to the second paragraph as well.
```

A paragraph can be marked as certain type of paragraph. Here for example is a heading:

```
=   Heading of Section 1

    Paragraph 1
```

→ Note that the empty line between the heading and the paragraphs is necessary, since otherwise the paragraph would belong to the heading.

In some circumstances it may come in handy that adjacent paragraphs are treated as one. Commands declaring such paragraphs start with a dot:

```
.verbatim
    This is code.

    As well as this is.
```

It is however terminated by a command as well as any other paragraph.

The contents of such paragraphs can also be included from a file, simply by substituting the dot with a hashbang (#).

```
#code Main.java
```

Commands that start with an at (@) are global declarations. An exhaustive list of commands, global declarations, and special paragraphs can be found in the appendix to this document.

```
@title
    Sample Simplex Document

@authors
    John Doe
    Michael Monk

=   Hello World!

    A sample document might look like this:

#code sample.simple
```

1.1 Document Structure

Headings can be designated using `=`, `==`, and `===`.

Furthermore `:=` and `:-` can be used to designate paragraphs and subparagraphs. The text upto the first colon (:) in a paragraph is used as bold title text for that paragraph:

Sample paragraph This is a sample paragraph.

```
:=  Sample paragraph: This is a sample paragraph.
```

Sample sub paragraph This is a sample sub paragraph.

```
:-  Sample sub paragraph: This is a sample sub paragraph.
```

Table of contents A table of contents can be inserted using the `tableofcontents` command. Note that by default simplex does not number sections (i.e. `=` is translated to `\section*{...}`). If a `tableofcontents` is present in the document, simplex will number sections (i.e. `=` is translated to `\section{...}` – note the missing asterisk).

Figure 1: A sample document featuring a tableofcontents

```
@title
    A Document with a Table of Contents

newpage

tableofcontents
newpage

=   Section One
==  Subsection One-One

    Morbi leo risus, porta ac consectetur ac, vestibulum at eros.
    Sed posuere consectetur est at lobortis. Aenean lacinia
    bibendum nulla sed consectetur.

->  A little advise in between.

    Vestibulum id ligula porta felis euismod semper.
    Cum sociis natoque penatibus et magnis dis parturient montes,
    nascetur ridiculus mus.

==  Subsection One-Two

newpage
=   Section Two

    Nulla vitae elit libero, a pharetra augue.

    Etiam porta sem malesuada magna mollis euismod.
    Donec id elit non mi porta gravida at eget metus.
    Maecenas faucibus mollis interdum.

:=  Nota bene: This is a paragraph.
```

1.1.1 Lists

Lists can be created using `*` and `+`. Lists can be nested:

Unordered lists

- One
  - One-One
- Two
  - Two-One
  - Two-Three

```
1 *   One
2 **  One-One
3 *   Two
4 **  Two-One
5 **  Two-Three
```

Numbered lists

1. One
   (a) One-One
2. Two
   (a) Two-One
   (b) Two-Three

```
1 +   One
2 ++  One-One
3 +   Two
4 ++  Two-One
5 ++  Two-Three
```

Description lists

```
1 :   coffee: the lifeblood of nerds, and the drink that keeps
2     America's workforce complacent on their journey to work.
3 :   demo: Demonstrate the capabilities of (software or equipment).
4 :   psychopsilocybin: A Mushroom...one of the crazy kind. ^.~
5     ALSO an awesome Incubus Song from the Fungus Amoungus CD!
```

coffee the lifeblood of nerds, and the drink that keeps America's workforce complacent on their journey to work.

demo Demonstrate the capabilities of (software or equipment).

psychopsilocybin A Mushroom... one of the crazy kind. ^.∼ ALSO an awesome Incubus Song from the Fungus Amoungus CD!

→ Using double-colons instead of single colons such a list may be rendered like so:

coffee the lifeblood of nerds, and the drink that keeps America's workforce complacent on their journey to work.

demo Demonstrate the capabilities of (software or equipment).

psychopsilocybin A Mushroom... one of the crazy kind. ^.∼ ALSO an awesome Incubus Song from the Fungus Amoungus CD!

1.2 Inline syntax features

The look and feel of the syntax is heavily influenced by Markdown. The following inline markup can be used:

`This is **bold** text`	This is **bold** text
`This is *italic text*`	This is *italic text*
`This _text_ is _underlined_`	This text is underlined
`This is $M_{ath}^2$`	This is M_{ath}^2
`This is \@inline verbatim@`	This is `inline verbatim`

→ Note that you can also use # and ! to delimit inline verbatim like so: `\!inline verbatim!` and `\#inline verbatim#`.

1.2.1 Escaping Special Characters

You should have noticed that there are certain inline control characters which you might want to escape. Doing so is easy. If any of these characters is followed by a white space they lose their special meaning. The white space immediately following the character is not printed, so in order to print such a character followed by a white space you need to type two spaces.

Here is an example (the dots denote spaces):

```
\.$.*._.\..$..*.._..Yea
```

⇒ \$*_\ $ * _ Yea

1.2.2 Hyperlinks, Labels, and References

`\[http://www.example.org/]` ⇒ `http://www.example.org/`
Backslash and brackets insert a hyperlink.

`\[http://www.example.org/ Website]` ⇒ Website
Everything after the first whitespace is treated as description.

`\{http://www.example.org/}` ⇒ `http://www.example.org/`
Square brackets basically work the same...

`\{http://www.example.org/ Website}` ⇒ `http://www.example.org/` – Website
... but they print the link in any case.

`label sec:document-structure`
Labels can be set using the command *label* <*name*> ...

`See section \<sec:document-structure>` ⇒ See section 1.1
... and referenced using backslash and angle brackets.

`See page \(sec:include-directive)` ⇒ See page 22
Pages can be references using parenthesis.

1.2.3 Footnotes and Endnotes

Footnotes[1] are declared like hyperlinks: `\^This is a footnote^`. If you want the footnote to contain a hyperlink as well, use two circumflex at the beginning: `\^^http://www.example.org/ Further reading, example.org, 2010.^` [2]

To turn the footnotes in your document into endnotes use the global declaration `@endnotes` (with at). You can then use the `endnotes` (no at) command to insert the list of endnotes.

Figure 2: A sample document with endnotes, labels, and references

```
@title
    Sample Document with Endnotes

@authors
    Donald Duck
    Daisy Duck

@endnotes

@abstract
    A demo with endnotes, labels, and references.

newpage
tableofcontents

newpage
=   Section One
label sec:one

    This is section one\^^http://www.example.org/
    See example.org for further information^.

=   Section Two
label sec:two

    This is\^yes it is^ section two. Have you
    read section \<sec:one> already?

appendix
=   First appendix

    Lorem ipsum dolor sit amet.

=   Second appendix

    Bilden Sie ganze Sätze man!

endnotes
```

[1] This is a footnote

[2] `http://www.example.org/` – Further reading, example.org, 2010.

1.3 Embedding LaTeX

If you feel the need to embed real LaTeX code into your document, you can easily do so via `.latex`. You can also include LaTeX from another file using `#latex <file>`.

Since simplex takes care of the head and meta data of your document it is not possible to include latex commands outside `\begin{document}` and `\end{document}` using this technique. Instead the global declaration `@preamble` can be used (for example for including additional packages and the like).

Figure 3: A sample document with embedded LaTeX

```
@title The Title

@preamble
    \usepackage{setspace}
    \doublespacing

=   Sample document

.latex
    Here there goes pure latex code.
    \thispagestyle{plain}
.
```

→ Note that the commands demoed here are available in simplex too: `@doublespacing` and `thispagestyle plain`.

1.4 Comments

You can comment out a paragraph by marking it `%`. Larger passages of comments can be declared using `.comment`.

Figure 4: A sample document with comments in it

```
%   A comment
=   Not a comment (a heading)

.comment
    Comment

    Also comment

==  Subsection (terminates .comment)
```

2 Advanced features

2.1 Graphics / Images

Images can be included using the `image` command:

```
image image.png
```

The size and rotation of images can be controlled using these commands:

image-width Sets the width for all following images to the given width. Example: `image-width 8cm`. Note that the values given here are passed to pdflatex literally, thus it is possible to use values like `\textwidth` here.
Example: `image-width \textwidth`.

image-height analog to `image-width`.

image-size Sets both width and heigth for all following images.
Example: `image-size 16cm 9cm`.

image-angle Sets the rotation of an image.

image-trim Trims the following images by the given lengths.
The arguments are *left bottom right top* (i.e. counter clockwise).
Example: `image-trim 1cm 0cm 2cm 1cm`

image-defaults Clears all image directives.

→ It is worth noting that image directives apply to all following images, not only the next image. Thus `image-angle 180` will rotate all following images upside down. Use `image-defaults` to clear the given commands.

→ Supported image types are **png**, **jpg**, and **pdf**.

Figure 5: Example image-inclusions

```
image-height 5cm
image-angle 170

columns 3
image zebra.jpg
image elephant.jpg
image-angle 350
image zebra.jpg
endcolumns
```

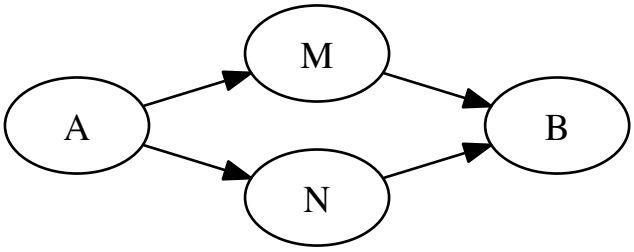
2.1.1 Builtin *Graphviz* support

Graphviz graphs can be included directly into a simplex document:

```
.digraph
    rankdir=LR;

    A -> M -> B;
    A -> N -> B;
```

is rendered as:



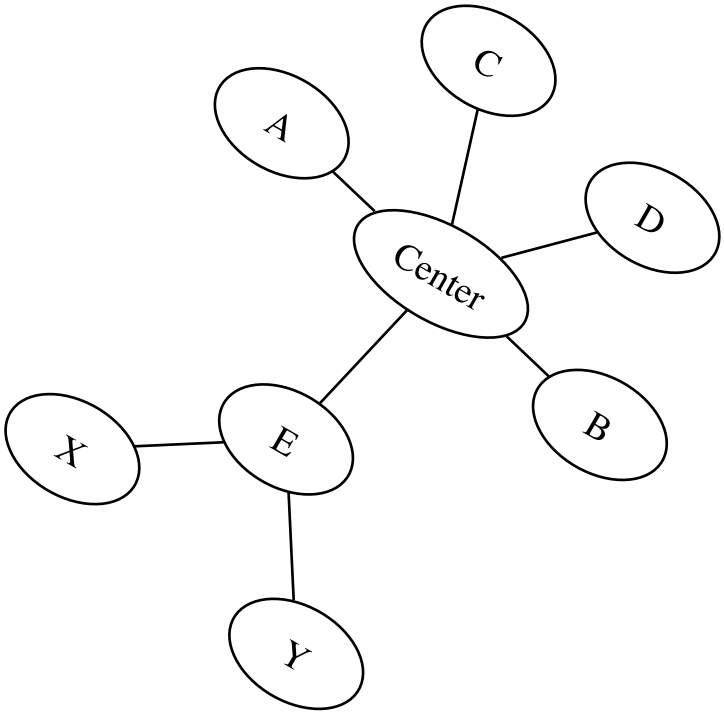
→ Note that the image-directives apply on graphviz graphs too.

→ `.digraph` is drawn using dot.

→ `.graph` is drawn using neato.

```
image-angle -30
.graph
    margin=0;

    A -- Center;
    B -- Center;
    C -- Center;
    D -- Center;
    E -- Center;
    X -- E -- Y;
```



2.2 Multi column layout

Text can be distributed over multiple columns via the `columns` command. `columns` are terminated by the `endcolumns` command.

```
columns 2
    Nam dui ligula...
endcolumns
```

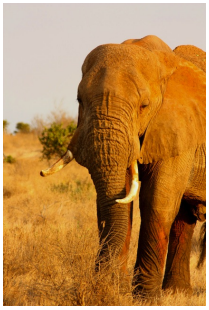
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Text can be forced to continue in the next column using the `colbreak` command. Here is an example using two images:

```
image-height 4cm
columns 2
center
image bushes.jpg
image duck.jpg
colbreak
image zebra.jpg
image elephant.jpg
endcolumns
```

→ this also demonstrates the usefulness of the image commands applying to all following images.

. . . which produces the following rendering:



2.3 Figures

Figures (like `\begin{figure}` in LaTeX) can be inserted using the `figure` and `endfigure` commands. A caption can be set using the `caption` command.

```
figure
image-width \textwidth
caption This is an example figure
image bushes.jpg
endfigure
```

Figure 6: This is an example figure

3 Tables

All table commands start with `>`. Every `>` declares a cell. `>+` starts a new table row. `>-` starts a new table row and paints a horizontal line. `>=` starts a new table row and paints a double horizontal line. A table format (just like in `\begin{tabular}{format}`) can be given with `>@`.

Figure 7: Simple sample tables

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

The Code to create these four tables can be seen below:

```
1  >   1
2  >   2
3  >   3
4  >+
5  >   4
6  >   5
7  >   6
8  >+
9  >   7
10 >   8
11 >   9
```

```
1  >@  l|c||r
2  >   1
3  >   2
4  >   3
5  >+
6  >   4
7  >   5
8  >   6
9  >+
10 >   7
11 >   8
12 >   9
```

```
1  >@  l|c||r
2  >-
3  >   1
4  >   2
5  >   3
6  >+
7  >   4
8  >   5
9  >   6
10 >+
11 >   7
12 >   8
13 >   9
14 >-
```

```
1  >@  |l|c||r|
2  >-
3  >   1
4  >   2
5  >   3
6  >-
7  >   4
8  >   5
9  >   6
10 >=
11 >   7
12 >   8
13 >   9
14 >-
```

3.1 Kinds of Cells

Cells can have different kinds of content. A hashbang (#) designates a cell having verbatim content. An exclamation mark (!) designates a header cell (that is: having a **bold** type face). A dollar sign ($) designates a cell having math content.

Figure 8: A sample table with different kinds of cells

```
1 >@  |c|c|
2 >-
3 >$  \sum_{i=1}^n n^2
4 >!  Heading
5 >-
6 >#  Verbatim \yeah <=> \yeah
7 >   Ordinary cell.
8 >-
```

$\sum_{i=1}^{n} n^2$	**Heading**
`Verbatim \yeah <=> \yeah`	Ordinary cell.

→ This way tables can be used to create complex layouts, for example equations. See section 3.8 for an advanced example.

3.2 Alignment of Cells

The alignment of cells can be changed per column (using `>@`) or per cell. All you need to do is to include `L`, `R`, or `C` (for Left, Right, or Center respectively) in the cell definition:

Figure 9: A table demoing custom alignment of cells

```
>@  clr
>!  Heading 1
>!  Heading 2
>!  Heading 3
>=
>   A
>   B
>   C
>-
>   D
>   E
>   F
>-
>L  Le
>R  Ri
>C  Ce
>=
```

Heading 1	**Heading 2**	**Heading 3**
A	B	C
D	E	F
Le	Ri	Ce

Note that this will redefine the cell apart from the table declaration (given via `>@  clr`). Thus if you specified vertical borders in `>@` you will have to specify them in the cell again. This is possible by putting a bar (|) left and/or right to the new cell definition:

Figure 10: A table demoing formatting of custom cells

```
>@  |c|l|r|
>!  Heading 1
>!  Heading 2
>!  Heading 3
>=
>   A
>   B
>   C
>-
>|L| Le
>R| Ri
>C| Ce
>=
```

Heading 1	**Heading 2**	**Heading 3**
A	B	C
Le	Ri	Ce

3.3 Text in Tables

Tables in LaTeXare not as easy as, say, HTML tables. Especially a problem are cells containing multiline text (which, by default, is not wrapped). Thus you have to declare a text column (using `>@`). The column specifier is `p`. You will also have to declare the width of the column, like so: `p{4cm}`.

Figure 11: An example table with multiline text

```
>@  |r|p{6cm}|
>-
>!  Head
>!  Yet another
>-
>   1337
>   ...
>-
```

Head	Yet another
1337	Nullam quis risus eget urna mollis ornare vel eu leo. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.

There is also a more flexible `tabular`-environment, called `tabluarx`. Is supports columns which contain text and adjust themselves accordingly. You will still have to designate a column as text column, but you do not need to specify the width. Simplex will turn your table into a `tabularx`-table automatically if you define the table using `>X` instead of `>@`. The column specifier for a `tabularx`-text-column is `X`.

Figure 12: An example X-table

```
>X  |r|X|
>-
>!  Head
>!  Yet another
>-
>   1337
>   Nullam quis risus eget urna
    mollis ornare vel eu leo. Integer posuere erat
    a ante venenatis dapibus posuere velit aliquet.
>-
```

Head	Yet another
1337	Nullam quis risus eget urna mollis ornare vel eu leo. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.

However, `>X` tables have shortcomings of their own. For example you can not use inline verbatim code in a cell inside a `tabularx`-table.

3.4 Rows spanning multiple columns

It is possible to let a row span multiple columns by simply adding the number of columns the row should span in the cell definition. Note that this will nullify all column declarations just as `L`, `R`, and `C` do, so you will have to specify borders as necessary. Columns spanning multiple rows are centered by default (you can combine column-spanning numbers and alignment specifiers, like `>|2L|` or `>11C!`).

Figure 13: A table with a single cell spanning two columns

```
>@  |l|l|
>-
>|2|
    Team sheet
>-
>   GK
>   Paul Robinson
>+
...
```

Team sheet	
GK	Paul Robinson
LB	Lucus Radebe
DC	Michael Duberry
DC	Dominic Matteo
RB	Dider Domi
MC	David Batty
MC	Eirik Bakke
MC	Jody Morris
FW	Jamie McMaster
ST	Alan Smith
ST	Mark Viduka

3.5 Columns spanning multiple rows

Figure 14: A table showing how to span multiple rows and columns

```
>@  |l|l|l|
>-
>|3|
    Team sheet
>-
>   Goalkeeper
>   GK
>   Paul Robinson
>-
>,4 Defenders
>   LB
>   Lucus Radabe
...
```

Team sheet		
Goalkeeper	GK	Paul Robinson
Defenders	LB	Lucus Radabe
	DC	Michael Duberry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

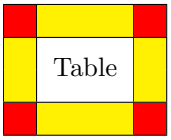
3.6 Colored Cells

Figure 15: Four coloured cells in a table

```
>@  cc
>red
    A
>blue
    B
>+
>green
    C
>yellow
    D
```

Figure 16: A very colorful table

```
>@  |c|c|c|c|
>-
>red
>2yellow|
>red
>-
>yellow,2
>2,2| Table
>yellow,2
>+
>yellow
>C
>C|
>yellow
>-
>red
>2yellow|
>red
>-
```



3.7 ASCII-Art Tables

Figure 17: An ASCII-Art Table

```
.table
    +-----+----+----+----+
    |   7 |  8 |  1 | 19 |
    +-----+----+----+----+
    | 172 | 19 | 42 | 20 |
    +-----+----+----+----+
    | 420 |  4 | 17 | 64 |
    +-----+----+----+----+
```

7	8	1	19
172	19	42	20
420	4	17	64

3.8 Typesetting Equations using Tables

$\Leftrightarrow$	$x^2 + bx + c$	$=$	0	isolate x: $-c$
$\Leftrightarrow$	$x^2 + bx$	$=$	$-c$	$+\left(\frac{b}{2}\right)^2$ *(completing the square)*
$\Leftrightarrow$	$x^2 + bx + \left(\frac{b}{2}\right)^2$	$=$	$-c + \left(\frac{b}{2}\right)^2$	binomial formula reverse
$\Leftrightarrow$	$\left(x + \frac{b}{2}\right)^2$	$=$	$\left(\frac{b}{2}\right)^2 - c$	$\sqrt{\ }$
$\Leftrightarrow$	$x + \frac{b}{2}$	$=$	$\pm\sqrt{\left(\frac{b}{2}\right)^2 - c}$	$-\frac{b}{2}$
$\Leftrightarrow$	x	$=$	$-\frac{b}{2} \pm \sqrt{\left(\frac{b}{2}\right)^2 - c}$	

```
>@  clcl | l

>$  <=>
>$  x^2 + bx + c
>$  =
>$  0
>   isolate $x$: $-c$

>+

>$  <=>
>$  x^2 + bx
>$  =
>$  -c
>  $+\left(\frac b 2\right)^2$ *(completing the square)*

>+

>$  <=>
>$  x^2 + bx + \left(\frac b 2\right)^2
>$  =
>$  -c +\left(\frac b 2\right)^2
>   binomial formula reverse

>+

>$  <=>
>$  \left( x + \frac b 2 \right)^2
>$  =
>$  \left(\frac b 2\right)^2 - c
>$  \sqrt{\text{\nbsp}}

>+

>$  <=>
>$  x + \frac b 2
>$  =
>$  \pm \sqrt{\left(\frac b 2\right)^2 - c}
>$  -\frac b 2

>+

>$  <=>
>$  x
>$  =
>$  -\frac b 2 \pm \sqrt{\left(\frac b 2\right)^2 - c}
```

A Blocks

Document Structure

`=`
`==`
`===`
	Headings

`!!`	Chapters
`!!!`	Parts
`:=` `:-`	Paragraphs with a defining word at the start of the line.

Lists & Items

`*`	Unnumbered items (like *itemize*).
`+` `_`	Numbered items (like *enumerate*).
`:` `::`	Items with a defining word at the start of the line.
`->`	Advise items.

Tables

`>`	A table cell. See "Tables" for more information.
`>+`	Next row, no line
`>-`	Next row, single line
`>=`	Next row, double line
`>^`	Turns the table into a *figure* with a *caption* above.
`>_`	Turns the table into a *figure* with a *caption* below.
`>@`	The table definition (like in LaTeX).
`>X`	The table definition, also turns the table into a *tabularx* table.

Others

`.`	A paragraph (like no control command at all, useful for explicitly creating a paragraph for example in order to terminate a table).
`=>`	A paragraph introduced by $\Rightarrow$.
`<=`	A paragraph introduced by $\Leftarrow$.
`<=>`	A paragraph introduced by $\Leftrightarrow$.
`=!>`	A paragraph introduced by $\not\Rightarrow$.
`<!=`	A paragraph introduced by $\not\Leftarrow$.
`<!>`	A paragraph introduced by $\not\Leftrightarrow$.

B Special blocks

→ Note: By exchanging the dot (`.`) with a hashbang (`#`) you can load data from a file and treat it like the corresponding special block (i.e. `#code myfile.java` will include the contents of *myfile.java* and treat it as a `.code` block).

.ascii An alias for `.verbatim`.

.code Verbatim text, formatted as source code.

.code$ Like `.code` but allows for mathescapes (`$...$`).

.comment Does not include its contents into the document.

.digraph Creates a graphviz `digraph` (a directed graph) and renders it using `dot`.

.equation Inserts raw math tex, like `.latex` with `\begin{equation}`.

.graph Creates a graphviz `graph` and renders it using `neato`.

.haskell Like `.code`, but loads definitions for *Haskell*.

.java Like `.code`, but loads definitions for *Java*.

.latex Inserts raw latex code.

.math Inserts raw math tex, like `.latex` with `\begin{displaymath}`.

.php Like `.code`, but loads definitions for *PHP*.

.python Like `.code`, but loads definitions for *Python*.

.table Parses an ASCII table and renders it using LaTeX.

.verbatim Inserts verbatim text (like `\begin{verbatim}`).

B.1 Shortcuts

.# Like `.code`.

.@ Like `.code$`.

.% Like `.comment`.

.$ Like `.math`.

.! Like `.verbatim`.

→ Note that replacing the dot with a hashbang works here too. Thus `## file.txt` will in fact include file.txt as code, `#! file.txt` as verbatim text, and so on.

B.2 #include Directive

#include Literally includes the contents of the file. Usage: `#include filename`. This works like the include-command for the c preprocessor. Note that no other features of cpp are available in simplex.

#image An alias for the ordinary command `image`. There is a slight difference: Like in #include, the file name may be given in quotation marks:

`#image "picture.png"` – works!
`image "picture.png"` – won't work (quots will be regarded as part of the filename)

C Declarations

@abstract Used to declare an abstract of the current document.

@address Your address if this is a `@letter`.

@article Declares the document to be an article. This will cause the `article` document class to be used.

@authors A list of authors of the document. Each line corresponds to one author. Note that there is no single `@author` declaration.

@book Declares the document to be a book. This will cause the `book` document class to be used.

@cfoot Useful with `@pagestyle fancy`. The text to be used in the center part of the foot. Use `\break` to insert line breaks in here.

@chead Useful with `@pagestyle fancy`. The text to be used in the center part of the head. Use `\break` to insert line breaks in here.

@closing The closing text if this is a `@letter`. Defaults to "Yours Faithfully".

@date Sets the date of the document. If none is given, `\today` is assumed.

@doublespacing Turns on double spacing. This is useful for drafts in order to correct things.

@draft Declares the document as a draft. This will, inter alia, cause images not to be included – which in turn will reduce build times.

@endnotes Turns all footnotes into endnotes. Use in conjunction with `endnotes` command to actually insert the endnotes (like `tableofcontents`).

@fontsize The standard font size in this document. Defaults to 12pt.

@landscape Sets `landscape` mode (90 degree rotated).

@language Specifies the language of the document. Translates directly into `\usepage[LANGUAGE]{babal}` in the preamble, where LANGUAGE is the value of the `@language` declaration. Values are for example `ngerman`, `frenchb`, or `francais`.

@letter Uses the `letter` document class. You should use it together with `recipient`, `address`, `signature`, `opening`, and `closing`.

@lfoot Useful with `@pagestyle fancy`. The text to be used in the left part of the foot. Use `\break` to insert line breaks in here.

@lhead Useful with `@pagestyle fancy`. The text to be used in the left part of the head. Use `\break` to insert line breaks in here.

@margin-bottom The bottom margin of each page.

@margin-left The left margin of each page.

@margin-right The right margin of each page.

@margin-top The top margin of each page.

@margins Sets the margins of the documents. A white-space separated list of lengths. The first value is the top-margin and then clockwise.

@newpagesections Automatically insert `newpage` commands before new top level sections.

@opening The opening text if this is a `@letter`. Defaults to "Dear Sir or Madam,".

@pagestyle For example `@pagestyle fancy`. Declares the global pagestyle (in contrast: the `pagestyle` command will declare the pagestyle locally).

@preamble Inserts raw LaTeXinto the head of the document (which is normally not accessible from within a simplex document).

@recipient The address of the recipient if this is a `@letter`.

@report Uses the `report` document class.

@rfoot Useful with `@pagestyle fancy`. The text to be used in the right part of the foot. Use `\break` to insert line breaks in here.

@rhead Useful with `@pagestyle fancy`. The text to be used in the right part of the head. Use `\break` to insert line breaks in here.

@scrartcl Uses the `scrartcl` document class.

@slides Used the `slides` document class (this affects the paper size) – useful for creating presentations.

@signature Your name if this is a `@letter`. Used in the address as sender and in the closing as signature.

@title The title of the document. Line breaks in here will be actual line breaks in the target pdf.

@tocdepth To what depth headings should be part of the table of contents. Defaults to 3.

D Lengths

baselineskip

baselinestretch

columnsep

columnseprule

columnwidth

evensidemargin

headheight

oddsidemargin

paperheight

paperwidth

parindent

parskip

tabcolsep

textfloatsep

textheight

textwidth

topmargin

E Commands

appendix Starts the appendix. Sections from this point on will be counted using letters. Does not insert an appendix heading.

bfseries Like `\bfseries` in LaTeX. Prints the following text in **bold**. Use `reset` or `normalfont` to terminate.

bold Alias for `\bfseries`. Prints the following text in **bold**. Use `reset` or `normalfont` to terminate.

caption Declares a caption for the current figure. Only works within `figure`.

center Centers the text from this point on.

colbreak Starts a new column within `columns`.

columns Start columnized text.
Usage: *columns <how-many-columns>*

em Like `\em` in LaTeX.

endcolumns Terminates `columns`.

endfigure Terminates `figure`.

endignore Terminates `ignore`.

endnoinclude Terminates `noinclude`.

endnotes Insert the list of endnotes, also creates a section with heading.

figure Starts a figure (like `\begin{figure}`). Takes an optional parameter which specifies the placement of the figure. If you omit it, `h!` is assumed. If you want to explicitly leave it blank, specify `auto`.
All possible values are: *h, t, b, p, H, !* (may be combined).
Examples: `figure htb`, `figure auto`, `figure h!`

figures Insert a list of figures here.

float-barrier `figure`s will not float beneath this command.

footnotesize Like `\footnotesize` in LaTeX.

hfill Like `\hfill` in LaTeX.

huge Like `\huge` in LaTeX.

Huge Like `\Huge` in LaTeX.

ignore Ignores the document upto the next `endignore`.

image-angle Sets the rotation of the following images.
Usage: *image-angle <angle>*

image-defaults Resets all image properties.

image-height Sets the height of the following images.
Usage: *image-height <height>*

image-page In case of a multi-pdf file, selects which page should be included.

image-scale Scales images by the given factor.

image-size Sets both image-height and image-width at once.
Usage: *image-size <width><height>*

image-trim Crop the included image.
Usage: *image-trim <left><bottom><right><top>*

image-width Sets the width of the following images.
Usage: *image-width <width>*

image Inserts the specified image, for example `image myfile.png`. Works like `\includegraphics` in LaTeX. Dimensions and other additional arguments can be set using the `image-...` commands. Note that the same limitations apply as for pdflatex and includegraphics in general (i.e. will only include **JPG**, **PNG**, and **PDF** files).

italic Alias for `\itshape`. Prints the following text in *italics*. Use `reset` or `normalfont` to terminate.

itshape Like `\itshape` in LaTeX. Prints the following text in *italics*. Use `reset` or `normalfont` to terminate.

large Like `\large` in LaTeX.

LARGE Like `\LARGE` in LaTeX.

Large Like `\Large` in LaTeX.

left Aligns the text from this point onward to the left. Note that this command will not justify the text (which LaTeXdoes by default). Use `reset` for this.

lipsum Includes sample text of lorem ipsum.

mdseries Like `\mdseries` in LaTeX.

newpage Like `\newpage` in LaTeX.

noinclude The following part of the document (up to the next `endnoinclude`) won't be processed when included using `#include` (see B.2).

noindent Like `\noindent` in LaTeX.

normalfont Like `\normalfont` in LaTeX.

normalsize Like `\normalsize` in LaTeX.

pagebreak Alias for `\newpage`.

pagenumbering Restart the page numbering here with the given style.
Possible values are: `arabic`, `roman`, `Roman`, `alph`, `Alph`

pagestyle Like `\pagestyle` in LaTeX. Sets the pagestyle for the next pages from this point on to the supplied argument. For example `pagestyle plain` or `pagestyle fancy`.

reset Cancels the effects of `left`, `center`, `right` and applies `normalsize` and `normalfont`.

right Aligns the text from this point onward to the right.

rmfamily Like `\rmfamily` in LaTeX.

scriptsize Like `\scriptsize` in LaTeX.

scschape Like `\scschape` in LaTeX.

sffamily Like `\sffamily` in LaTeX.

slshape Like `\slshape` in LaTeX.

small Like `\small` in LaTeX.

tableofcontents Like `\tableofcontents` in LaTeX. Will additionally turn numbering of sections on (which is turned off by default in simplex).

thispagestyle Changes the style of the current page, for example `thispagestyle plain`.

tiny Like `\tiny` in LaTeX.

ttfamily Like `\ttfamily` in LaTeX.

upshape Like `\upshape` in LaTeX.

vfill Like `\vfill` in LaTeX.

F Symbols

```
\A  \B  \C  \D  \E  \F  \G  \H  \I  \J  \K  \L  \M
\N  \O  \P  \Q  \R  \S  \T  \U  \V  \W  \X  \Y  \Z
```

$\mathbb{A}\ \mathbb{B}\ \mathbb{C}\ \mathbb{D}\ \mathbb{E}\ \mathbb{F}\ \mathbb{G}\ \mathbb{H}\ \mathbb{I}\ \mathbb{J}\ \mathbb{K}\ \mathbb{L}\ \mathbb{M}\ \mathbb{N}\ \mathbb{O}\ \mathbb{P}\ \mathbb{Q}\ \mathbb{R}\ \mathbb{S}\ \mathbb{T}\ \mathbb{U}\ \mathbb{V}\ \mathbb{W}\ \mathbb{X}\ \mathbb{Y}\ \mathbb{Z}$

```
\Ac  \Bc  \Cc  \Dc  \Ec  \Fc  \Gc  \Hc  \Ic  \Jc  \Kc  \Lc  \Mc
\Nc  \Oc  \Pc  \Qc  \Rc  \Sc  \Tc  \Uc  \Vc  \Wc  \Xc  \Yc  \Zc
```

$\mathcal{A}\ \mathcal{B}\ \mathcal{C}\ \mathcal{D}\ \mathcal{E}\ \mathcal{F}\ \mathcal{G}\ \mathcal{H}\ \mathcal{I}\ \mathcal{J}\ \mathcal{K}\ \mathcal{L}\ \mathcal{M}\ \mathcal{N}\ \mathcal{O}\ \mathcal{P}\ \mathcal{Q}\ \mathcal{R}\ \mathcal{S}\ \mathcal{T}\ \mathcal{U}\ \mathcal{V}\ \mathcal{W}\ \mathcal{X}\ \mathcal{Y}\ \mathcal{Z}$

```
\Af  \Bf  \Cf  \Df  \Ef  \Ff  \Gf  \Hf  \If  \Jf  \Kf  \Lf  \Mf
\Nf  \Of  \Pf  \Qf  \Rf  \Sf  \Tf  \Uf  \Vf  \Wf  \Xf  \Yf  \Zf
```

$\mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{E}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{H}\ \mathfrak{I}\ \mathfrak{J}\ \mathfrak{K}\ \mathfrak{L}\ \mathfrak{M}\ \mathfrak{N}\ \mathfrak{O}\ \mathfrak{P}\ \mathfrak{Q}\ \mathfrak{R}\ \mathfrak{S}\ \mathfrak{T}\ \mathfrak{U}\ \mathfrak{V}\ \mathfrak{W}\ \mathfrak{X}\ \mathfrak{Y}\ \mathfrak{Z}$

F.1 Magic Symbols

→ Magic Symbols are those whose values change by context.

\lastpage Inserts the number of the last page.

\thepage Inserts the number of the current page.

\thesection Inserts the number of the current section.

\thechapter Inserts the number of the current chapter.

\today Inserts the current date.

F.2 Special Symbols

→ Special Symbols do not need to be introduced by a backslash. They are recognized and substituted in math mode as well as in text.

<=	$\Leftarrow$	=>	$\Rightarrow$	<=>	$\Leftrightarrow$	!=	$\neq$	(-)	$\ominus$	[-]	$\boxminus$
<==	$\Longleftarrow$	==>	$\Longrightarrow$	<==>	$\Longleftrightarrow$	===	$\equiv$	(+)	$\oplus$	[+]	$\boxplus$
<!=	$\not\Leftarrow$	=!>	$\not\Rightarrow$	<!>	$\not\Leftrightarrow$	~>	$\rightsquigarrow$	(x)	$\otimes$	[x]	$\boxtimes$
<-	$\leftarrow$	->	$\rightarrow$	<->	$\leftrightarrow$	<-<	$\rightarrowtail$	(.)	$\odot$	[.]	$\boxdot$
<--	$\longleftarrow$	-->	$\longrightarrow$	<-->	$\longleftrightarrow$	>->	$\leftarrowtail$	(*)	$\circledast$	|-->	$\longmapsto$
<!-	$\not\leftarrow$	-!>	$\not\rightarrow$	_|_	$\perp$	|=	$\models$	(/)	$\oslash$	<--|	$\longleftarrow\!\shortmid$
<-|	$\leftarrow\!\shortmid$	|->	$\shortmid\!\rightarrow$	<=|	$\Leftarrow\!\shortmid$	|=>	$\shortmid\!\Rightarrow$	<==|	$\Longleftarrow\!\shortmid$	|==>	$\shortmid\!\Longrightarrow$

G Command Line Usage

When invoking `simplex` in a terminal, it will automatically process all `*.simplex` files in the current working directory.

You can use the `-w` switch to have simplex watch your directory for changes and automatically processing your files as you edit and save them.

```
simplex [options] [files...]

  -h        --help              Print this help text.
            --version           Print version information.
  -v        --verbose           Verbose output.
  -d        --dry-run           Dry run (do not create any files).
  -n        --no-clean          Do not clean up after building.
  -p        --print             Print processed tex to stdout.
  -c        --crop              Crops the document so that no margin
                                are left.
  -f        --force             Forces the creation of output files.
  -t , -T   --type=             Specify type of output (pdf, png, tex)
  -x        --pdflatex=         Path to ‘pdflatex’ executable
  -k        --pdfcrop=          Path to ‘pdfcrop’
  -z        --graphviz=         Path to ‘dot’ (graphviz)
  -g        --gnuplot=          Path to ‘gnuplot’
  -m        --convert=          Path to ‘convert’ (ImageMagick)
  -w[]      --watch[=]          Watch files or folder (optionally
                                amount of time in ms)
  -3        --three-times       Execute ‘pdflatex’ three times instead
                                of the default two times.
            --density=, --dpi=  For output type ‘png’ only,
                                specifies dpi.
            --quality=          For output type ‘png’ only,
                                specifies quality.
```

H Installation

Installation is pretty much straight forward. `simplex` is known to work with the Haskell Platform 2012.2.0.0 and 2012.4.0.0. The Haskell Platform 2012.2.0.0 is also the one included in the most recent Debian distribution (Debian 7.0 "Wheezy" at the time of this writing).
⟶ `http://haskell.org/platform`

If TeX Live is already installed than all you have to do is to install simplex via cabal install, as simplex is available as a package on hackage:
⟶ `cabal install simplex`.

H.1 Ubuntu / Debian

In order to install simplex you need to install the Haskell Platform, which can be done via
⟶ `apt-get install haskell-platform`

Furthermore you should install TeX Live (or another LaTeX distribution which provides `pdflatex`):
⟶ `apt-get install texlive texlive-latex-extra texlive-math-extra`

If you want to use graphviz graphs within your simplex documents, you should also install graphviz:
⟶ `apt-get install graphviz`

Finally:
⟶ `cabal install simplex`

If the `simplex` executable is not on your path, you might want to add `$HOME/.cabal/bin` to your `$PATH`.
to your `PATH`.

H.2 Mac OS X

You have to have a LaTeX distribution installed in your Mac OS, such as TeX Live. If you do not, there is an excellent package named *MacTeX* available at the TeX Users Groups Website:
⟶ `http://www.tug.org/mactex/`.

You should also install `graphviz` and `ImageMagick`, both of which are available via brew – a simple package manager for OS X:
⟶ `brew install graphviz` ⟶ `brew install imagemagick`

Finally:
⟶ `cabal install simplex`

If the `simplex` executable is not on your path, you might want to add `$HOME/Library/Haskell/bin`
to your `PATH`.

H.3 Windows

In theory it should suffice to have the Haskell Platform and TeX Live installed, but the installation of Simplex will most probably pull in some newer versions of certain Haskell packages which might require a Unix Toolchain on your computer. *Cygwin* is an excellent choice:
⟶ `http://cygwin.com/install.html`

TeX Live is also availbe for Windows. The easiest way to install it is probably the network installer. Inside `install-tl.zip` there is a batch file (`*.bat`) which will download and install all the necessary LaTeX packages to your system:
⟶ `http://www.tug.org/texlive/acquire-netinstall.html`

Finally (in a cygwin terminal):
⟶ `cabal install simplex`

The `simplex` executable should be on your path now, both in cygwin terminals as well as in ordinary dos prompts, as the Haskell Platform will have added the necessary directories to your `PATH` already.