Sample LaTeX document

A sample file to showcase LaTeX

Pieter van den Hombergh
879417

drawing by Duane Bibby

Fontys Hogeschool voor Techniek en Logistiek
Software Engineering/Business Informatics
Tegelseweg 255
5912 BG Venlo
The Netherlands

Fontys
Hogescholen
Fontys Hogeschool voor Techniek en Logistiek

Contents

List of Figures

Introducing oneself properly is always hard.

Anonymous

1 Introduction

One of the standards for documentation in open source and hence in Linux land is LaTeX, a text processing package. LaTeX is available for free and available with all Linux distributions and installed in the Lab.

TeX is the machinery of LaTeX and was defined in the TeX book [Knu84] and implemented by Prof. Donald Knuth. LaTeX is a (nowadays HUGE) set of macros built on top of that. LaTeX in its initial form is described by Leslie Lamport in [Lam86]. If you like your book thick, try the LaTeX companion [MG04].

The web is also a very good source of LaTeX documentation. A good starting point is `http://en.wikibooks.org/wiki/LaTeX`, useful for beginners and pros alike.

This is a simple multi part document. It's purpose is to show how easy it is to create a multi part document, one that, for instance, can be worked on simultaneously by several authors. Note that most of the settings for this document are set in the file `mydefs.tex`. Look in that file too.

This document consists of the following files

1. `main.tex`
2. `motivation.tex`
3. `graphics.tex`
4. `codelisting.tex`
5. `poser.tex`
6. `hello.c`
7. `Makefile` and
8. `mydefs.tex`
9. `servlet3.png`, the pie chart
10. `Diagram1.pdf`, the UML diagram

You are kindly advised to keep your lab logs in simple text files. These can be turned into latex files easily, which can be used to produce a nice looking report.

1.1 Some hints to start with

Sometimes things do not work out the way you think. LaTeX interpretes some character codes in it's own way. Things like dollar signs or even underscore are special. LaTeX source are littered with accolades or *curly braces* if that's the way you call them. They are special too. So here is some advice:

funny file names Do not use *funny file names*. That is: stick to ASCII filenames without spaces or even underscores. These will bring only you into trouble. If you want to keep things portable, don't use camel case (like in JavaClassNames) either, because some OS-es do not distinguish between upper and lower case. You may of course brake this rule if the files are program things like Java source files.

1.1.1 Hints for informatics (use version control)

In software projects, versioning is important. LaTeX and SubVersion work nicely together here. By using the LaTeX package `svn-multi` you can have svn+ latex insert meta information about your files in the produced output, for instance in the page footers, as in this document.

If you add the codes below at the top of each .tex file, these codes will be expanded/updated by svn *on checkin*.

```
\svnidlong{$HeadURL$}%
{$LastChangedDate$}%
{$LastChangedRevision$}%
{$LastChangedBy$}
\svnid{$Id$}
```

You must also tell subversion to expand these keywords for those files with for each file the command:

```
svn propset svn:keywords "Id Author File Date LastChangeDate
  Revision HeadURL Header" filename
```

To keep these version codes up to date, first check if your LaTeX files compile, then check them in and do your final LaTeX run.

To make the version codes of the files appear on the bottom line, have a look at the mydefs.tex file where the fancy headers and footers are defined. In group work you may also find it comfortable to create a file named `myauthors.tex` with contents like below, which is then input into the main or mydefs at the proper place and can translate student numbers from svn and you peerweb account into a humanly readable authors name.

```
\svnRegisterAuthor{879417}{Pieter van den Hombergh}
```

What moves, motivates.

Ancient wisdom of technical teachers

2 Motivation

To write a simple document, like a letter, a word processing package is just fine. To automate the creation of a document this is a bit harder. Using a word processor to automatically create documents out of unrelated sources is difficult at best, if doable at all.

However, as long as the only thing that those unrelated sources should produce are simple ASCII documents, things get much more doable. It compares like HTML to Microsoft Word documents. The power you have in defining the layout of a document in HTML (combined with css) with "simple" ASCII document is almost as powerful as what you can do with Word. But now try writing the same thing with a (self written) program. You know it is easy in HTML (certainly if you ever did some programming in e.g. PHP), but producing a Word document with a program is hard work[1].

If the document should include complex things like mathematical formulas that are layed out properly, it becomes difficult in HTML too.

Enter LaTeX.

2.1 Multiperson literal work

One of the much overlooked advantages of LaTeX(and of any multi-file source code application like java projects) is that the fact that you can split up the document into multiple but still coherent parts. This fact allows you to work on one final document with multiple persons as in team members.

This make LaTeX almost ideal for project work in which several authors have to contribute to the final product and where source files are shared by means of a repository. Even more so in cases where you want to include (part of) program source code into the document to explain or show certain implementation aspects. Such source code is not copied and pasted but in stead read on the fly from the original file(s) during text processing. This allows you to always have the most up to date version.

As this sample is a multipart document, it can be used as a reference or a start for your own document.

[1] It becomes easier in modern version of word processing packages, they tend to use xml as their internal document format

Beautiful math is the purpose of TEX and LATEX.
Mantra of TEX-ies

3 Mathematics to show off

Here, you see how a mathematical equation can be generated in line, for instance $f(x) = \frac{1}{1+25x^2}$. The `$`-symbols enclose the formula. As a so-called displayed formula, it would look like

$$f(x) = \frac{1}{1+25x^2}.$$

It is customary that mathematical functions are *not* set in math-italics, so LATEX has the basic ones pre-defined; you should use the commands `\cos`, `\exp`, etc. to get $f_1(x) = \cos x$, $f_2(x) = -e^x \sin^2 x$, etc.

Here, I use some of my commands defined above: I like $\varepsilon = \varepsilon$ better than the default ϵ. A partial derivative (with 2 arguments) would be obtained as follows. If $f(x, y) = x^2y^3$, then

$$\frac{\partial f}{\partial x} = 2xy^3, \quad \frac{\partial f}{\partial y} = 3x^2y^2.$$

3.1 Sums and Integrals

When you say "capital sigma," you probably did not really mean Σ, but rather a summation symbol. You would get that as in

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \quad \text{for all } |r| < 1.$$

Finally, we have

$$\int_0^1 \sin(2\pi x)\, dx = 0$$

and

$$\iint f(x)g(y)\, dx\, dy = \int f(x)\, dx \int g(y) dy.$$

Here, `\,` gives a small space, while `\!` forces things closer together; you have to work on the proper spacing for integrals, as LATEX does not understand, what is going on.

3.2 Matrices in LaTeX

A matrix $A \in \mathrm{R}^{m \times n}$ could be defined by

$$A = \begin{pmatrix} 11 & 12 & 13 & \cdots & 1n \\ 21 & 22 & 23 & \cdots & 2n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m1 & m2 & m3 & \cdots & mn \end{pmatrix}$$

Here, the word `dots` in the commands stands for an ellipsis (i.e., three dots) placed horizontally in the centre (`\cdots`), vertically (`\vdots`), or diagonally (`\ddots`); what is not mentioned is `\ldots` for horizontal dots at the baseline. Use the baseline or central version as appropriate, for instance

$$\begin{aligned} a_1, a_2, \ldots, a_n \quad &\text{and not} \quad a_1, a_2, \cdots, a_n, \\ a_1 + a_2 + \cdots + a_n \quad &\text{and not} \quad a_1 + a_2 + \ldots + a_n, \end{aligned}$$

Some more comments on the matrix are needed, I suppose: The `\left(` and `\right)` create the variable-sized parentheses around the actual array of terms. You can also use `\left[` and `\right]`, or `\left\{` and `\right\}` in other situations. The actual array arrangement is organised by the `array` environment; you need the arguments `ccccc` to indicate that there are five columns and you want the entries centered ("c"), other options are left ("l") and right ("r"). Notice how `&` separate columns and `\\` the rows.

Here is another matrix example. A matrix multiply used with 3D graphics:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A picture is worth 1000 words

Anonymous

4 Graphics as easy as pie

LaTeX, combined with pdf in `pdflatex`, supports the following graphic file types: pdf, png, jpeg or jpg and gif in that order of preference. Using the vector format pdf gives the added benefit that the graphic file can be scaled up and down without loss of quality. If you want to include bitmaps, try to get them in png format which is open and patent free. It has the advantage over jpeg or jpg that is is loss-less, so you do not see any artifact if you blow them up in your inclusion. Converting back from jpg to png is useless, because the damage is already done in the jpeg format. JPEG is excellent for photographs. For all the bitmap formats: try to get them at the intended size with a resolution of 300dpi for printouts. 75 dpi is acceptable for screen reading.

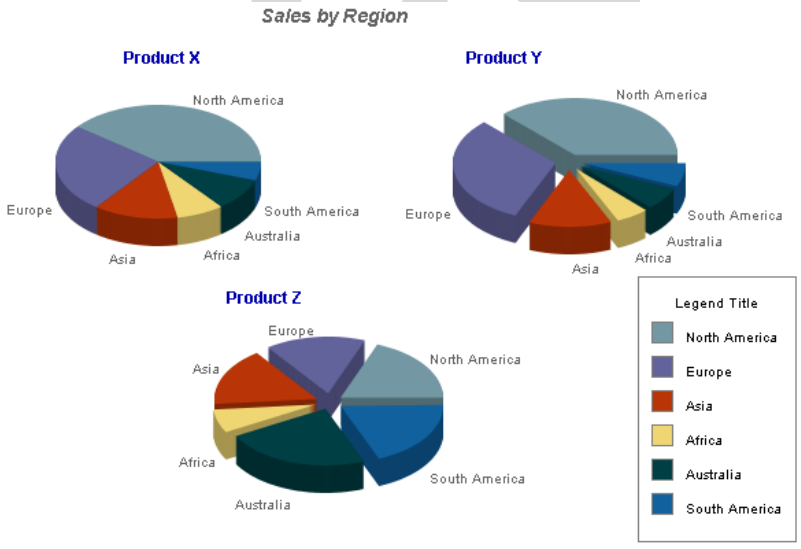

Figure 4.1: Stolen from the net. Google for a pie chart...

Many graphics packages can produce pdf files Embedded postscript (file extension .eps) are also a good candidate, after converting them to pdf with the epstopdf tool. By the way: the native format of Adobe Illustrator (ai) is similar enough to eps, so that too can be processed with ps2pdf. Programs like *Visual Paradigm* are able to produce pdf files too. And sometimes open-office can lend a helping hand, for by cutting and pasting Windows graphics into a single page oodraw drawing, you can produce an very usable pdf file.

Bitmap file types like png and jpeg take up a lot of space in your final pdf document. Bitmap files take even more space if encoded into a pdf file. If at

all possible, stick to a vector format like eps or pdf (if necessary derived from eps files).

4.1 A png example

If latex cannot fit the diagram on this page (page 7), then you may find the diagram as figure 4.1 on the facing page. And as you can see, you can easily reference pages and figures.

4.2 PDF from an UML package

If the documentation you write is a design document of some software package, you may want to include design diagrams. No software engineering without a UML diagrams.... You can see one generated with "dia", a vector drawing program that understands the UML in figure 4.2 on the current page. This diagram is 'wrapped' in a **`wrapfigure`** environment, so the text may flow around it

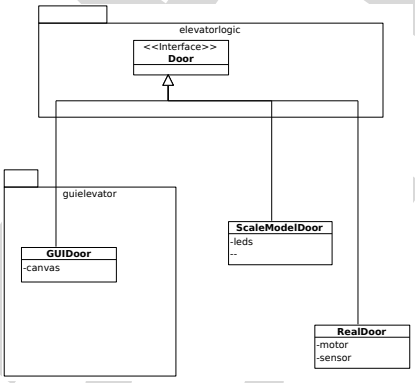

Figure 4.2: A class diagram made with Visual Paradigm

The diagram is not very sophisticated but shows an example of a vector format file included via an eps→ pdf conversion by epstopdf.

Open source programs like umlet and argouml are also able to produce vector format graphics files. And sometimes it is helpful to add a box that is a bit bigger then the picture you want to include. This ensures that the so called bounding box does not cut of any lines you want in your picture. Sometimes it is necessary to give these tools a helping hand with inkscape, that is do a bit of tinkering to get all details right.

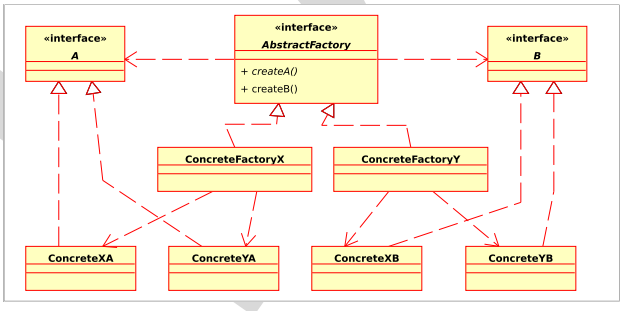

Figure 4.3: A class diagram by Umbrello, saved as svg and processed with inkscape.

Use the source Luke.

The most accurate documentation is in the source.

Addendum: Document your source well! And choose proper names.

Pieter van den Hombergh

5 Listings and code documentation

For these functions to work you need to use the `package` listings. See mydefs.tex for the inclusion.

Note that the line numbers in the right hand border are the line numbers in the included sources.

doxygen

A good advice is to start using *doxygen* for your code documentation. It can produce nicely formatted HTML and latex documents and can be tuned in various ways with the help of doxywizard. The way of working is a lot like using javadoc, but also works for C and C++ files. See e.g. the documentation on the zthreads package at `http://zthread.sourceforge.net` or Qt at `http://doc.trolltech.com/3.3/index.html`

5.1 Source code

The most simple case: include the whole thing with a command like

```
\lstinputlisting[language=java]{code/Hi.java}
```

```
/**
 * The obiquiteous Hello World program, a Java variant.
 * @author Pieter van den Hombergh (879417)
 */
public class Hi{
    @Override
    public String toString(){
        return "Hello world";
    }
    public static void main(String[] args){
        System.out.println(new Hi());
    }
}
```

Sometimes it is useful to include just a part of a file, for instance when you want to explain things. Like what line 11 is all about.

```
\lstinputlisting[language=java, firstline=11,lastline=11]{Hi.java}
```

Listing 5.1: use toString to let an object represent itself.

```
        System.out.println(new Hi());
```

5.2 Makefiles

You can also include make files. Note that makefiles have a peculiar syntax. *Spaces and tabs in Makefiles* are meaningful. That's why I made them show up in the next listing with the command

Spaces and tabs in Makefiles

```
\lstinputlisting[language=make,showtabs=true,
    showspaces=true,basicstyle={\ttfamily\scriptsize},
    numbers=right,language=make]{Makefile}
```

Spaces show up as ␣, tab characters as an extended version of the same thing (␣␣␣␣␣␣␣␣). As can be expected, spaces and tabs have no special meaning in makefile comments, the lines starting with a hash (#) sign. If you want to know more on makefiles try google or info:make in konqueror on a decently installed Linux box.

The make file for this entire document looks like this:

```
#
# Makefile for the latex documentation of this project
# @author Pieter van den Hombergh number 879417
# $Id: Makefile 320 2007-05-17 19:10:36Z hom $
# define some flags
LATEX=pdflatex
BIBTEX=biber
# bibtex
LATEXFLAGS=--interaction=batchmode
# How to produce the pdf from .tex files:
# run pdfelatex to
# get the references right.
# to process a Latex doc into a PDF document
%.pdf: %.tex
        $(LATEX) $(LATEXFLAGS) $<
        $(BIBTEX) $*
        $(LATEX) $(LATEXFLAGS) $<
        $(LATEX) $(LATEXFLAGS) $<

# first some definitions
TARGET=main.pdf
BARCODE=12345
PARTS=main.tex mydefs.tex intro.tex \
        motivation.tex \
        graphics.tex mathematics.tex \
        codelisting.tex Makefile code/hello.c figures/barcode.pdf \
        poser.tex frontpage.tex simple.tex programmedgraphics.tex \
        figures/servlet3.png figures/Diagram1.pdf figures/fon000_00c.pdf

# things that may be thrown away
DISPOSABLES= *~ *.aux *.log *.o hello *.blg *.bbl *.bcf *.lol *.run.xml *.fls *.lof *.out *.svt *.toc


# define the default target
all: main.pdf

main.pdf: main.tex $(PARTS)

#define a phony target
.PHONY: clean distclean

clean:
        rm -f $(DISPOSABLES)

distclean: clean
        rm -f $(TARGET) *.toc *.out *.url *.lof
```

Never start reading a difficult book at the wrong side.
It makes you feel stupid.

Private experience

6 So *you* know how to show off, but how do *I* start?

This document is indeed a bit of a showcase, but there is more to it. In essence, most documents are mainly texts. And those plain texts take mainly typing and not much more. The minimal, hello world style LaTeX file is not much longer[1] then the C classic.

```
\svnid{$Id: simple.tex 31 2013-09-08 10:54:29Z 879417 $}
\renewcommand\TheFile{simple.tex}
\documentclass{report}
\begin{document}
\chapter{Hi}
\pagestyle{empty}
  Hello world.
\end{document}
\end{lstlisting}
```

And the pictures, well they are made with other packages, and as long as those can produce a supported format, you can use them. TeX and LaTeX have an own drawing language, but explaining that would blow up this space. There is a lot of documentation available on TeX and LaTeX and I could recommend some good books on it. But you need not run off to the nearest book shop. Lots of documentation is on the Net, just as the TeX program suite itself.

Look for instance at `http://www.ntg.nl` for the Dutch TeX user group and `http://www.dante.de` for the German user group. They are both very alive and kicking.

A very good starting point nowadays is `http://en.wikibooks.org/wiki/LaTeX/`

6.1 My own definitions

Standard LaTeX output looks a bit dull but are nicely formatted nonetheless. If you want to make the most of your own style for your own or your projects documentation, put your style definitions into a separate file. That way you can keep all definitions of style and *macros* in one spot. This works especially nicely in an multi part document, so the other files (and authors) can concentrate on content.

macros

You can *define* your own macros in LaTeX. As an example a macro, `\define`,

define

[1] Counting headers too!

which I use to let words stand out at the outer margin. I use it at the first use of a word or concept in the text, to aid the reader in finding the definition. Of course this macro could be extended to put the word into an index. Which makes it a nice exercise. The macro is defined as follows:

```
\setlength\headheight{16pt}
\setlength\footskip{60pt}
%%
% This macro '\define' puts the argument in em
% and in boldface in the margin.
\newcommand{\define}[1]{% 1 argument
  \mbox{}{\textit{#1}}% italics or em
  \marginpar{\raggedright% no adjust
```

and is used as `\define{new word}`.

A drawing created with just a few words.
Or the old adagium 'a thousand words' reversed.

Private experience

7 Drawing in LaTeX

There are quite a few developers who add usefull packages to the TeXworld. One of them is Till Tantau of the "Technische Üniversität" in Berlin Germany. He produced the package called pgf, which stands for portable graphics format.

The pgf package contains the macro tool `tikz`, that allows you to draw quite nice graphics with just a few commands. Like this small ellipse: ⬭

The pgf manual has many nice examples that may prove usefull, certainly if you want to use the package beamer[1] to create professional looking pdf presentations with LaTeX.

A simple figure from the pgf manual:

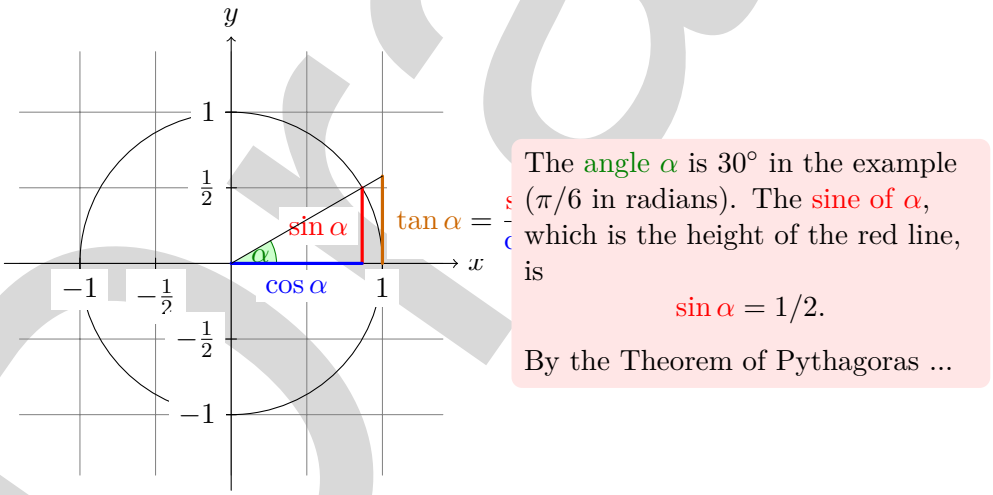

Figure 7.1: A drawing taken from the pgf manual/tutorial by Till Tantau

[1] also by Till Tantau

Everything should be made as simple as possible, but not simpler.

Albert Einstein

8 Citing is simple

Adding proper references and citations to you document can be a real burden. Not so in LaTeX where you can use the facilities of a kind "database" and many entries available on the web that be added to that database.

This database can be one file, but just as well a set of files, which you use to organize you bibliography. The files with these data are called .bib files.

8.1 bibtex

Using bibtex is easy. For instance if your bib contains this entry [bib14]

```
@misc{ Nobody06,
       author = "Nobody Jr",
       title = "My Article",
       year = "2006" }
```

Then citing Nobody [**Nobody06**] is easy as pie:\cite{Nobody06}

Then you need to add one compilation step to your normal workflow:

```
$ latex myarticle
$ bibtex myarticle
$ latex myarticle
$ latex myarticle
```

You can of course easily add that to the makefile introduced in an earlier chapter.

Bibtex is the standard you want to use if preparing an article of a journal or magazine. The journal typically also prescribes a specific bibliography style (defined in a .bst file), which is most likely already define for or by that journal

8.1.1 Biblatex and biber

A bit more modern is biblatex, which has a much simpler definition format for bst files. The is a separate **biber** program to do the processing instead of the bibtex run. I have used biber in this version of this latex sample.

Listings

References

[bib14] bibtex. *Home page BibTeX project*. web. 2014. URL: `http://www.bibtex.org/Using/`.

[Knu84] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[Lam86] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

[MG04] Frank Mittelbach and Michel Goossens. *The LaTeXCompanion Second Edition*. Addison-Wesley, 2004.

Fontys Hogeschool voor Techniek en Logistiek
Software Engineering/Business Informatics
Tegelseweg 255
5912 BG Venlo
The Netherlands

Fontys
Hogescholen
Fontys Hogeschool voor
Techniek en Logistiek


CAUTION
YOU HAVE REACHED THE
LAST PAGE
OF THE INTERNET
TURN OFF YOUR BROWSER AND GO BACK TO WORK
THERE'S NOTHING ELSE TO SEE HERE