Search
(courtesy of Google)

# THE PracTEX Journal

**The online journal of the TeX Users Group**
**ISSN 1556-6994**

## Current Issue 2010, Number 1

[Published 2010-06-08]

### About *The PracTeX Journal*

General information
Submit an item
Download style files
Copyright
Contact us
About RSS feeds XML HTML

### Archives of *The PracTeX Journal*

Back issues
Author index
Title index
BibTeX bibliography

### Next issue

Fall 2010

### Editorial board

**Lance Carnes, editor**
Kaja Christiansen
Peter Flom
Hans Hagen
Robin Laakso
Tristan Miller
Tim Null
Arthur Ogawa
Steve Peter
Yuri Robbers
Will Robertson

**Other key people**

**More key people wanted**

### Notices



### Articles

**Columns**

THE PracTEX Journal

# From the Editor: In this issue; Next issue: LaTeX for Teachers; Editorial: *LaTeX Work Bench*

Francisco Reinaldo


- Comment on this paper
- Send submission idea to editor




## In this issue

Since its first edition in 2005, the PracTeX Journal has accepted papers not only in English but also Portuguese, Spanish, Dutch, German, Norwegian, Chinese, Korean, Romanian, and Italian. Our mission is to provide the best possible quality of papers on the practical use of LaTeX and TeX for our readers. The gratifying response to the 2007-3 issue, and others, testifies to the readers' need to understand how to produce documents with LaTeX.

This PracTeX Journal 2010-1 issue has the theme "LaTeX Academic Work Bench". The goal of this issue is to present ideas on the use of LaTeX tools for education, teaching, and classroom purposes.

In the development of this issue, the strategy applied was to provide a detailed survival manual to captivate those who use LaTeX. The goal was to elaborate a basic plan for common users, but also focus on advanced learners. Detailed and elaborate theory was omitted, leaving only that which was necessary for user understanding, and enough detail to get the job done. Additionally, a variety of models were added following practical activities to improve students' LaTeX skills.

Continuing our mission to provide the best possible articles for learning/research, it is our pleasure to tell you that this new issue is the largest one ever produced. This issue consists of 18 articles with the latest news about development of the tools, and their use and effectiveness from basic to advanced learning levels. The issue is intend as a practical guide to the use of LaTeX resources, techniques and tools. Simple examples

of usage are given throughout. Brief explanations accompany the examples wherever necessary, and videos are available to illustrate selected topics.

We continue, as well, to publish our traditional columns: News from Around, Distractions, Ask Nelly, and Book Reviews. The *Travels in TeX Land* column, a regular feature since the Journal began, will make its last appearance in this issue.

In this issue, our authors submitted papers describing their experiences using LaTeX and TeX tools in an academic setting. If you are interested in beginning or improving the development and stylization of documents with higher typographical quality, then this issue is highly relevant to you. We have included articles that improve your understanding and introduce you to new tools and resources. Further, they present a LaTeX learning process that encourages the successful completion of your work.

Thank you and enjoy this brand new issue!

## Next issue: *LaTeX for Teachers*

Our intention is to select articles describing LaTeX tools useful for teachers: packages for constructing examinations, homework problem sets, textbooks, presentations and handouts, curriculum vitae, and others.

If you would like to write an article for the Journal send your idea or outline to the editors.

## Thanks

Many people have collaborated directly or indirectly to the success of this issue: the authors, particularly the ones who have worked with me in the revision process, the production editors, and the readers. They have discussed with me, suggesting all kinds of topics or helped me in the revision.

There is no way to write all the names, so I would like to mention Lance Carnes, Paul Blaga, Will Robertson, Yuri Robbers and David Walden for their intensive work in this issue; and especially Thomas E. Price and Lance Carnes who kindly donated two copies of their book "LaTeX Quick Start: A first guide to document preparation" as a gift for me and my students (LIC-UnilesteMG(BR)); and some authors with whom friendships were formed during the preparation of this issue, the most gratifying part of this work. This issue is dedicated to all of you!

Many thanks also to the reviewers and proofreaders who checked the articles and sent comments and corrections.

## Editorial: LaTeX Academic Work Bench

When I was invited by Lance Carnes to co-edit this issue with Paul Blaga, it was an

honour. I was fortunate to be able to work with Paul Blaga, who has many years of experience working with LaTeX in an academic setting. Myself, well, I am an enthusiastic guy and have been using LaTeX for about six years.

This issue was developed with the goal of demystifying LaTeX for the reader who is or would like to be in the LaTeX world. We decided to take a different approach and went beyond the bounds of traditional journals, and possibly broke some rules along the way.

This issue may look similar to previous ones, but in fact it is not. It has many differences that can be summarized in five main features. First, we had many students from my home country of Brazil submitting their work, and we hope this continues. Second, this issue is the first one that offers short videos to summarize the articles - we do not know of any other journal that has these. Third, our authors were instructed to produce papers almost in a step-by-step manner - it felt odd to require this, but it was for a noble cause! Fourth, in this issue you will find heterogeneous experiences brought by authors from at least eight countries: France, Germany, Italy, Brazil, Ireland, Belgium, USA, and Spain. For us, it was amazing collaborating with all of them. Lastly, this is the largest PracTeX Journal issue ever - a challenge we gave ourselves when we accepted Lance's invitation.

Francisco Reinaldo
Paul Blaga
Issue Editors 2010-1

---

THE PracTEX Journal

# News from Around: Knuth update, 1,000,000 math formulas

The Editors

- Comment on this paper
- Send submission idea to editor

## Knuth Volume 4 and Google talk

The first 30% of Donald Knuth's long awaited Volume 4 of *The Art of Computer Programming* is available in 5 small pieces: http://www.informit.com/store/product.aspx?isbn=0321637135

Here is a video of Knuth's March 16, 2009 visit to Google to discuss the interactions between faith and science http://www.youtube.com/watch?v=JPpk-1btGZk

## CSI Typeface

A valuable poster was put up for auction, but was found to be a phony. The tip-off was the typography.

http://www.boingboing.net/2009/06/16/1978-sex-pistols-pos.html

http://www.typophile.com/node/58945

## GuIT print journal

Our Italian friends have been busy, and published two print issues of ArsTeXnica this year.
http://www.guit.sssup.it/

## Greek TeX Journal

We are pleased to tell you that we have a new issue of Eutypon. Its contents are here:

http://www.eutypon.gr/eutypon/e-cont-22-23.html

Apostolos Syropoulos

## One million LaTeX math formulas

Springer, one of the largest scientific publishers, announces the launch of http://LaTeXSearch.com, a free online service which makes over 1 million LaTeX equations discoverable. LaTeXSearch.com lets users search for LaTeX code in one of three ways:

1) LaTeX Search: Directly typing an equation, in LaTeX, which they would like to find

2) Article title search: Where a user searches against all articles that contain LaTeX

3) Digital Object Identifier (DOI) search: For when a user knows which article they would like to see equations from.

LaTeXSearch.com is currently powered by a collection of over 120,000 peer- reviewed articles published by Springer, such as Inventiones Mathematicae, Acta Mathematica, and Mathematical Programming. One of the challenges facing readers of mathematics-intensive journals is that they have no way to access the code that generated the equations. LaTeXSearch.com solves this problem by giving users multiple ways to find equations and, with one click, access the LaTeX code that generated those equations.

Recognizing that LaTeX's flexibility means users might type the same equation in multiple different ways, Springer invested over eight months in engineering a process that normalizes LaTeX equations. This way, equations that are similar (or even exactly the same but written slightly different) will also appear under the "Similar Results" tab. This ability to provide a comprehensive set of results is unique to LaTeXSearch.com and will provide users with the most relevant results possible.

LaTeXSearch.com is actively soliciting the participation of other publishers and Open Access sources of LaTeX documents to make their equations searchable as well.

http://www.LaTeXSearch.com

## Asian Journal of TeX

I am pleased to announce that all the articles in The Asian Journal of TeX, Volume 3 are available to download from the following URL:

http://ajt.ktug.kr/2009/volume3.html

It is notable that this volume contains TeX articles written in three different languages, Korean, Japanese, and English.

Jin-Hwan Cho.

## Second Romanian TeX Conference

On September 8–9, 2010, there will be held, at Targu-Mures (Romania), the second *Transylvania TeX Conference*. The conference is organized by Paul Blaga, Sandor Horvath and Zoltan Kasa. The organizing institutions are the "Petru Maior" University and the "Sapientia" University from Targu-Mures and the "Babes-Bolyai" University from Cluj Napoca. Details about the program will be announced soon.

---

THE PracTEX Journal

# Some PDF/A tricks

Claudio Beccari

Abstract

This short contribution explains how to fix some font problems when creating PDF/A documents, the new standard for archival PDF documents.

Claudio Beccari is a long time LaTeX user and in 1991 wrote a book in Italian with the title "LaTeX — Guida a un sistema di editoria elettronica". Since then he is considered one of the gurus of the Italian TeX Users, even if this fame is totally undeserved. He has contributed several papers to TUGboat; he produced the hyphenation patterns for Italian and Latin, still in use today; he designed the default Greek fonts for use with Babel and supplied the Greek hyphenation patterns (luckily enough the Greek Users produced better patterns and these replaced Claudio's). Presently the Italian language definition file of the Babel package, although under the full control of Johannes Braams, is regularly updated and enriched by Claudio. He has participated in various TeX conferences and is a memeber of TUG GuIT, the official Group of the Italian TeX Users. He can be reached at claudio.beccari@polito.it

- PDF version of paper
- Article source files
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/06/06

# Some PDF/A tricks

Claudio Beccari

Email claudio dot beccari at gmail dot com
Address Villarbassse (TO), Italy

Abstract This short contribution explains how to fix some font problems when creating PDF/A documents, the new standard for archival PDF documents.

## 1 Introduction

At the beginning of December 2008 the package pdfx was released on CTAN. During its gestation the developer, Hàn Thế Thành, set up a web site[1] where he described the work in progress and the solution to some of the encountered problems.

First of all, what is a PDF/A document? It is an electronic document that can be archived for an unlimited period of time while being readable exactly as when it was first typeset, in spite of the fact that, say, 50 years later the reading software that existed at the time of creation does not exist any more; the new software that will be developed shall conform to the ISO 19005-1:2005 regulation, and will ensure that the document has been typeset in conformance with the same regulation.

It's the existence of this regulation that assures readability from now on. This regulation therefore has to put some restrictions on the way the document is typeset, on how the fonts used (only outline fonts are accepted) are embedded into the document, on the kind of pictures included in it, and on the format of the *metadata* the document should contain, in order to be usable as an archivable object.

1. http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex

## 2 Problems with some fonts

Fonts pose some problems, especially the Computer Modern ones and those that derive from this collection. Specifically, it's the `cmsy`*xx* fonts that cause some problems.

The ISO regulation requires that all glyphs used in the document have a non-vanishing width; it is surprising to find that some glyphs pertaining to the `cmsy`*xx* fonts have zero width; specifically the glyph corresponding to `\not` and the tail of the special arrow created with the command `\mapsto`; this tail is made up of a zero width vertical small bar, `\mapstochar`, joined to a normal right-pointing arrow.

It's understandable that the `\not` glyph has zero width, as it has to be superimposed onto the other binary relation operators in order to negate them; the same does not appear (to me) to be true for the `\mapstochar` glyph. In any case, this is the situation and this situation must be corrected because, even if the documents print and/or display in a perfect way, it violates the ISO regulation and becomes non-archivable.

Since a PDF/A document may be generated by running `ghostscript` in a special way, with suitable options and configuration files, the font problem cannot be circumvented even by resorting to this auxiliary program.

## 3 A possible solution

Hàn Thế Thành proposes to solve this problem by creating suitable virtual fonts that map the `cmsy`*xx* fonts (or any other font with the same features) onto itself but with a different metric file that uses a non-vanishing width, say, 0.001 em; this width is non-zero and is acceptable to the ISO regulation. However, at this point another problem arises because the metric information is not consistent with what is stored in the scalable Type 1 font with the extension `.pfb`; this requires another adjustment of the `.pfb` file, in order to store the same width used in the metric file.

Hàn Thế Thành used this technique to overcome the described problems. I believe that fiddling with back and forth transformations of binary and ASCII files, metric and real or virtual property list files, is error-prone and may not be

suitable for users who are not able to make modifications at that level. Therefore, I propose the following macros that do not fiddle with anything.

My macros rely on a zero-width box that contains a non-zero-width glyph; the zero-width box is `\rlap`, a Plain TEX control sequence more or less equivalent to `\makebox[0pt][l]`; and this box overlaps its contents on top of whatever happens to be on its right.

The non-zero-width glyph replacing the `\not` oblique bar is the slash taken from the `cmmmi`*xx* fonts, the ones that contain the normal math italic letters. The non-zero-width glyph replacing the `\mapstochar` one is a rule that uses the LATEX command `\rule`.

Since typesetting mathematics implies four different styles, the correct style must be chosen for inserting the appropriately-sized replacement glyph in every mode; to this end, the primitive TEX command `\mathchoice` can be used and the trick is complete.

The two macros are the following:

```
% New \not command
\renewcommand*\not{\mathrel{\mathchoice
  {\rlap{$\displaystyle
       \mkern2.5mu\mathnormal{/}$}}%
  {\rlap{$\textstyle
       \mkern2.5mu\mathnormal{/}$}}%
  {\rlap{$\scriptstyle
       \mkern2.5mu\mathnormal{/}$}}%
  {\rlap{$\scriptscriptstyle
       \mkern2.5mu\mathnormal{/}$}}%
}}
%
% New \mapstochar command
\renewcommand\mapstochar{%
\mathrel{\mathchoice
   {\rlap{\rule[.05ex]{.1ex}{1ex}}\mkern-.5mu}%
   {\rlap{\rule[.05ex]{.1ex}{1ex}}\mkern-.5mu}%
   {\rlap{\rule[.035ex]{.08ex}{.75ex}}\mkern-.5mu}%
   {\rlap{\rule[.025ex]{.065ex}{.55ex}}\mkern-.5mu}%
}}
```

Of course the "magic" numbers inserted in either macro should be verified with any font different from the assumed `cmsy`*xx* and `cmmmi`*xx* fonts, although the values should remain approximately the same.

With these macros in place it is not necessary to touch any font file, either the `.tfm` metric one or the `.pfb` binary one.

What I hope is that the small corrections indicated by Hàn Thế Thành be introduced into the original fonts distributed with every TEX distribution, so that my macros become superfluous, and that the indicated problems with the `cmsy`*xx* fonts in connection with the archivable PDF/A format vanish at their very origin.

THE PracTEX Journal
TEX USERS GROUP

# Chemical structures with PPCHTeX

Alan Braslau

Abstract

Chemical formulas and chemical structures can be included in a LaTeX or a ConTeXt document easily using the PPCHTeX macros. We present here a simple introduction to their use. Additionally, a more extensive tutorial is available in the documentation of the package.

The author is a researcher at the French Commissariat à l'Énergie Atomique at Saclay, working in the field of fundamental research in condensed-matter physics. I study beautiful single-crystals of solid helium at very low temperatures as well as the mysteries of life through the biophysics of DNA.

I started typesetting before the development of TeX, even before the invention of laser printers, starting with nroff/troff. I then (reluctantly at first) switched to TeX, and then to LaTeX. In particular, I used the RevTeX macros, developed by the American Physical Society as a submission standard for many physics journals. In recent years, I have completely converted to using ConTeXt for my daily work and presentations, though I still sometimes use RevTeX for official submissions to many scientific journals. I am currently trying to finish work on a very long book (written in both French and English and typeset using ConTeXt) on Liquid Crystal polymorphism as studied by X-ray diffraction.

I own a 1957 Massy-Harris tractor that I use to cut hay, and commute 50 km to work daily by bicycle.

You can contact me by sending an email to alan.braslau@cea.fr.

- PDF version of paper
- Article source
- Answer to exercise
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No.1
Article revision 2010/05/23

# Chemical structures with ppchTEX

Alan Braslau

Email alan.braslau@cea.fr

Address Service de Physique de l'État Condensé
CEA/Saclay
Orme des Merisiers
91191 Gif-sur-Yvette cedex FRANCE

Abstract Chemical formulas and chemical structures can be included in a LaTeX or a ConTeXt document easily using the ppchTEX macros. We present here a simple introduction to their use. Additionally, a more extensive tutorial is available in the documentation of the package.

## 1 Introduction

In a scientific/academic environment, one may find the need to represent chemical formulas and draw chemical structures. This can be easily reached and integrated directly in a document by taking advantage of the beauty of TEX output and its logical construction!

There are many commercial applications that are extensively used by chemists to produce graphics that can be included in a TEX document using standard external graphics macros. CTAN contains a number of free macro packages aimed at writing chemical formulas, drawing reaction arrows, numbering reactions and even drawing structures such as `bpchem`, `chemarrow`, `chemcompounds`, `chemcono`, `chemsym`, `mhchem`, and others[1]. One of the most sophisticated packages is XyMTEX written by Shinsako Fujita, available from the XyMTeX homepage.

Whereas XyMTEX appears to be a great package, I was never able to successfully learn the complexity of it. Thus, this paper talks about a simpler yet very powerful macro package,

1. These macro packages are available in TEX Live under many distributions in the `texlive-science` package.

ppchTEX, that was developed for ConTEXt and may be successfully used under LATEX as well[2].

The goal here is not to present a tutorial, much less to provide a complete exposition of the possibilities of ppchTEX. Nevertheless, this article aims only at giving a small taste of how you can easily draw chemical structures using TEX.

A few details: Under LATEX, positioning is performed by using PICTEX and line drawing using `pstricks`; Under ConTEXt `MkII`, positioning is performed by using PICTEX and line drawing using METAPOST; Under ConTEXt `MkIV` under LuaTEX, the internals of ppchTEX have been re-written to be included natively as core macros and no longer relies on PICTEX, and both positioning and drawing are performed using METAPOST. This re-implementation has lead to a considerable improvement in the performance and flexibility of the macro programming, so I expect a continued development of its possibilities.

## Resources

- A more complete manual (currently being updated and re-written) can be found at:
  ppchTeX manual
- A presentation with many ppchTEX examples can be found at:
  ppchTeX examples
- Other resources can be found on the web, such as the ConTEXt wiki:
  ConTeXt Chemistry wiki
- Finally, a very nice set of examples of drawing amino-acids:
  drawing organic molecules in laTeX II - amino acids

# 2 Drawing a simple molecule

## 2▷1 A first example using LATEX2e

Let's draw a simple chemical structure. First, you need to load the following macro definitions[3]:

2. Indeed, it was through my learning to use ppchTEX under LATEX that I was first introduced to ConTEXt!
3. These macros are available in any ConTEXt distribution, such as from TEX Live.

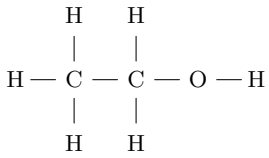
```
\documentclass{article}
\usepackage{m-ch-en} % English interface
% m-ch-nl for a Dutch interface.
% m-ch-de for a German interface.
```

The PICTEX macros are loaded automatically (via `\usepackage{m-pictex}`) if not called previously.

We conclude the example above by drawing the following simple molecule below:

```
\documentclass{article}
\usepackage{m-ch-en}
\begin{document}
\startchemical
\chemical [ONE,ZO357,SB1357,MOV1,ZO37,SB137,MOV1,ZO1,SB1]
          [C,H,H,H,C,H,H,O,H]
\stopchemical
\end{document}
```

Yielding $C_2H_5OH$:



which may make your head turn!

This code looks pretty complicated at first glance. But once you begin using ppchTEX, you should be able to write the above line of code without even looking at the documentation.

You will notice that the chemical structure was drawn between a `\startchemical-\stopchemical` pair, rather than within a proper `\begin{chemical}-\end{chemical}` LATEX environment. Whereas the ppchTEX package works perfectly well under LATEX, its syntax more closely matches that of plain TEX and ConTEXt.

For the rest of this article I will work with ConTEXt only. However, keep in mind that the syntax is identical under LATEX once the packages are loaded as described above. The macro `\chemical`, either alone or within a `\startchemical \stopchemical` pair, produces a chemical structure object (a TEX box) that can be used with LATEX or plain TEX.

## 2▷2 A first example using ConTEXt

For those unfamiliar with ConTEXt, I present an easy introduction. ConTEXt is currently distributed in two flavors: MkII that is compiled using pdfTEX and MkIV that uses LuaTEX. MkII is very stable and MkIV is under very active development. Nevertheless, it has now reached a point of stability that I find suitable for my daily production work, and that presents many advantages. The present article was typeset using ConTEXt MkII, using the macro set in common with LATEX.
ConTEXt is distributed with TEX Live and may already be installed along with TEX. The most up-to-date version is available through the ConTeXt garden. A ConTEXt source file is processed using the scripts `texexec` or `context`. The first is a `Ruby` script that, by default, uses the `MkII` macros; the second is pure LuaTEX and defaults to the `MkIV` macros. Both handle the processing of multiple TEX runs, the automatic processing of METAPOST code, and the running of bibTEX, if necessary. Therefore, creating `filename.pdf` from a ConTEXt source `filename.tex` is as simple as typing:

```
texexec filename
```

or

```
context filename
```

In order to use ppchTEX, you may first need to load these macro definitions:

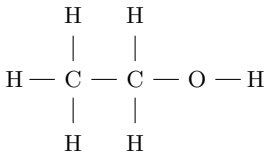
```
\usemodule [chemic]
\setupcolors [state=start]
```

PICTEX is automatically loaded in this case (ConTEXt `MkII`), as is the language variant. Under ConTEXt `MkIV` `\usemodule [chemic]` can be skipped altogether as the macros are included natively; indeed, it is ignored. `\setupcolors [state=start]` is also enabled by default in `MkIV` and it is probably a good idea to include this if using `MkII`. The order of the two lines is unimportant.

The example above can be completed by drawing the same simple molecule below:

```
\usemodule[chemic]
\starttext
\startchemical
```

```
        \chemical [ONE,Z0357,SB1357,MOV1,Z037,SB137,MOV1,Z01,SB1]
                  [C,H,H,H,C,H,H,O,H]
\stopchemical
\stoptext
```

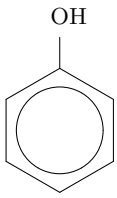
again yielding $C_2H_5OH$: 

Note that the chemical formula in the text above was written using `\chemical{C_2H_5OH}` ($C_2H_5OH$) which differs from `$C_2H_5OH$` ($C_2H_5OH$) both in their source-code typesetting and readability.

# 3 Drawing another simple chemical structure

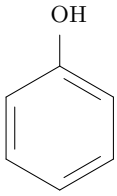
Another, perhaps, easier example is the chemical compound phenol, a six-atom ring structure that is very important in molecular biology.

```
\usemodule[chemic]
\starttext
\startchemical
        \chemical [SIX,B,C,R6,RZ6] [OH]
\stopchemical
\stoptext
```




A second representation is drawn using

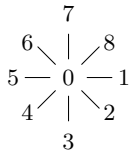
```
\usemodule[chemic]
\starttext
\startchemical
        \chemical [SIX,B,EB246,R6,RZ6] [OH]
\stopchemical
\stoptext
```



# 4 Chemical syntax

The syntax of the command `\chemical` has two optional arguments. The first argument defines the structure and the second presents substituents to be placed in the structure, namely atoms. This allows you to easily define a general molecular form that can be reused, including different substituents to draw different molecules. The inconvenience is keeping track of the correspondence between positions and substituents for a very complicated structure.

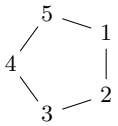
The basic molecular forms are `ONE`, `THREE`, `FOUR`, `FIVE`, `SIX` and `EIGHT`.[4]



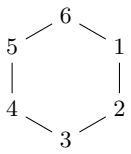

```
\chemical
[ONE,SB1..8,Z0..8]
[0,1,2,3,4,5,6,7,8]
```


```
\chemical
[THREE,SB,Z1..3]
[1,2,3]
```

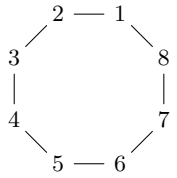

```
\chemical
[FOUR,SB,Z1..4]
[1,2,3,4]
```




```
\chemical
[FIVE,SB,Z1..5]
[1,2,3,4,5]
```



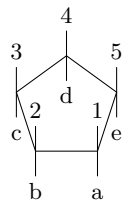

```
\chemical
[SIX,SB,Z1..6]
[1,2,3,4,5,6]
```



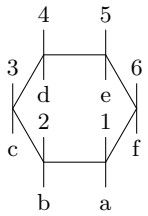

```
\chemical
[EIGHT,SB,Z1..8]
[1,2,3,4,5,6,7,8]
```

`FIVE` and `SIX` have `FRONT` variants:

4. Curiously, with ConTEXt `MkII` (or LaTeX) in this last form, `EIGHT`, the numbering of atoms runs counterclockwise rather than clockwise as for the other structures. This is, in fact, a bug that will probably *not* be fixed in order to maintain compatibility with existing documents. Under ConTEXt `MkIV`, the numbering of atoms runs clockwise coherently for all structures.



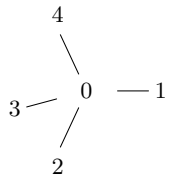

```
\chemical [FIVE,FRONT,B,R]
\chemical [+RZ1..5] [1,2,3,4,5]
\chemical [-RZ1..5] [a,b,c,d,e]
```



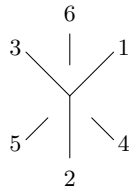

```
\chemical [SIX,FRONT,B,R]
\chemical [+RZ1..6] [1,2,3,4,5,6]
\chemical [-RZ1..6] [a,b,c,d,e,f]
```

The syntax is rather flexible and it can be broken down to multiple `\chemical` calls or else written as one long chain; both approaches are equivalent. For readability, I constructed the last molecules with three calls to `\chemical`.

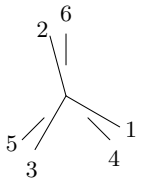
Further structures `CARBON`, `NEWMANSTAGGER` and `NEWMANECLIPSE` and `CHAIR` also are presented:



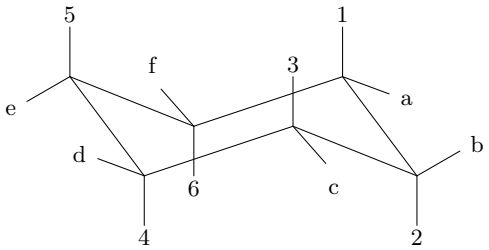

```
\chemical
[CARBON,B,Z0..4]
[0,1,2,3,4]
```




```
\chemical
[NEWMANSTAGGER,B,Z1..6]
[1,2,3,4,5,6]
```




```
\chemical
[NEWMANECLIPSE,B,Z1..6]
[1,2,3,4,5,6]
```




```
\chemical [CHAIR,B]
\chemical [+R,+RZ1,+RZ2,+RZ3,+RZ4,+RZ5,+RZ6] [1,2,3,4,5,6]
\chemical [-R,-RZ1,-RZ2,-RZ3,-RZ4,-RZ5,-RZ6] [a,b,c,d,e,f]
```

# 5 Bounding box

`\startchemical \stopchemical` creates a TEX box having a certain bounding box. By default, this is a fixed and standard size: a square of dimensions corresponding to four bond lengths centered on the molecular "origin". Often, one would like this box to bound the real extension of the molecular structure, and this can be obtained by setting the following options:

```
\setupchemical [width=fit,height=fit]
```

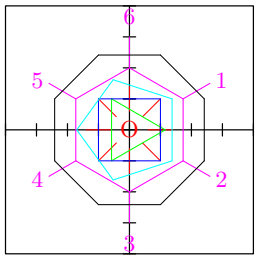
Alternatively, you can select options unique to each chemical structure drawn as:

```
\startchemical [width=fit,height=fit]
\stopchemical
```

Another option to be mentioned is `scale=small` that I have employed in this present article.

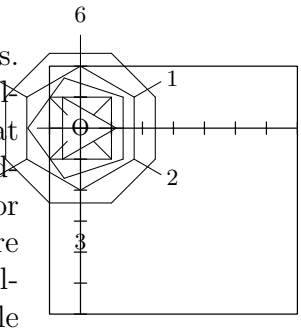
Let me better illustrate the size of the different chemical geometries[5]:

```
\usemodule [chemic]
\starttext
\startchemical [frame=on,axis=on]
        \chemical [ONE,SB,ZO] [O]
        \chemical [THREE,B]
        \chemical [FOUR,B]
        \chemical [FIVE,B]
        \chemical [SIX,B,R,RZ] [1,2,3,4,5,6]
        \chemical [EIGHT,B]
\stopchemical
\stoptext
```




5. In this illustration, in fact, I draw each structure using a different color by using the following options `rulecolor=` and `color=`. Each `\startchemical [rulecolor=...] \stopchemical` is then overlaid using a `collector`. I do not display the code here in order to avoid confusion, but you can find details by looking in the source code of this paper that is available for download.

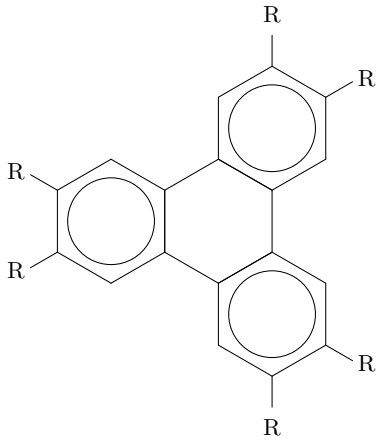
ppchTEX is programmed to use bond lengths of 1000 `chemical` units. Thus, the default options are `width=4000` and `height=4000`. Also available are the options `left=2000` and `top=2000` (values by default) that allows us to place the molecular origin with respect to a fixed bounding box (automatically they are adjusted when specifying `width=fit` or `height=fit`). To illustrate the bounding box, the same previous Figure is redrawn on the right exactly as shown in the code above (without colors), adding the options `left=500` and `top=1000`. Indeed, the molecule is here located partially outside of the bounding box.




# 6 Combinations

You also can combine basic forms and build complicated chemical structures, such as this disk-like molecule that forms columnar liquid-crystal mesophases.

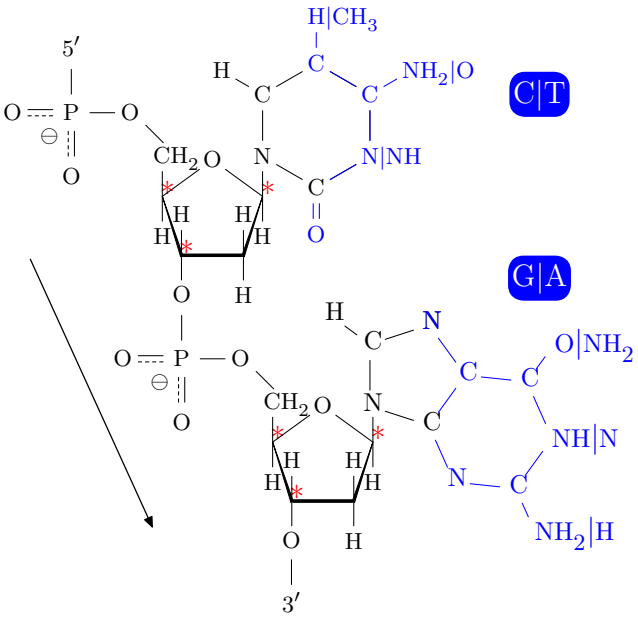
```
\startchemical
        \chemical [SIX,B,MOV2]
        \chemical [B,C,R23,RZ23,MOV5,MOV4] [R,R]
        \chemical [B,C,R45,RZ45,MOV1,MOV6] [R,R]
        \chemical [B,C,R61,RZ61] [R,R]
\stopchemical
```




Building combined structures can result in complicated forms, and I invite you to refer to the ppchTeX manual for further explanation.

# 7 Exercise for readers

Some structures appear to be extremely complicated to construct. So, I conclude this article with one very important, and not so easy, example as an exercise. See if you can figure out how it was done. Hint: this molecule was assembled like a jigsaw puzzle from individual pieces.




The primary structure of DNA

# 8 Acknowledgements

I would like to thank the creators of ppchTEX, Hans Hagen and Ton Otten, for providing this amazing tool and encourage them and others to continue its development.

THE PracTEX Journal

# Dual Screen Presentations with the LaTeX Beamer Class under X

Klaus Dohmen

Abstract

We show how the X Resize, Rotate and Reflect Extension of the X Window System can be used to display a LaTeX beamer presentation on one or two beamers while simultaneously displaying the output of both beamers on the lecturer's display. If only one beamer is used, the lecturer's display might show both the beamer output and hidden notes.

Klaus Dohmen is a professor of mathematics at Mittweida University of Applied Sciences in Mittweida, Germany. He holds a doctoral degree in mathematics from the University of Düsseldorf, and a habilitation degree in computer science from Humboldt-University Berlin. His primary research interests are combinatorics and graph theory. He has been an enthusiastic user of LaTeX since 1991 when he wrote his diploma thesis. Since then, he uses LaTeX to typeset articles, course materials, letters, and presentations, and encourages students to learn LaTeX early in their undergraduate studies.

You can contact him by sending an email to dohmen@hs-mittweida.de

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# Dual Screen Presentations with the LaTeX Beamer Class under X

Klaus Dohmen

Email dohmen@hs-mittweida.de
Website http://www.hs-mittweida.de/dohmen
address Hochschule Mittweida, University of Applied Sciences, Technikumplatz 17, 09648 Mittweida, Germany

Abstract We show how the X Resize, Rotate and Reflect Extension of the X Window System can be used to display a LaTeX beamer presentation on one or two beamers while simultaneously displaying the output of both beamers on the lecturer's display. If only one beamer is used, the lecturer's display might show both the output of the beamer and hidden notes.

## Contents



## 1 Introduction

In recent years, the LaTeX beamer class has become a very popular tool for creating PDF presentations [1]. An interesting feature of the beamer class is its ability to put additional material like notes, translations or previous slides on a secondary screen [1, pp. 187, 197–198]. This is accomplished by employing the *pgfpages*

package to create pages which are twice as wide as usual, but which have the same usual height.

The intention is that the left-hand side of each "double" page is directed to the first video output (the beamer) and the right-hand side to the second (the internal screen or a secondary beamer). To achieve this, the windowing system must be configured to use a virtual display, which is large enough to accomodate both screens. As an application, a lecturer might present slides on a beamer while reading notes not intended for the audience on his laptop (see Figure 1 for an example). Alternatively, he might present different slides on two beamers, e.g., the current and the previous slide, or slides in different languages.

Even if cloning of screens is supported by the windowing system, the lecturer is faced with the problem that he cannot see both the slides and the notes, respectively the slides on each of the two beamers, *simultaneously* in one place.

In this note, we present a solution to this problem. Our method relies on recent features of X.Org — the free open-source implementation of the X Window System. It works without configuring a virtual display.

The method was tested on a Lenovo W500 with Kubuntu 9.10 (X.Org version 7.4, XRandR version 1.3, Adobe Reader 9.2). The W500 has an ATI Mobility Radeon HD 3650 graphics card and a $1920 \times 1200$ TFT display.

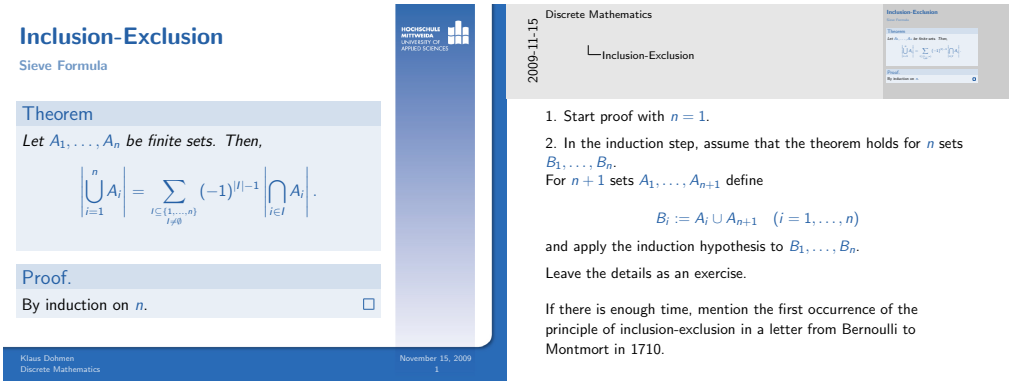



Figure 1: Only the left part of this double page is intended for the audience.

## 2 The X Resize, Rotate and Reflect Extension

Our method makes use of the X Resize, Rotate and Reflect Extension, which is abbreviated to XRandR [2]. The basic usage of XRandR is

`xrandr --output` *screen* `--mode` *width* `x` *height* `--scale` $s_x$`x`$s_y$ `--pos` $p_x$`x`$p_y$

For a default presentation on a beamer with $1024 \times 768$ pixels, one throws the command

```
xrandr --output LVDS1 --mode 1024x768 --output VGA1 --mode 1024x768
```

without scaling and positioning parameters.[1][2] In our case, VGA1 refers to the beamer, and LVDS1 to our laptop's internal display. For other video cards, it might be necessary to replace VGA1 by VGA or VGA-0, likewise for LVDS1.

To turn off the beamer, and to reset the laptop's internal display to its default maximum resolution, we use the command

```
xrandr --output LVDS1 --auto --output VGA1 --off
```

For more options and details on the usage of XRandR, the reader is referred to its manual page.

## 3 Standard presentations

The method of the preceding section requires that the internal display is capable of the beamer's resolution. While most laptops and desktop computers satisfy this constraint, there are good reasons to consider a more flexible approach. First of all, the user might decline changing the configuration of his primary display. In fact, changing the configuration is not necessary as we will see below. Second, small netbooks become more and more popular, which offer an even lower resolution than beamers. Another challenge arises from the fact that modern laptops and netbooks have an aspect ratio which is greater than $4 : 3$, which is the de-facto standard for beamers.

1. For an automatic configuration, first connect the beamer to the graphics port and then start X.
2. If you try out the commands in Section 3 and 4 and return to the present one, be sure to add `--scale 1x1 --pos 0x0 --fb 1024x768` to reset the scaling and positioning parameters.

A configuration leaving the resolution of the user's display unattached is obtained by invoking XRandR with appropriate scaling and positioning parameters. For the computation of these parameters, we introduce the following variables (see also Figure 2a on the following page):

$$
\begin{aligned}
b_x, b_y &= \text{width and height of the beamer's screen, } b_x : b_y = 4 : 3;\\
f_x, f_y &= \text{width and height of each frame in the user's display, } f_x : f_y = 4 : 3;\\
l_x, l_y &= \text{width and height of the user's display;}\\
p_x, p_y &= \text{coordinates of the (0,0) position on the beamer in the user's display;}\footnote{4}\\
s &= \text{horizontal and vertical scaling factor for the beamer.}
\end{aligned}
$$

Throughout, width and height are given in pixels. As a requirement, $l_x : l_y \geq 4 : 3$; otherwise, the method will not work. By this requirement and since the ratio of each LaTeX beamer frame is $4 : 3$, it follows that $f_y = l_y$ and hence,

$$f_x = \frac{4}{3} f_y = \frac{4}{3} l_y\,; \quad s = \frac{f_x}{b_x} = \frac{4 l_y}{3 b_x}\,.$$

Evidently,

$$p_x = \frac{l_x - f_x}{2} = \frac{1}{2} l_x - \frac{2}{3} l_y\,; \quad p_y = 0\,.$$

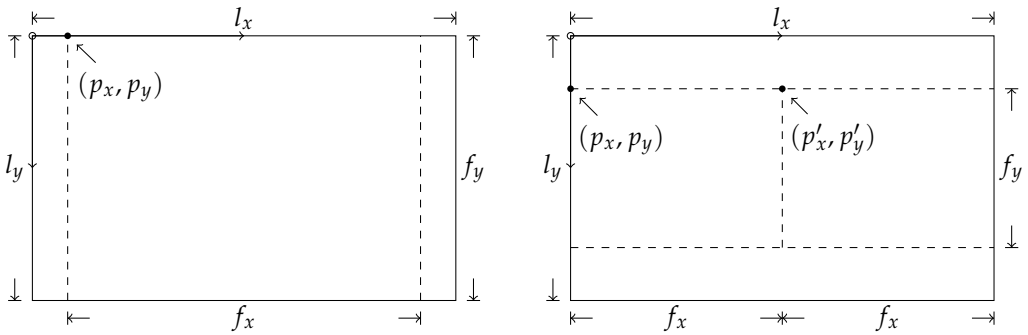
The initial requirement guarantees that $p_x \geq 0$.

As an example, consider a laptop with internal screen resolution $l_x = 1920$, $l_y = 1200$, which is connected to a beamer with resolution $b_x = 1024$, $b_y = 768$. We calculate $s = 1.5625$ and $p_x = 160$, so we invoke[5]

```
xrandr --output VGA1 --mode 1024x768 --scale 1.5625x1.5625 --pos 160x0
```

in order to display the LaTeX beamer frame both on the laptop and the beamer.

4. Note that, by convention, the (0,0) position of any screen is on its top left corner.
5. If you tried out the commands in Section 2, be sure to add "`--output LVDS1 --mode` $l_x$x$l_y$" to the invocation of `xrandr` in order to reset the resolution of the laptop's internal display.




(a) Standard presentation

(b) Presentation with notes or dual screen

Figure 2: Lengths and positions as defined on the preceding page.

# 4 Presentations with notes and dual screen presentations

In a dual screen presentation with the LaTeX beamer class, pages are twice as wide as usual, while having the usual height. The left-hand side of each double page is to be shown on a beamer, while the right-hand side (containing notes or other second mode material) is to be shown on the user's display or on a secondary beamer.

For a lecturer it is important to have all necessary information simultaneously in one place. So our intention is that *both* the left-hand side and the right-hand side of each double page is shown on the lecturer's display. Moreover, only the left-hand side of each double page is presented on the beamer, while the right-hand side remains hidden to the audience or is shown on a secondary beamer.

As single beamer frames have an aspect ratio of $4 : 3$, the ratio of a double page is $8 : 3$, so $l_x : l_y \leq 8 : 3$ is a requirement.

We use the same variables as in Section 3, with primed versions in case of a secondary beamer, see Figure 2b for details. Evidently, $2f_x = l_x$ and hence,

$$s = \frac{f_x}{b_x} = \frac{l_x}{2b_x}\,; \quad s' = \frac{f_x}{b'_x} = \frac{l_x}{2b'_x}\,.$$

Evidently, $p_x = 0$, $p'_x = l_x/2$, and

$$p_y = p'_y = \frac{l_y - f_y}{2} = \frac{1}{2}l_y - \frac{3}{8}f_x = \frac{1}{2}l_y - \frac{3}{16}l_x\,.$$

The initial requirement guarantees that both $p_y$ and $p'_y$ are non-negative.

As an example, let us again consider a laptop with internal resolution $l_x = 1920$, $l_y = 1200$, which is connected to a beamer with resolution $b_x = 1024$, $b_y = 768$. We calculate $s = 0.9375$ and $p_y = 240$, so we invoke

```
xrandr --output VGA1 --mode 1024x768 --scale 0.9375x0.9375 --pos 0x240
```

in order to display the left-hand side of each double page on the beamer.

If, in addition, the right-hand side is to be shown on a secondary beamer connected to, say VGA2 (e.g., by using an external graphics adapter), then since the $x$-coordinate of the left-upper point on the right-hand side is given by $p'_x = l_x/2 = 960$, the invocation becomes

```
xrandr --output VGA1 --mode 1024x768 --scale 0.9375x0.9375 --pos 0x240
     --output VGA2 --mode 1024x768 --scale 0.9375x0.9375 --pos 960x240
```

Eventually, the width, height and scaling factor have to be adapted for the secondary beamer. In some cases it might be necessary to try out positioning parameters close to the calculated values in order to improve the result.

# 5 Conclusion

We presented a method for showing LaTeX beamer frames on one or two beamers, where the output of both beamers, respectively the output of one beamer together with additional notes, are simultaneously visible on the lecturer's display. This is achieved by making use of XRandR — the X Resize, Rotate and Reflect Extension of the X Window System.

Dual screen presentations with two beamers, of course, require a machine having two output devices. Alternatively, one might consider an approach where the frames for the secondary beamer are sent via network connection to another machine to which the secondary beamer is connected to. In this case, the scaling and positioning parameters calculated in Section 4 can be used as well.

# References

[1] T. Tantau, *User's Guide to the Beamer Class*, Version 3.01. CTAN:macros/latex/contrib/beamer/doc/beameruserguide.pdf

[2] X.Org Foundation, *RandR Documentation*, http://www.x.org/wiki/Projects/XRandR.

THE PracTEX Journal
TEX USERS GROUP

# LaTeX e CSV (Italian)

Massimiliano Dominici

Abstract

English

In this paper we will present some techniques and a few examples about handling data in comma separated values. We will focus mainly on two packages specifically aimed at this purpose: datatool and pgfplots.

Italian

Questo articolo presenta alcune tecniche e alcuni esempi per gestire, all'interno di documenti LaTeX, dati organizzati in tabelle di comma separated values. In particolare l'attenzione verrà focalizzata su due pacchetti scritti appositamente per facilitare questo compito: datatool e pgfplots.

I have a background in physics, but for the past few years I have been working as a freelance consultant for digital typesetting, with a special interest in critical editions. I am also involved in the Italian TeX community: I became a member of GuIT in 2004, in 2006 was among the founders of ArsTeXnica and its first editor in chief. I have been serving as president of GuIT since October 2007.

You can contact me by sending an email to mlgdominici@interfree.it

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# LaTeX e CSV

Massimiliano Dominici

Email mlgdominici@interfree.it
Address Pisa, Italy

Abstract Questo articolo presenta alcune tecniche e alcuni esempi per gestire, all'interno di documenti LaTeX, dati organizzati in tabelle di *comma separated values*. In particolare l'attenzione verrà focalizzata su due pacchetti scritti appositamente per facilitare questo compito: datatool e pgfplots.

## 1 Introduzione

CSV (*comma separated values*) è un particolare formato di file adatto a rappresentare strutture di dati in forma tabulare. Un file CSV è a tutti gli effetti un file di testo, con alcuni vincoli che stabiliscono l'interpretazione del suo contenuto. Questi vincoli sono i seguenti:

1. I dati sono organizzati in *record* e ogni *record* è suddiviso in *campi* o *colonne*;
2. ogni *record* è delimitato da un'interruzione di riga (l'ultimo *record* può anche esserne privo);
3. la prima riga può contenere un'intestazione, ovvero il nome assegnato a ciascuna colonna;
4. ogni *campo* all'interno di un *record*, o dell'intestazione, deve essere separato da una virgola; il numero di campi deve essere lo stesso per ogni *record* e per l'intestazione; un campo può anche essere vuoto, ovvero non contenere dati;
5. gli spazi, anche iniziali e finali, fanno parte del campo e devono essere preservati;
6. i doppi apici (") funzionano da carattere di *escaping*: se un *campo* contiene virgole, interruzioni di riga o doppi apici, deve essere racchiuso tra doppi apici; i doppi apici contenuti in un *campo* devono sempre essere raddoppiati.

In base a quanto detto sopra, il codice seguente è una corretta rappresentazione del contenuto di un file CSV:

```
Autore,Titolo,Anno,Casa Editrice
C. E. Gadda,La meccanica,1958,"Garzanti, Milano"
```

A questo punto, sono però necessarie alcune precisazioni. La struttura di un file CSV non è mai stata formalizzata in uno standard. I vincoli descritti sopra sono quelli riportati in un documento informale dello IETF (International Engineering Task Force) per il tipo MIME `text/csv` [IETF, 2005]. Tuttavia non tutte le applicazioni che gestiscono file CSV implementano lo stesso insieme di specifiche: in particolare quasi tutte disattendono quanto stabilito al punto 5 e scartano automaticamente gli spazi a inizio e fine campo. Se si vogliono preservare questi spazi è quindi bene racchiudere il *campo* tra doppi apici. Inoltre quasi tutte le applicazioni sono in grado di usare (in lettura e scrittura) caratteri diversi per i delimitatori di *campo*. Per esempio "punto e virgola" o "TAB", al posto della virgola.[1]

Poiché un CSV (nel senso esteso specificato nella nota 1) è molto adatto a rappresentare dati di qualsiasi tipo in forma tabulare, i campi di applicazione in cui viene usato sono numerosi e di ambito differente. In particolare in questo articolo prenderemo in considerazione la gestione di semplici basi di dati e la manipolazione di tabelle di valori numerici provenienti da registrazioni di dati sperimentali o simulazioni numeriche. I dati organizzati nel CSV potranno essere stati scritti manualmente, o, più verosimilmente, provenire da un'applicazione. In quest'ultimo caso è probabile che si tratti di un gestore di basi di dati (Oracle, MySQL, PostgreSQL, ecc.), di un foglio di calcolo (MS Excel, OpenOffice Calc, ecc.) oppure di un programma per il calcolo numerico o simbolico (Matlab, Octave, Mathematica, Sagemath, ecc.).

1. In questo caso, bisognerebbe piuttosto parlare di "Fielded Text" (http://www.fieldedtext.org/). Ma siccome il "Fielded Text" copre casi più generali del CSV, e le varie implementazioni di quest'ultimo esistono ormai da molti anni, si continua a considerare CSV anche un insieme di dati separati da caratteri diversi dalla virgola.

# 2 Gestire semplici basi di dati

Il formato CSV si presta abbastanza bene a rappresentare semplici basi di dati di uso quotidiano in applicazioni da ufficio, come rubriche di indirizzi o registrazioni di punteggi scolastici. I dati possono essere compilati manualmente, oppure essere estratti da un foglio di calcolo o da una base di dati.

È bene far notare che il formato CSV è un formato "appiattito", ovvero tutti i dati sono contenuti in un'unica tabella. Chi ha dimestichezza con le basi di dati sa che invece, in genere, la loro struttura è più complessa, organizzata in diverse tabelle coordinate tra loro. Il modello più diffuso è il cosiddetto *Relational Database Management System* (RDBMS), alla base di numerosi prodotti commerciali, *open source* o *free software*. Non è questa la sede adatta per illustrare tale modello; basterà dire che di solito le applicazioni che implementano questo tipo di basi di dati offrono la possibilità di esportare in formato CSV, quindi in un'unica tabella, dati provenienti dalle varie tabelle interne, opportunamente organizzati (per esempio filtrati secondo le indicazioni passate al programma, o ordinati in base a una o più colonne scelte dall'utente).

Per quanto riguarda i fogli di calcolo, invece, i dati sono solitamente già "appiattiti" all'origine e potrà essere necessario, al limite, solo l'ordinamento in base ad una colonna data.

Lo strumento migliore per gestire un CSV di questo tipo all'interno di un documento LaTeX è il pacchetto datatool.

## 2.1 Generalità su datatool

Il pacchetto datatool [Talbot, 2009], successore di csvtools [Talbot, 2007], è stato disegnato per poter creare, modificare, caricare e gestire delle semplici basi di dati. Per semplice intendiamo sia "dalla struttura semplificata" (con tutti i dati, generalmente, contenuti in un'unica tabella) che di ridotte dimensioni. TeX infatti non è particolarmente efficiente nel gestire questo tipo di operazione.

Nel seguito dell'articolo ci occuperemo principalmente della gestione di file CSV creati esternamente, quindi non accenneremo, se non di sfuggita, ai comandi che si occupano dei primi due aspetti citati. Esamineremo, invece, nei dettagli quanto attiene agli ultimi due aspetti.

I primi due comandi che l'utente deve conoscere sono quelli relativi al caricamento e alla visualizzazione dei *database*.

```
\DTLloaddb[<opzioni>]
  {<nome_database>}{<nome_file>}
```

carica il *database* contenuto nel file `nome_file` assegnandogli il nome `nome_database`. Se non vengono specificate opzioni il comando assume implicitamente che la prima riga del file CSV contenga un'intestazione e di conseguenza contrassegnerà ogni colonna con la stringa contenuta nel corrispondente campo della prima riga. Le `<opzioni>` servono appunto a modificare questo comportamento:

*noheader* segnala che la prima riga non contiene l'intestazione ma valori normali;

*keys* acetta un insieme di valori, separati da virgole, e li usa per contrassegnare le varie colonne, sovrascrivendo quelle eventualmente presenti nell'intestazione;

*headers* acetta un insieme di valori, separati da virgole, e li usa per definire l'intestazione del *database* in fase di visualizzazione.

Immaginiamo di avere un ipotetico file `messaggi.csv` in cui sia riportato, mese per mese, il numero di messaggi postati nelle principali categorie del Forum GUIT nel periodo novembre 2008/febbraio 2009, ripartito per ogni singola categoria presa in considerazione:

```
TeX e LaTeX,235,212,289,321
ConTeXt,0,0,0,0
Altri programmi,24,18,19,31
Tipografia,13,11,18,22
Corsi e didattica,3,2,4,4
Edizioni critiche,2,0,5,3
```

Come si può vedere, il CSV non ha un'intestazione e, senza opzioni, `\DTLloaddb` interpreterebbe scorrettamente la prima riga. Il seguente comando, risolve il problema:

```
\DTLloaddb[noheader,
  keys={cat,nov,dic,gen,feb},
  headers={Categoria, Novembre 2008,
    Dicembre 2008,Gennaio 2009,
    Febbraio 2009}]{messaggi}{messaggi.csv}
```

| Categoria | Novembre 2008 | Dicembre 2008 | Gennaio 2009 | Febbraio 2009 |
|---|---|---|---|---|
| TEX e LATEX | 235 | 212 | 289 | 321 |
| ConTeXt | 0 | 0 | 0 | 0 |
| Altri programmi | 24 | 18 | 19 | 31 |
| Tipografia | 13 | 11 | 18 | 22 |
| Corsi e didattica | 3 | 2 | 4 | 4 |
| Edizioni critiche | 2 | 0 | 5 | 3 |

Figura 1: Visualizzazione automatica del *database* `messaggi`.

Il file `messaggi.csv` segue lo standard per quanto riguarda l'uso dei separatori di campo, ma, se così non fosse, datatool fornisce i comandi `\DTLsetseparator{`⟨*char*⟩`}` e `\DTLsettabseparator` per cambiarlo in maniera opportuna.[2] Può anche rendersi necessario impostare il carattere che delimita i campi (in mancanza di specifiche: "`"`"); è possibile farlo con il comando `\DTLsetdelimiter{`⟨*char*⟩`}`.

Infine, nel caso che il file CSV contenga caratteri che LATEX interpreta in maniera speciale (`$`, `&`, `%`, ecc.), il *database* va caricato con il comando `\DTLloadrawdb`, che ha la stessa sintassi e fa in modo che tali caratteri vengano interpretati correttamente.[3]

Per visualizzare in maniera, semplice il precedente database, è sufficiente dare l'istruzione

```
\DTLdisplaydb{messaggi}
```

che dispone i dati in una tabella, come quella riportata nella figura 1. In caso di *database* corposi, la tabella può essere particolarmente lunga e occupare più pagine. Sarà opportuno, allora, usare una longtable, tramite il comando

```
\DTLdisplaylongdb[<opzioni>]
  {<nome_database>}
```

dove le ⟨*opzioni*⟩ servono a impostare l'aspetto dei vari elementi dell'ambiente longtable.

Tornando alla figura 1, si può notare che le varie colonne sono allineate automaticamente in considerazione del tipo di dato contenuto. Il testo, per esempio,

2. Il primo dei due comandi serve per impostare il separatore a un carattere generico (per esempio "`;`"), il secondo nel caso speciale che il separatore sia il carattere `TAB`.
3. È perfettamente lecito introdurre nel file CSV comandi LATEX, che verranno correttamente interpretati, per cui, per esempio, il primo campo della prima riga del file `messaggi.csv` può essere riscritto così: "`\TeX{} e \LaTeX`".

è allineato a sinistra, mentre i numeri sono allineati a destra. datatool, infatti, è in grado di riconoscere quattro diversi tipi di dati: stringhe di testo, numeri interi, numeri reali, valuta.[4] L'utente può modificare gli allineamenti predifiniti usando comandi della forma `\dtl`⟨*tipo_dato*⟩`align`, dove ⟨*tipo_dato*⟩ è uno dei seguenti: `string`, `int`, `real`, `currency`. Molti altri aspetti della tabella così ottenuta possono essere modificati in maniera limitata (presenza di filetti orizzontali o verticali, aspetto dell'intestazione, applicazione di particolari istruzione a colonne contenenti un certo tipo di dato, ecc.). Tutto ciò è adeguatamente descritto nel manuale.

Questo modo di visualizzare un *database*, è comodo solo se non si deve operare in qualche modo sui dati, per organizzarli, filtrarli, manipolarli, ecc. A questo scopo è preferibile usare il comando `\DTLforeach`, che permette di scorrere tutti i *record* e operare sui campi ivi contenuti. La sua sintassi è la seguente:

```
\DTLforeach[<condzioni>]{<nome_database>}
  {<assegnazioni>}{<istruzioni>}
```

Le ⟨*condizioni*⟩ sono, appunto, opzioni di filtro da applicare ai dati (ad esempio ci interessa di visualizzare solo *record* che contengano, per un dato campo, valori maggiori di zero). Le ⟨*assegnazioni*⟩ costituiscono un modo, invece, per filtrare le colonne. Possiamo infatti essere interessati a visualizzare solo una parte dei campi riportati nel *database*; assegnando un particolare campo a un comando, possiamo richiamare il valore di quel campo, per ogni *record*, tramite il relativo comando. Le ⟨*istruzioni*⟩, infine, determinano ciò che vogliamo fare con i dati in questione.

Possiamo, ad esempio, ricostruire (approssimativamente) la tabella illustrata nella figura 1, con l'aggiunta di una ulteriore colonna con i totali, per mezzo del seguente codice:

```
\begin{tabular}{lrrrrr}
  \bfseries Categoria &
  \bfseries Novembre 2008 &
  \bfseries Dicembre 2008 &
```

4. datatool è inoltre corredato di una serie di comandi accessibili all'utente, che permettono di identificare, comparare e manipolare questi quattro tipi di dati. In questo articolo non prenderemo in esame tali comandi, se non dove è necessario per la comprensione degli esempi. Per una panoramica completa si rimanda al manuale [Talbot, 2009].

| Categoria | Novembre 2008 | Dicembre 2008 | Gennaio 2009 | Febbraio 2009 | Totale |
|---|---|---|---|---|---|
| TEX e LATEX | 235 | 212 | 289 | 321 | 1,057 |
| ConTeXt | 0 | 0 | 0 | 0 | 0 |
| Altri programmi | 24 | 18 | 19 | 31 | 92 |
| Tipografia | 13 | 11 | 18 | 22 | 64 |
| Corsi e didattica | 3 | 2 | 4 | 4 | 13 |
| Edizioni critiche | 2 | 0 | 5 | 3 | 10 |

Figura 2: Aggiunta di una colonna alla visualizzazione del *database* `messaggi`.

```
  \bfseries Gennaio 2009 &
  \bfseries Febbraio 2009 &
  \bfseries Totale%
  \DTLforeach*{messaggi}{%
    \cat=cat,\nov=nov,\dic=dic,%
    \gen=gen,\feb=feb}{%
    \\
    \cat \gdef\tot{0} &
    \nov \DTLgadd{\tot}{\nov}{\tot} &
    \dic \DTLgadd{\tot}{\dic}{\tot} &
    \gen \DTLgadd{\tot}{\gen}{\tot} &
    \feb \DTLgadd{\tot}{\feb}{\tot} &
    \tot
  }
\end{tabular}
```

Il risultato è visibile nella figura 2. Per generare i totali dell'ultima colonna, usiamo il comando `\DTLgadd`[5] i cui tre argomenti obbligatori rappresentano rispettivamente: il registro in cui viene immagazzinato il risultato della somma e i due addendi. All'inizio di ogni riga azzeriamo il valore del totale. Poiché ci troviamo all'interno di una cella l'assegnazione deve essere globale (`\gdef`). Si noterà che `\DTLforeach` è chiamato, in questo caso, nella versione con asterisco. Questo significa che il *database* viene usato in modalità di sola lettura. In precedenza, infatti, è stato già accennato che datatool permette anche di creare e modificare un *database*. Se si vuole essere sicuri che nessuna modifica venga apportata al file è bene usare la versione con asterisco dei comandi che manipolano i *database*.

A questo punto, avendo esaminato gli strumenti da usare, possiamo introdurre un paio di esempi concreti.

5. `\DTLgadd` fa parte di una libreria di comandi per effettuare calcoli aritmetici in virgola fissa inclusa in datatool e basata sul pacchetto fp.

## 2.2 Primo esempio: *mail merging*

Le maggiori distribuzioni di TEX contengono più di una classe disegnata per comporre lettere formali [Kohm and Morawski, 2009, Thompson, 2009, Lamport et al., 2008, Dekker, 2008, Mezzetti, 2006]. La maggior parte di queste forniscono anche meccanismi per produrre automaticamente le etichette con gli indirizzi, e qualcuna anche la possibilità di creare numerose istanze della stessa lettera verso diversi destinatari, il cui indirizzo viene estratto da un file esterno (*mail merging*). In caso queste funzionalità non venissero fornite dalla classe in questione, si possono usare pacchetti dedicati [Emmel, 2006, Rahtz et al., 2003, Veytsman, 1997, Braams, 1994].

Il problema è che, in ogni caso, il formato accettato da tutti questi pacchetti per il file che contiene la rubrica, non è compatibile con il formato CSV. In alcuni casi è possibile costringere l'applicazione in cui originariamente si trovano i dati ad esportare nel formato richiesto, ma in genere sarebbe più comodo poter lavorare con un file CSV. Grazie a datatool questo è possibile.

L'esempio che proporremo usa la classe guitletter. Questa è semplicemente una versione della classe standard letter personalizzata in modo da inserire nella testatina e nel piè di pagina il logo del GUIT e altre informazioni relative al gruppo.[6] Tutto ciò che si può fare con guitletter si può fare anche con letter. In particolare vedremo come preparare automaticamente una lettera di benvenuto nel gruppo per diversi destinatari e come generare automaticamente le etichette con gli indirizzi da apporre sulle buste.

Per comodità dividiamo in più parti l'analisi del codice, cominciando dal preambolo:

```
\documentclass[12pt,color]{guitletter}
\usepackage[italian]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{datatool}

\DTLloaddb{indirizzi}{indirizzi.csv}

\signature{Massimiliano Dominici}
\city{Pisa}
```

6. La classe non è disponibile pubblicamente.

```
\date{\today}
\subject{Adesione al \GuITtext}

\makelabels
```

Conosciamo già il comando `\DTLloaddb` che serve, in questo caso, a caricare la nostra rubrica degli indirizzi:

```
Nome,Cognome,Via,Numero,Città,Prov,CAP
Mario,Bianchi,Piazza Cavour,7,Novi Ligure,AL,15067
Carlo,Rossi,Via Roma,8,Noto,SR,96017
Franco,Verdi,Corso Garibaldi,3,Senigallia,AN,60017
```

Le altre istruzioni riguardano la composizione delle lettere e delle etichette: `\signature`, `\city`, `\today` e `\subject` fanno parte delle informazioni su mittente, oggetto, ecc., mentre `\makelabels` specifica che vogliamo anche stampare le etichette con gli indirizzi dei destinatari.

Il corpo del documento, è invece il seguente:

```
\begin{document}

\DTLforeach*{indirizzi}{%
  \Nome=Nome,\Cognome=Cognome,%
  \Via=Via,\Numero=Numero,%
  \Citta=Città,\Prov=Prov,\CAP=CAP}{%
\begin{letter}{\Nome\ \Cognome\\
               \Via, \Numero\\
               \CAP\ -- \Citta\ (\Prov)}

\opening{Caro socio,}

<testo della lettera>

\closing{Cordialmente,}

\end{letter}
}

\end{document}
```

Ogni singola lettera deve essere racchiusa all'interno di un ambiente letter. Mentre `\opening` e `\closing` si limitano ad inserire le formule di saluto, formattandole adeguatamente, le informazioni da variare per ciascuna lettera si trovano nell'argomento obbligatorio dell'ambiente. Proprio perché tali informazioni devono variare, le inseriamo sotto forma di *registri*. Di volta in volta, leggendo i vari *record* della rubrica, il comando `\DTLforeach` assegna a tali registri il valore del campo corrispondente, generando tre lettere (una per *record*).

Le etichette vengono generate automaticamente, al momento della chiamata `\end{document}`. L'istruzione `\makelabels` che abbiamo visto in precedenza, infatti, abilita la scrittura delle informazioni relative ai vari indirizzi sul file `.aux`, dove `\end{document}` può leggerle e poi provvedere a stamparle.

## 2.3 Secondo esempio: produrre schede bibliografiche

In questo secondo esempio mostreremo che è possibile, fino a un certo punto, operare con diversi file CSV come se fossero le tabelle di un *database* relazionale. Attenzione: questo tipo di approccio è, in genere, sconsigliabile, ed è preferibile, come detto in precedenza, organizzare appropriatamente i dati al momento dell'esportazione in CSV dall'applicazione in cui i dati sono contenuti, e lavorare quindi su un'unica tabella. Pianificare questa operazione in funzione di ciò che si vuole ottenere è fondamentale per avere buoni risultati.

Tuttavia, allo scopo di esplorare le potenzialità di datatool, e in considerazione del fatto che non sempre l'utente è in grado di organizzare da sé tali dati, ma può trovarsi nella condizione di dover lavorare su materiale fornito da altri, daremo anche un esempio dell'uso contemporaneo di più file CSV in relazione l'uno con l'altro.

Una possibile applicazione di questo principio è la costruzione di schede bibliografiche (per esempio per una biblioteca personale) a partire da tre file in cui siano contenute, rispettivamente, le informazioni riguardanti gli *autori*, le *case editrici* e i singoli *titoli*. Uno degli scopi di un *database* relazionale è quello di evitare la ridondanza dei dati: è più comodo, infatti, dover gestire le informazioni relative, ad esempio, a un autore in una singola collocazione e richiamare poi queste informazioni tramite un riferimento.

I primi due file CSV riportati nella figura 3 non contengono riferimenti a dati presenti in altri file, ma il terzo CSV, invece, contiene due campi (`RefAutore`

```
Id,Nome,Cognome
001,Carlo Emilio,Gadda
002,Ryunosuke,Akutagawa
003,Francesco,De Sanctis
004,Louis,Stevenson
005,Jan,Tschichold
006,Matthew P.,Shiel
007,Giorgio,Manganelli
008,Galileo,Galilei
009,,Erodoto
```

```
Id,Nome,Luogo
001,UTET,Torino
002,Garzanti,Milano
003,Adelphi,Milano,
004,TEA,Milano
005,BUR,Milano
006,Mondadori,Milano
007,Bompiani,Milano
008,Edizioni Sylvestre Bonnard,Milano
```

```
Codice,Titolo,RefAutore,Anno,RefEditrice,Note
001,Storia della letteratura italiana,003,2006,005,
002,Weir di Hermiston,004,2000,006,
003,Le storie,009,2006,001,2 volumi
004,La Meccanica,001,1999,002,
005,Un fulmine sul 220,001,2005,002,
006,La nube purpurea,006,2004,003,
007,Rashomon e altri racconti,002,2008,004,
008,Opere,008,2005,001,
009,Teatro,007,2008,007,
010,La forma del libro,005,2003,008,
```

Figura 3: File CSV rappresentanti tabelle di un database bibliografico.

e `RefEditrice`) che prevedono un riferimento agli altri due file. Il riferimento si effettua tramite il codice dell'autore, o della casa editrice corrispondenti, per come sono specificati nell'apposito campo del relativo file. Così, nella seconda riga di `titoli.csv`, il terzo campo (`RefAutore`) riporta il valore `003` che corrisponde, nel file `autori.csv` all'autore `Francesco de Sanctis`. Nella sesta riga di `titoli.csv`, il quinto campo (`RefEditrice`) riporta il valore `002` che corrisponde, nel file `case_editrici.csv` alla casa editrice `Garzanti`.

Con datatool è possibile usare filtri sui valori di tali campi per connettere i dati contenuti nei vari file CSV.

Il codice seguente produce come risultato una serie di schede bibliografiche (una per pagina), come quelle riportate nella figura 4.

```
\newcounter{scheda}
\setcounter{scheda}{0}

\DTLforeach*{autori}{%
  \AutId=Id,\Nome=Nome,\Cognome=Cognome}{%
  \DTLforeach*[\AutId=\RefAutore]{titoli}{%
    \RefAutore=RefAutore,%
    \RefEditore=RefEditrice,\Titolo=Titolo,%
    \Anno=Anno,\Note=Note,\Codice=Codice}{%
    \DTLforeach*[\EdId=\RefEditore]{editori}{%
      \EdId=Id,\NomeEditore=Nome,\Luogo=Luogo}{%
      \vspace*{\stretch{1}}
      \stepcounter{scheda}
      \centering
      \fbox{
        \begin{tabular}[t]{%
          >{\columncolor[gray]{.8}}r
          >{\raggedright\arraybackslash}p{3.5cm}
        }
        \multicolumn{2}{l}{Scheda n° \thescheda}\\
        \midrule
        Autore:  &
          \Nome
          \CheckEmpty{\Nome}{}{\ }
          \Cognome\\
        \midrule
        Titolo:  & \Titolo\\
        \midrule
        Editore: &
          \NomeEditore
          \CheckEmpty{\NomeEditore}{}
          {\CheckEmpty{\Luogo}{}{, }}
          \Luogo\\
        \midrule
        Anno:    & \Anno\\
        \midrule
        Note:    & \Note\\
        \midrule
        Codice:  & \Codice\\
        \midrule
        \end{tabular}
      }
```

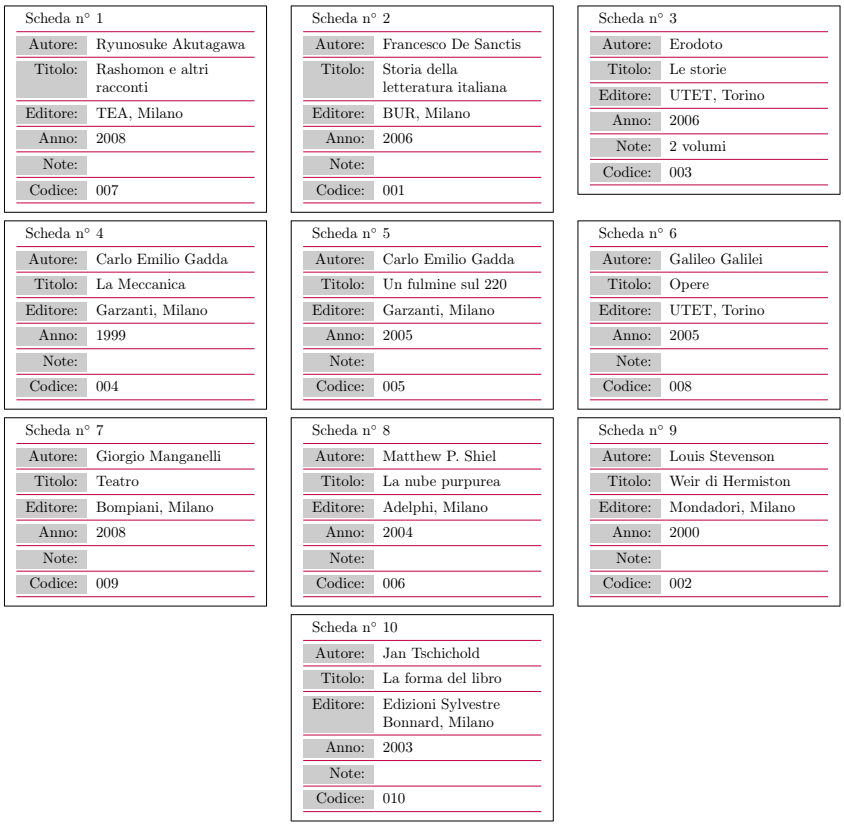



Figura 4: Schede bibliografiche.

```
      \vspace*{\stretch{1}}
      \clearpage
    }
  }
}
```

Il risultato è stato ottenuto annidando tre diverse chiamate a `\DTLforeach` (una per file CSV) e imponendo la condizione, nei cicli interni, che i valori dei campi di riferimento del file CSV principale (`titoli.csv`) coincidessero con il valore del campo di identificazione degli altri due file CSV. Ogni scheda ha un numero progressivo di identificazione, ottenuto con `\thescheda` che visualizza il valore corrente del contatore che tiene traccia dell'avanzamento nella "produzione" di schede.

Le schede sono ordinate per autore. datatool consente infatti di imporre ai *database* caricati un ordinamento, con il comando `\DTLsort`. Le impostazioni usate nel nostro caso sono state le seguenti:

```
\DTLloaddb{autori}{autori.csv}
\DTLloaddb{editori}{case_editrici.csv}
\DTLloaddb{titoli}{titoli.csv}

\DTLsort*{Cognome,Nome}{autori}
\DTLsort*{Nome,Luogo}{editori}
\DTLsort*{Titolo,Anno,Codice}{titoli}
```

Come si nota dal codice precedente, l'ordinamento può interessare anche più campi. Ovviamente, nel nostro caso, l'ordinamento principale è quello imposto dal ciclo più esterno.

Il comando `\CheckEmpty`, costruito a partire da *macro* fornite dal pacchetto datatool, serve ad evitare la presenza di spazi o punteggiatura spuri. La sua definizione è la seguente:

```
\newcommand\CheckEmpty[3]{%
  \ifthenelse{\DTLiseq{#1}{}}{#2}{#3}%
}
```

Infine, il codice sopra esposto richiede di caricare i pacchetti booktabs e colortbl per le rifiniture della tabella e i tocchi di colore.

# 3 Tabelle di dati numerici

Come abbiamo accennato nell'introduzione, i vari formati di esportazione sotto forma di matrici di valori, da parte di numerose applicazioni di calcolo numerico, possono essere assimilati, agli effetti pratici, a dei CSV. La loro utilità sta nel fatto che possono essere usati come formato di interscambio tra applicazioni diverse e, per ciò che ci riguarda più da vicino, per tracciare grafici con LaTeX.

Di norma i vari *sotware* che abbiamo citato nell'introduzione (vedi sezione 1) hanno anche la capacità di generare grafici (tramite funzionalità interne, o appoggiandosi a programmi come Gnuplot). Tuttavia la possibilità di poter controllare l'aspetto tipografico delle illustrazioni, in modo da avere una stretta integrazione con il testo è fondamentale per ottenere un risultato estetico di ottima qualità (si veda a questo proposito De Marco [2008]).

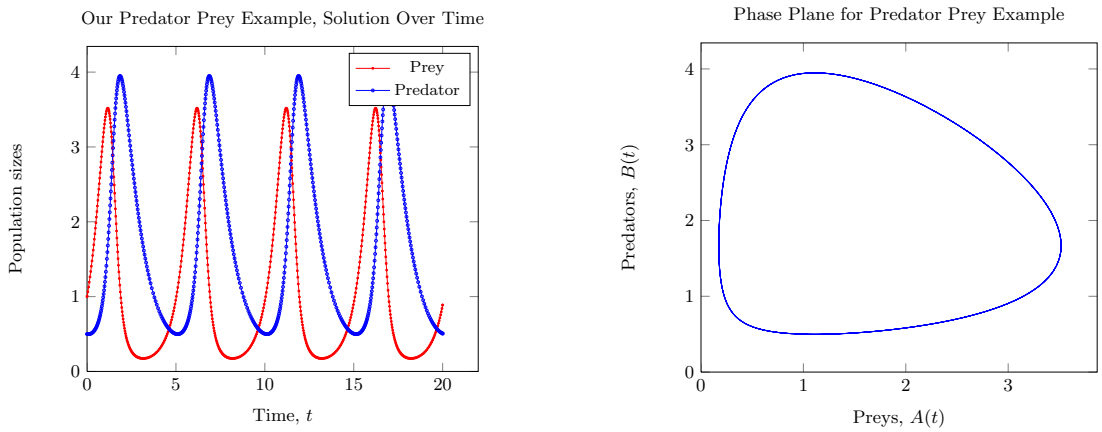



Figura 5: Equazioni di Lotka-Volterra

## 3.1 Il pacchetto pgfplots

Il pacchetto pgfplots [Feuersänger, 2009a] si appoggia al sistema grafico PGF/Ti*k*Z [Tantau, 2008] per tracciare grafici a partire da tabelle di coordinate che possono essere passate alle *macro* che si occupano del tracciamento secondo diverse modalità: esplicitamente, implicitamente sotto forma di espressione matematica, delegando a GNUPLOT[7] il calcolo esplicito, caricandole da un file esterno.

L'ambiente principale è axis, che traccia assi e griglia secondo le indicazioni dell'utente, e all'interno del quale è possibile aggiungere i singoli grafici tramite il comando `\addplot`. I tipi di grafici supportati sono numerosi e comprendono istogrammi, diagrammi di dispersione, diagrammi cumulativi, ecc.

Come tutti i pacchetti che si appoggiano a PGF, e come PGF stesso, l'uso di pgfplots è configurabile praticamente all'infinito. In questa sede ci limiteremo a mostrare solo alcune possibili modifiche alle impostazioni predefinite, rimandando ad un'attenta lettura del manuale per chi volesse acquistare dimestichezza con le varie opzioni.

## 3.2 Esempio: importare dati da GNU Octave

GNU Octave[8] è un programma *free software* per l'analisi numerica, parzialmente compatibile con Matlab e corredato di funzionalità per il calcolo matriciale. Il programma avvia una *shell* all'interno della quale è possibile passare le istruzioni all'interprete, il quale restituisce (implicitamente o esplicitamente) un *output* sotto forma matriciale. Le istruzioni possono essere anche preparate in un file esterno, sotto forma di funzioni o di *script*, ed essere poi richiamate dalla *shell*. I risultati ottenuti possono essere visualizzati sotto forma di grafici oppure esportati in file esterni sotto diversi formati. Quello che ci interessa è il formato ASCII. I valori vengono espressi in notazione scientifica, che pgfplots è in grado di comprendere, e organizzati in tabelle.

Questo, per esempio, è l'inizio di un file del genere:

```
0.00000000e+00 1.00000000e+00 5.00000000e-01
5.66993145e-03 1.00797044e+00 4.99726732e-01
4.51639430e-02 1.06532387e+00 4.98401027e-01
8.51085701e-02 1.12669546e+00 4.98125368e-01
1.25796018e-01 1.19282722e+00 4.99007906e-01
1.67235284e-01 1.26407660e+00 5.01190251e-01
2.06413960e-01 1.33517539e+00 5.04527388e-01
2.43682108e-01 1.40627123e+00 5.08935002e-01
```

Il file immagazzina i risultati di un'analisi numerica effettuata su equazioni di Lotka-Volterra (note anche come equazioni preda-predatore, in quanto forniscono un modello matematico per un ecosistema in cui interagiscono due specie animali: una come predatore, l'altra come preda).

Si noterà che il formato differisce da quello di un CSV standard: infatti il separatore di campo è in questo caso uno spazio (o più spazi successivi). Tuttavia, come premesso nell'introduzione, in questo articolo consideriamo un formato CSV in senso esteso (si veda la nota 1). pgfplots, d'altra parte, è in grado di interpretare correttamente anche CSV veri e propri ed altri formati con separatori non standard. Per una lista completa si consulti il manuale. Le righe precedute da # o % vengono automaticamente saltate (per mantenere compatibilità con file

7. http://www.gnuplot.info/.
8. http://www.gnu.org/software/octave/.

generati da GNUPLOT), e la prima riga viene considerata come intestazione che definisce i nomi delle colonne se contiene almeno un valore non numerico.[9]

Gli script di Octave usati per generare i dati sono i seguenti:

```
t0 = 0;
tf = 20;
init_vals = [1; 0.5];
options=odeset('AbsTol',1.e-12,'RelTol',
  1.e-9,'InitialStep',2,'MaxStep',2);
[t,x] = ode45(@pred_prey_odes,[t0,tf],
  init_vals,options);
A = [t,x];
save -ascii PredPrey.dat A

function deriv_vals = pred_prey_odes(t,x)
deriv_vals = zeros(size(x));
deriv_vals(1) = 2*x(1) - 1.2*x(1).*x(2);
deriv_vals(2) = -1*x(2) + 0.9*x(1).*x(2);
```

e sono stati adattati a partire da quelli presenti su questa pagina *web*: `http://paws.wcu.edu/emcnelis/NCSI_Houston08.html`. L'adattamento è consistito nel sostituire le istruzioni per generare i grafici con quelle per salvare i dati nel file `PredPrey.dat`.

Tali dati sono stati poi importati in un file `.tex`, in cui era stato precedentemente caricato il pacchetto pgfplots.

In primo luogo tracciamo il grafico con l'andamento delle soluzioni nel tempo:

```
\begin{tikzpicture}
  \begin{axis}[ymin=0,xmin=0,
      xlabel={Time, $t$},
      ylabel={Population sizes},
      title={Our Predator Prey Example,
        Solution Over Time},
      legend entries={Prey,Predator}]
    \addplot[red,mark=asterisk,
```

9. Appoggiandosi sul pacchetto pgfplotstable [Feuersänger, 2009b] per la lettura di tabelle di valori da file esterni, pgfplots riconosce i quattro tipi di dati che hanno senso per il primo: stringhe di caratteri alfanumerici, interi, reali e date. Tuttavia è in grado di usare realmente solo gli ultimi tre. Come si vede dall'esempio riportato sopra, i reali possono essere espressi anche in notazione scientifica.

```
        mark options={scale=.35}]
      plot table[x index=0,
        y index=1] {PredPrey.dat};
    \addplot[blue,mark=o,
        mark options={scale=.35}]
      plot table[x index=0,
        y index=2] {PredPrey.dat};
  \end{axis}
\end{tikzpicture}
```

I grafici vengono tracciati, come abbiamo detto sopra, tramite il comando `\addplot`. Poiché ci troviamo di fronte a un file che contiene più di due colonne di dati, dobbiamo aggiungere la specificazione `table`, che ci permette di scegliere di volta in volta le colonne da usare per le coordinate.[10] Le colonne possono essere identificate da una stringa in un'intestazione, oppure, se il file ne è privo, da un indice progressivo, come in questo caso. Scegliamo per il grafico della prima soluzione le colonne 1 e 2 (indici `0` e `1`) e per il grafico della seconda le colonne 1 e 3 (indici `0` e `2`). Le altre opzioni di `\addplot` servono a personalizzare l'aspetto visuale del grafico (colore, stile e dimensioni dei marcatori, ecc.). Altre personalizzazioni sono state effettuate sull'aspetto degli elementi "esterni" della figura: assi, etichette, legenda, e sono contenute nell'argomento opzionale dell'ambiente `axis`.

Tracciamo anche, in una illustrazione a parte, il diagramma nel piano delle fasi, scegliendo le colonne 2 e 3 (indici `1` e `2`):

```
\begin{tikzpicture}
  \begin{axis}[ymin=0,xmin=0,
      xlabel={Preys, $A(t)$},
      ylabel={Predators, $B(t)$},
      title={Phase Plane for Predator Prey Example},
      line width=.1pt]
    \addplot[blue,mark=none]
      table[x index=1,y index=2] {PredPrey.dat};
  \end{axis}
\end{tikzpicture}
```

In questo caso abbiamo omesso i marcatori (`mark=none`) e modificato lo spessore delle linee. Dopo aver aggiunto, nel preambolo, alcune impostazioni globali per cambiare le dimensioni dei caratteri

10. Altrimenti avremmo potuto usare la specificazione `file`.

```
\pgfplotsset{
  tick label style={font=\small},
  label style={font=\small},
  title style={font=\small},
  legend style={font=\footnotesize},
}
```

otteniamo finalmente le due illustrazioni contenute nella figura 5.

# 4 Raffronto tra datatool e pgfplots

Come si può arguire dalla scelta degli esempi, benché entrambi i pacchetti siano disegnati per gestire tabelle di dati organizzati in *record* e *campi*, essi sono tuttavia indirizzati a diversi ambiti di applicazioni. Questo nonostante entrambi, accanto alle funzionalità fondamentali (visualizzazione e manipolazione di basi di dati per datatool e tracciamento di grafici per pgfplots) consentano anche, tramite pacchetti "ancillari" distribuiti insieme a quello principale, di eseguire compiti secondari che finiscono per sovrapporsi. datatool, infatti, permette di visualizzare il contenuto di un *database* anche sotto la forma di un semplice grafico, o di un istogramma o di un diagramma circolare (grafico a torta), grazie ai pacchetti dataplot, databar e datapie. D'altra parte pgfplotstable può caricare, manipolare e visualizzare una tabella contenente dati di diverso tipo (testi, numeri interi e reali, date).

Il fattore discriminante nell'uso dell'uno o dell'altro pacchetto rimane il tipo di dati con i quali si deve lavorare e la loro organizzazione. Se i dati sono costituiti da matrici di valori numerici da visualizzare con grafici, pgfplots è la scelta più opportuna, scelta che diventa addirittura obbligata se i dati sono forniti in notazione scientifica, che datatool non è in grado di riconoscere. D'altra parte, se lo scopo è quello di manipolare e visualizzare dati di diverso tipo organizzati in un *database*, datatool risulta essere lo strumento più adatto, grazie alla flessibilità di comandi come `\DTLforeach`, alla libreria di funzioni per confrontare stringhe di caratteri di ogni tipo, e, conseguentemente, alla possibilità di applicare filtri ai dati da visualizzare.

# 5 Ringraziamenti e note

L'impulso iniziale a scrivere questo articolo lo devo a Maurizio W. Himmelmann e all'esigenza di trovare un sistema "comodo" per gestire in via TEXnica la corrispondenza del GUIT. Per i sensibili miglioramenti rispetto alla versione iniziale dell'articolo devo ringraziare l'anonimo recensore e la redazione di ArsTEXnica.

È superfluo far notare, infine, che tutti i dati esposti nella sezione 2, tranne quelli relativi alle schede bibliografiche, sono inventati di sana pianta e che ogni eventuale riferimento a fatti e persone realmente esistenti è puramente casuale.

# Riferimenti bibliografici

Johannes Braams. Creating a mailing. http://www.ctan.org/get/macros/latex/contrib/mailing/mailing.pdf, 1994.

Agostino De Marco. Gestione avanzata delle figure in LaTeX. *ArsTEXnica*, (6):10–27, Ottobre 2008. ISSN 1828-2350. URL http://www.guit.sssup.it/arstexnica.php.

Wybo Dekker. The isodoc class. http://www.ctan.org/get/macros/latex/contrib/isodoc/isodoc.pdf, 2008.

Thomas Emmel. Making labels, visiting cards, pins and flash-cards with LaTeX. http://www.ctan.org/get/macros/latex/contrib/ticket/doc/manual.pdf, 2006.

Christian Feuersänger. Manual for package PGFPLOTS. http://www.ctan.org/get/graphics/pgf/contrib/pgfplots/doc/latex/pgfplots/pgfplots.pdf, 2009a.

Christian Feuersänger. Manual for package PGFPLOTSTABLE. http://www.ctan.org/get/graphics/pgf/contrib/pgfplots/doc/latex/pgfplots/pgfplotstable.pdf, 2009b.

IETF. Rfc4180. http://tools.ietf.org/html/rfc4180, Ottobre 2005.

Markus Kohm and Jens-Uwe Morawski. Koma script. the guide. http://www.ctan.org/get/macros/latex/contrib/koma-script/scrguide.pdf, 2009.

Leslie Lamport, Frank Mittelbach, and Rainer Schöpf. Standard letter document class for LaTeX2e version. http://www.ctan.org/get/macros/latex/base/letter.dtx, 2008.

G. Mezzetti. The c.d.p. bundle. http://www.ctan.org/get/macros/latex/contrib/cdpbundl/cdpbundl.dtx, 2006.

Sebastian Rahtz, Leonor Barroca, Grant Gustafson, and Julian Gilbey. A package for making sticky labels in LaTeX. http://www.ctan.org/get/macros/latex/contrib/labels/labels.pdf, 2003.

Nicola Talbot. csvtools v1.24 : A LaTeX2e package providing access to data saved in a csv file. http://www.ctan.org/tex-archive/obsolete/macros/latex/contrib/csvtools/doc/csvtools.pdf, 2007.

Nicola Talbot. datatool v 2.02: Databases and data manipulation. http://www.ctan.org/get/macros/latex/contrib/datatool/datatool.pdf, 2009.

Till Tantau. The ti*k*z and PGF packages. manual for version 2.00. http://www.ctan.org/get/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf, 2008.

Paul A. Thompson. A new letter, fax, memo document class for LaTeX2e. http://www.ctan.org/get/macros/latex/contrib/newlfm/manual.pdf, 2009.

Boris Veytsman. Printing envelopes and labels in LaTeX2e: Envlab package. user guide. http://www.ctan.org/get/macros/latex/contrib/envlab/envlab.pdf, 1997.

THE PracTEX Journal

# About LaTeX tools that students of Logic should know (Portuguese)

Aracele Garcia and Arthur Buchsbaum

Abstract

English

In this article, we share our experience with PracTeX readers about LaTeX and the toolbox that the students of Formal Logic of the Master in Computer Science from the Federal University of Santa Catarina (UFSC) in Brazil are using to prepare handouts, books, articles, dissertations, and to solve exercises. We present some tools we have found useful for students who are developing projects in Formal Logic: a proof style, some useful sites, styles of numbering, and referencing of proclamations, references in BibTeX format and directions reading. The work done in this area requires a certain formality and rigor, and we believe that these can be successfully achieved by the use of LaTeX.

Português

Este artigo tem o objetivo de compartilhar a nossa experiência sobre LaTeX e apresentar a caixa de ferramentas que os alunos da disciplina de Lógica Formal do Mestrado em Ciência da Computação da Universidade Federal de Santa Catarina (UFSC) utilizam para elaborar apostilas, livros didáticos, artigos, dissertações de mestrado e resolver exercícios. Apresentamos algumas ferramentas que consideramos úteis para os alunos que estão desenvolvendo trabalhos na área de Lógica Formal: estilos de provas, sítios úteis, estilos de numeração e referenciação de proclamações, referências bibliográfias em formato BibTeX e indicações de leitura. Os trabalhos desenvolvidos nessa área necessitam de um certo formalismo e rigor; nós creditamos que tais características podem ser alcançadas através do LaTeX.

Aracele Garcia has a M.S. in Computer Science from the Department of Informatics and Statistics of the Federal University of Santa Catarina, Brazil. She was awarded her masters degree in the area of Formal Logic. During her studies her advisor was Arthur Buchsbaum, who encouraged her to learn LaTeX. For more details, please visit Aracele's site at http://www.inf.ufsc.br/~aracele, and her CV at http://lattes.cnpq.br/4653358157110108.

Arthur Buchsbaum is an Associate Professor of Logic and Discrete Mathematics at the Department of Informatics and Statistics of the Federal University of Santa Catarina, Brazil. He started using LaTeX in 1992, and gradually realized how important it is for composing well structured papers and books. He loves logic not just as an academic activity, but mainly as a powerful tool for learning how to think in a precise way, and to avoid being limited by preconceived ideas. Visit Arthur's professional site at http://www.inf.ufsc.br/~arthur.

You can contact the authors by sending an email to aracele.garcia@gmail.com or arthur@inf.ufsc.br

- PDF version of paper
- Article source
- Samples
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# Sobre as ferramentas em LaTeX que os estudantes de Lógica deveriam conhecer*

Aracele Garcia e Arthur Buchsbaum

Email aracele.garcia@gmail.com, arthur@inf.ufsc.br
Address Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis-SC
Brasil

Abstract In this article, we share our experience with PracTeX readers about LaTeX and the toolbox that students of Formal Logic of the Master in Computer Science from the Federal University of Santa Catarina (UFSC) in Brazil are using to prepare handouts, books, articles, dissertations and solving exercises. We present some tools we have found useful for students who are developing projects in Formal Logic: proof style, useful sites, styles of numbering and referencing of proclamations, references in BibTeX format and suggestions of reading. The work done in this area requires a certain formality and rigor, thus we believe that such features can be successfully aimed by the use of LaTeX.

## 1 Introdução

Os alunos do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina (UFSC) que optam por trabalhar com a linha de pesquisa em Inteligência Computacional devem cursar as disciplinas de Lógica Formal[1] I, II, III e Teoria dos Conjuntos. Nessas disciplinas eles estudam e desenvolvem atividades relacionadas com a lógica clássica e algumas lógicas deviantes nos níveis proposicional, quantificacional, equacional, descritiva e conjuntista.

*About LaTeX tools that students of Logic should know.

1. Esta disciplina também pode ser designada por Lógica Simbólica, Lógica para Computação, Lógica Matemática, Introdução à Lógica ou Lógica do Conhecimento Científico.

Eles são incentivados a elaborar diversos documentos, tais como artigos, livros, relatórios técnicos, apostilas e listas de exercícios resolvidos. Também recebem sugestões para a elaboração de seus trabalhos de conclusão de curso, os quais, no caso de orientandos do segundo autor, estão relacionados com algum tema relevante para a Lógica Formal. Para produzir esses documentos os alunos precisam familiarizar-se com LaTeX, pois através desta linguagem é possível apresentar diversos conteúdos com o devido rigor que áreas tais como a Lógica exigem.

Dessa forma, nós apresentamos as ferramentas e técnicas relacionadas ao LaTeX que normalmente utilizamos para a confecção de tais documentos.

# 2 Preparando o ambiente

Para obter uma estação de trabalho completa com LaTeX, utilizamos os seguintes sistemas:

- **MiKTeX**: uma implementação de LaTeX para computadores; é um freeware;
- **WinEdt**: uma shell (interface) para editar arquivos-texto na linguagem LaTeX; é um shareware, do qual o INE (Departamento de Informática e Estatística da UFSC) adquiriu uma cópia licenciada;
- **Adobe Reader**: um visualizador para documentos em pdf, que é um dos formatos usados para submissão de artigos; é um freeware;
- **Ghostscript** e **GSview**: para visualizar documentos em ps, que é outro dos formatos usados para submissão de artigos, o primeiro é o núcleo e o segundo a interface visual.

Recomendamos aos alunos instalarem esses sistemas na ordem *Adobe Reader, Ghostscript, GSview, MikTeX* e finalmente *WinEdt* (não confundir com `WinEdit`, que é um outro sistema).

# 3 Estudo do LaTeX

Os alunos são encorajados a estudar o LaTeX logo nas primeiras semanas de aulas da disciplina de Lógica Formal. Para isso, várias referências são apresentadas

com o objetivo de apoiá-los no primeiro contato com a parte básica da criação de textos.

Apresentamos abaixo uma pequena lista das referências indicadas:

1. "LaTeX2$\varepsilon$ Reference", de Tony Roberts;
2. "A Beginner's Introduction to Typesetting with LaTeX", de Peter Flynn;
3. "Essential LaTeX", de Jon Warbrick;
4. "LaTeX Maths and Graphics", de Tim Love;
5. "Short Math Guide to LaTeX", de Michael Downes;
6. "Essential Mathematical LaTeX2$\varepsilon$", de D. P. Carlisle.

# 4 A caixa de ferramentas dos alunos da disciplina de Lógica Formal

Todo aluno precisa conhecer um conjunto básico de ferramentas que possam ajudá-lo a se desenvolver como profissional e como pessoa.

Esta seção é um compêndio das ferramentas que podem auxiliar os alunos da área de Lógica Formal a prepararem documentos lógico-matemáticos adequados, acessíveis, de alta qualidade e precisos. As informações descritas fazem parte da nossa experiência de trabalho na disciplina de Lógica Formal do Mestrado em Ciência da Computação e contemplam várias informações que um lógico normalmente precisa saber em relação ao LaTeX: escrita de símbolos, desenvolvimento de provas lógicas, consulta de referências bibliográficas da área, estilos de numeração e referência, entre outras.

## 4.1 O pacote *turnstile*

É um pacote muito utilizado pelos lógicos para denotar uma relação de consequência, em uma dada Lógica, entre uma coleção de fórmulas e uma fórmula.

*Descrição*: Foi desenvolvido para desenhar a barra de Frege de várias formas e para colocar dados adicionais abaixo e acima desse símbolo, sempre que necessário. Ele fornece meios para desenhar a barra de Frege de uma forma mais exata que as disponíveis no LaTeX básico.

*Fonte*: Pode ser obtido nas versões para português e para inglês, em
`http://tug.ctan.org/tex-archive/macros/latex/contrib/turnstile/`.

*Exemplo*:

No preâmbulo do documento deve ser incluído o pacote `\usepackage{turnstile}`.

Podemos dizer que a fórmula $P$ é uma consequência lógica de uma coleção de fórmulas $\Gamma$ em uma certa lógica $\mathcal{L}$ através do seguinte código:

```
$\Gamma\sststile{\mathcal{L}}{}P$,
```

que resulta em $\Gamma \sststile{\mathcal{L}}{} P$ [1].

## 4.2 Métodos de Prova

Existem diversos métodos para provar ou verificar se um determinado sequente é correto ou incorreto: dedução natural, sistemas de Hilbert, resolução, tablôs[2], sequentes de Gentzen[3], entre outros. Esses métodos permitem demonstrar quando uma fórmula $P$ é consequência de um conjunto de premissas $\Gamma$, ou seja, que $\Gamma \vdash P$.

Nas aulas de Lógica Formal nós fazemos uso do método dos tablôs para demonstrar quando um dado sequente é incorreto e, através do tablô[4] desenvolvido, podemos obter um ou mais contra-exemplo(s)[5]. Adicionalmente, utilizamos provas por sequentes simples[6] conforme a notação de Fitch [2] para provar que um dado sequente simples é correto.

2. Do francês *tableaux*.
3. São sequentes cuja conclusão é uma coleção de fórmulas.
4. Do francês *tableau*.
5. É uma situação em que todas as premissas do sequente são verdadeiras e o seu consequente é falso.
6. São sequentes nos quais a conclusão é uma fórmula.

Abaixo apresentamos os pacotes que usamos para compor provas por tablôs e por sequentes simples.

## 4.3 O pacote *qtree*

Um *tablô* é uma árvore de fórmulas em uma dada lógica. Este método de prova por refutação consiste na geração de uma árvore (tablô) a partir do tablô inicial para o sequente a ser demonstrado. Esta árvore é a primeira de uma sequência de árvores e cada árvore não inicial sucede a anterior através da aplicação de uma regra de expansão a um nó não usado ou marcado[3]. Este processo pára quando for encontrado um tablô com todos os ramos fechados (então o sequente dado é correto), ou quando for encontrado um ramo aberto em que todos os nós não usados não forem aplicáveis a nenhuma regra (neste caso, o sequente é incorreto e pelo menos um contra-exemplo pode ser encontrado).

*Descrição*: Esse pacote é utilizado para desenhar diagramas de árvores. Nós o usamos para o desenvolvimento das provas via método dos tablôs.

*Fonte*: Pode ser obtido em `http://www.ling.upenn.edu/advice/latex/qtree`.

*Exemplo*:

No preâmbulo do documento deve ser incluído o uso do pacote *qtree*, `\usepackage{qtree}`.

Uma árvore inicia com o comando `\Tree` e a hierarquia é definida através de colchetes '[ ]'.

Como foi dito em 4.2, utilizamos o método dos tablôs para demonstrar que um sequente é incorreto. Nesse caso, o ambiente de demonstração é dividido em três colunas através do ambiente *multicols*[7]. A primeira coluna contém o tablô, a segunda contém o contra-exemplo, e a terceira a resposta final "*O sequente é incorreto*".

Além disso, utilizamos o comando `$\smile$`, o qual produz o símbolo $\smile$, para indicar que um ramo está fechado.

## 4.4 O pacote *fitch*

Para demonstrar que um sequente é correto nós utilizamos provas via cálculo de sequentes no estilo de Fitch [2].

7. É necessário utilizar o pacote *multicol*.

*Descrição*: É um pacote em LaTeX para codificar provas no estilo de Fitch. É bem fácil de usar e produz provas bem elegantes.

*Fonte*: Pode ser obtido em `http://folk.uio.no/johanw/FitchSty.html`.

*Exemplo*:

Primeiro é necessário incluir o pacote `\usepackage{fitch}` no preâmbulo.

A sintaxe básica é

```
\begin{equation*}
  \begin{fitch}
    ...
  \end{fitch}
\end{equation*}
```

Nós utilizamos os comandos principais `\fh` (usado como hipótese; exibe uma linha vertical com uma barra) e `\fa` (usado como passo padrão da prova; exibe uma linha vertical) diferentemente do que foi proposto pelo autor do pacote: o comando `\fa` é utilizado somente quando queremos continuar uma linha que foi iniciada dentro de um ambiente de suposição/hipótese.

Para esse tipo de prova o ambiente é dividido em duas colunas. A primeira contém a prova no estilo de Fitch e a segunda contém a resposta "*O sequente é correto*".

O resultado final é apresentado:

O sequente é correto.

| | | |
|---|---|---|
| 1 | $A \to B \vee C$ | pr |
| 2 | $\neg B$ | pr |
| 3 | $\vert\ A$ | sup |
| 4 | $\vert\ B \vee C$ | 3,1,MP |
| 5 | $\vert\ C$ | 4,2,SD |
| 6 | $A \to C$ | 3,5,RD |

## 4.5 Símbolos lógicos frequentemente utilizados

Diversos são os símbolos lógicos utilizados em nossos documentos. Apresentamos os principais:

Table 1: *Símbolos comumente utilizados*

| Símbolo | LaTeX | Descrição |
|---|---|---|
| $\rightleftharpoons$ | `\lsim` | Definição |
| $\mathcal{L}$ | `\mathcal{L}` | Uma certa Lógica |
| $\approx_c$ | `\approx_c` | Relação de congruência |
| $\Gamma$ | `\Gamma` | Coleção de Fórmulas |
| $\overrightarrow{\Psi}$ | `\overrightarrow{\Psi}` | Lista de Quantificadores |
| $\models_{\mathcal{L}}$ | `\sdtstile{\mathcal{L}}{}` | Consequência Semântica |
| $\vdash_{\mathcal{L}}$ | `\sststile{\mathrmbf{\mathcal{L}}}{}` | Consequência Sintática |
| $D(x\|t)$ | `D(x\vert t)` | Instanciação |
| $D(E\|\|G)$ | `D(E\Vert G)` | Substituição |

## 4.6 Referência e numeração de proclamações

Nos documentos que produzimos é comum a apresentação de proclamações, tais como teoremas, lemas, corolários, escólios, notações e definições.

Normalmente utilizamos as ideias abaixo para numerar e referenciar proclamações:

- Na numeração das proclamações só é dado o número da seção em que esta estiver localizada e o seu número de aparecimento na seção.
- Para referenciar proclamações no mesmo capítulo, só é dado o número da seção em que ela está contida e o seu número na seção, e, em um capítulo externo, é incluído o número do capítulo.

As seguintes ferramentas são utilizadas para que este estilo de numeração e referência de proclamações seja alcançado:

1. No preâmbulo do documento nós incluímos o pacote *smartref*, que pode ser obtido em http://www.ctan.org/tex-archive/macros/latex/contrib/smartref/. Esse pacote estende as capacidades do comando `\ref`.

   Ainda no preâmbulo, acrescentamos o comando abaixo para que o número do capítulo não apareça na numeração da seção:

   ```
   \renewcommand{\thesection}{\arabic{section}}
   ```

   Criamos um novo comando de referência, `\Ref`. Ele controla a exibição do capítulo quando a referência (ou citação) é feita fora do capítulo original da proclamação referenciada.

   ```
   %Utilizado em Referências (exceto referências às Páginas)
   \newcommand{\Ref}[1]
   {%
    \ischapterchanged{#1}% verifica se o número do capítulo mudou
    \hbox
    {%
     \ifchapterchanged% se o capítulo é diferente
     \chapterref{#1}.% acrescenta o capítulo na referência
     \fi % senão, não faz nada
     \ref{#1}%
    }%
   }
  \addtoreflist{chapter}
   ```

2. Na definição das proclamações, informamos que o número da seção será exibido na numeração:

   ```
   \newtheorem{theor}{Teorema}[section]
   \newtheorem{defn}[theor]{Defini\c{c}\~{a}o}
   ```

3. Finalmente, para referenciarmos uma proclamação, nós usamos o comando `\Ref` e o seu rótulo, como, por exemplo, `\Ref{teorema-legibilidade-unica}`.

Um resultado similar ao que normalmente alcançamos é exibido abaixo:

```
1 Introdução
1 Algumas motivações para o estudo da Lógica
Teorema 1.1
%Referenciando dentro do capítulo
[...] segundo o teorema 1.1
2 Lógica Proposicional Clássica
1 Uma linguagem para LPC
%Referenciando fora do capítulo
[...] conforme apresentado no teorema 1.1.1
```

## 4.7 Bibliografia rápida e prática

Normalmente utilizamos o *BibTEX* para formatar as referências bibliográficas que utilizamos em nossos documentos. Ele representa uma ferramenta e um formato de arquivo que são usados para descrever o processo e as listas de referências, principalmente em conjunto com os documentos no formato LaTeX [4].

Obtemos as referências no formato BibTEX através do portal "The Collection of Computer Science Bibliographies".

*Fonte*: Em http://liinwww.ira.uka.de/bibliography/index.html.

Este portal possui uma coleção de bibliografias em Ciência da Computação. As bibliografias estão agrupadas em subáreas. As principais são Inteligência Artificial, Lógica de Programação, Teoria e Fundamentos da Ciência da Computação, Engenharia de Software e Métodos Formais.

É possível pesquisar uma bibliografia através do título, ano de publicação e autor.

## 4.8 LaTeX para Lógicos

O sítio http://www.logicmatters.net é um importante guia de recursos para os lógicos que usam a linguagem LaTeX para produzir apresentações, materiais didáticos, teses ou livros e especialmente para aqueles que precisam incluir provas lógicas e diversos símbolos em seus textos. Ele possui várias informações sobre

símbolos lógicos, layout de proclamações, diversos pacotes de provas formais, diagramas para lógica e teoria das categorias, além de uma seção classificada como "`Miscellany`".

# 5 Conclusão

Além do conhecimento básico em LaTeX, é aconselhável que um aluno, ou qualquer outra pessoa interessada em Lógica Formal, consiga criar seus textos de forma adequada. E, além disso, os textos produzidos por tais pessoas certamente envolverão a criação de provas formais, inclusão de símbolos lógicos e outros.

O propósito desse texto foi apresentar a experiência dos autores na elaboração de documentos em LaTeX na área de Lógica e expor algumas das principais ferramentas ou técnicas utilizadas pelos alunos durante as aulas da disciplina de Lógica Formal em um curso de pós-graduação.

Esperamos que esse texto possa contribuir de alguma forma para aqueles que já trabalham na área de Lógica e ainda não possuem um conhecimento das ferramentas existentes para lógicos na linguagem LaTeX.

# References


[1] A. Buchsbaum and F. Reinaldo, "A tool for logicians", *The PracTeX Journal*, Aug. 2007.

[2] F. B. Fitch, "Symbolic Logic, An Introduction", *The Ronald Press Company*, 1952.

[3] A. Buchsbaum e T. Pequeno, "O Método dos Tableaux Generalizado e sua Aplicação ao Raciocínio Automático em Lógicas Não Clássicas", Revista "O que nos faz pensar", Cadernos do Departamento de Filosofia da PUC-Rio, v. 3, pgs. 81-96, 1990.

[4] *Your bibtex resource*
`www.bibtex.org`.

THE PracTEX Journal

# Continuous Integration in LaTeX

Marco Antonio Gomez-Martin and Pedro Pablo Gomez-Martin

Abstract

English

Have you ever co-written a paper using LaTeX together with some version control system such as SVN? Have you ever updated your local copy and the compilation has become broken due to a previous bad commit? Continuous integration avoids this problem using an auxiliary server that constantly checks the sanity of the repository, compiling the LaTeX documents for each commit and reporting possible problems. This paper describes how to configure this environment. The configuration effort is detailed but has to be done only once. Once in place, it provides other secondary advantages apart from the compilation tests: all authors can be automatically informed by e-mail when a new version is committed, and the current .pdf version can be made available to third parties on the Web.

Spanish

¿Has colaborado alguna vez en la escritura de un artículo utilizando LaTeX junto con algún sistema de control de versiones como SVN? ¿Has actualizado alguna vez tu copia local y ha dejado de ser compilable por culpa de un commit previo incorrecto? La integración contínua evita este problema utilizando un servidor auxiliar que comprueba constantemente la corrección de la copia oficial del repositorio, compilando los documentos LaTeX resultantes de cada commit y notificando posibles problemas. El presente artículo describe cómo configurar este entorno de trabajo. Si bien el esfuerzo de configuración inicial es innegable, proporciona la ventaja de comprobación automática de compilación así como otras ventajas adicionales: todos los autores pueden ser informados automáticamente por correo electrónico cuando se envía una nueva versión al servidor, y además la versión actual del fichero .pdf se puede hacer accesible via Web a terceras personas.

Pedro Pablo and Marco Antonio Gómez-Martín received their MS and PhD degrees in Computer Science in 2000 and 2008 from the University Complutense of Madrid. They have taught at the same university in the Master of Videogame Programming department since it first began in 2004. Their research involves the Artificial Intelligence and Software Engineering aspects of the development of games. This includes educational games, where they search for new methods to improve the way students learn, and strive to keep development costs under control. For their prototypes they use Continuous Integration as a development tool to ensure the quality of the code base. They use LaTeX for nearly all of their documentation, including papers, assignments, exams, and manuals.

You can contact them by sending emails to marcoa@fdi.ucm.es or pedrop@fdi.ucm.es

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# Continuous Integration in LaTeX

Marco Antonio Gómez-Martín and Pedro Pablo Gómez-Martín

Email marcoa@fdi.ucm.es, pedrop@fdi.ucm.es

Abstract Have you ever co-written a paper using LaTeX together with some version control system such as SVN? Have you ever updated your local copy and the compilation has become broken due to a previous bad commit? Continuous integration avoids this problem using an auxiliary server that constantly checks the sanity of the repository, compiling the LaTeX documents after each commit, and notifying authors of possible problems. This paper describes how to configure this environment. Although the configuration effort is detailed, it is done only once and provides many benefits. In addition to doing compilation tests, all authors can be automatically informed by e-mail when a new version is committed, and the current `.pdf` version can be made available to third parties on the Web.

## 1 Introduction

Many of the documents that we write using LaTeX are created not by just one author but two or more. In fact, our analysis performed over the database of the DBLP computer science bibliography[1] reveals that the average number of authors of books, journal articles and articles in proceedings written until February, 2004 was 2.26 (see table 1).

This justifies the use of some kind of version control system, such as CVS (2) or Subversion (4) (also known as SVN) for collaborative writing of LaTeX documents. As the use of SVN has spread over the LaTeX community, different packages have appeared that allow documents to incorporate references to the properties of the last revision (10). There are also different LaTeX editors that ease the task of using SVN (3).

When creating a document using this software, every author has a local copy of the source files where they add their contributions. Periodically, they *commit*

1. http://dblp.uni-trier.de/

| # of authors | # of contribs | Percentage |
|---|---|---|
| 1 | 150320 | 31.83% |
| 2 | 162546 | 34.42% |
| 3 | 93148 | 19.73% |
| 4 | 39371 | 8.34% |
| 5 | 14660 | 3.1% |
| More than 5 | 12160 | 2.58% |
| Total | 472205 | 100.0% |

Table 1: Numer of authors per contribution in DBLP until Feb, 2004

their changes to the server in order for other users/authors to be able to *update* their local copy with those changes. The result is a boost in the productivity because the coordination is much easier, something especially important when the deadline is near.

However, the use of this approach has a small issue: when an author commits changes, the build may be accidentally broken. In our experience, this usually happens when an author forgets to upload auxiliary files, such as images. Another source of problems is the difference between platforms: when the author is using Windows or Mac, where the file names are case insensitive, he may break the document generation of another author using Linux.

This issue is well known in the software development, where programmers have used version control systems for decades. One of the solutions they have found is called "*continuous integration*". As we will see in the next section, it consists of having a dedicated machine that continuously checks if there are changes in the SVN repository. When it detects a commit, it updates automatically its local copy and tries to build the application in development. If something is wrong with the commit, it sends an e-mail to the programmer that performed it, to kindly ask him to fix it, to avoid inconveniences to other developers.

In this paper, we extrapolate the idea into the LaTeX world. We will describe how to set up a server machine in order to use continuous integration in LaTeX projects. As we will see, this will bring us two other benefits: the machine may send an e-mail to every author of the document, which helps them to know its progress, and the generated document for every revision may be placed on a

web server for other authors (or reviewers) to download without forcing them to generate the PDF.

This paper runs as follows. The next section describes in detail the ideas behind the continuous integration concept. After that, we present our motivation for having a machine for the continuous integration of LaTeX projects. Section 4 describes the piece of software that we have used. Section 5 explains the steps that authors should take in order to allow the server machine to manage their projects. This is followed by Section 6 that describes how to set up the server. After that we present some advanced topics and ideas for users that need more control over the installation of the server. The paper ends with some conclusions.

Over the entire paper we assume that the reader is already writing their documents using Subversion. If this is not his/her case, you may find the continuous integration approach presented here useless. We encourage the use of SVN (or any other version control system). For more information, please consult (8) or (6).

## 2 Continuous Integration

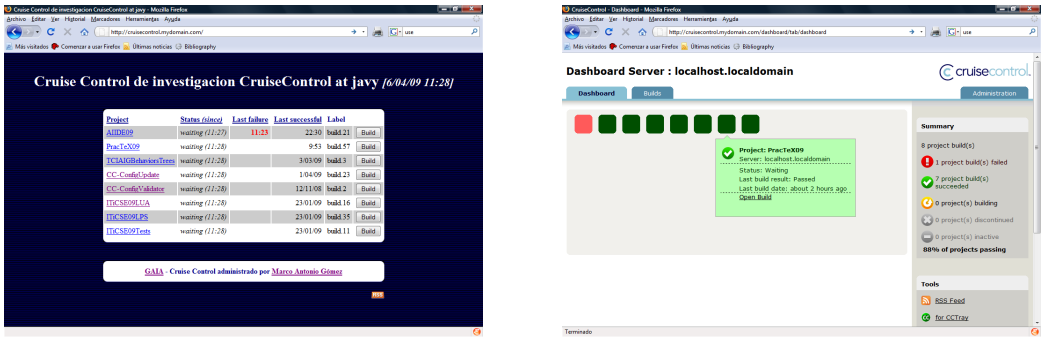
When a team uses, for any purpose, a version control system, some discrepancies can occur between the official content in the repositories and the local copies of the collaborators. When commits are done after each participant has made many local changes, problems such as file conflicts or incompatibilities can arise.

Software development teams, generally composed of many members, have dealt with this issue for decades. The inherent problems of the late integration caused (and still causes) many problems (and delays) when different software components had to be put together.

These difficulties are also present, in a smaller extent, when using a version control system while co-writing papers or documentation with LaTeX. Maybe the more common problem in this context is when someone forgets to commit a new file, typically an image, or when a file is incorrectly committed, preventing the rest of the team from compiling the new version.

A way to deal with all these problems is the known as *continuous integration*. Martin Fowler, a software engineering guru, defines it as (5):

> [It] is a software development practice where members of a team integrate their work frequently, usually each person integrates at least




(a) Cruise Control main page (b) Cruise Control dash board

Figure 1: Cruise control screenshots

daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

Therefore, *continuous integration* consists of encouraging developers to commit their changes as soon as possible to the mainstream, in order to provide fast feedback for developers.

An important aspect of this methodology is that an *infrastructure* is needed to allow programmers to build and test their changes as early and often as possible. The infrastructure will automatically do all these checks using the more recent project version when a new commit is done.

Fortunately, in recent years, continuous integration has had a major boost in the computer engineering area. This momentum has produced some tools for providing the previously mentioned infrastructure, some of them Open Source. We wondered if they could also be used to automatically test the commits into a papers' SVN, obviously a more reduced and simple environment than the bigger development teams. The next sections describe our decisions and experiences.

# 3 Motivation

One of the more important tasks of a research group is to generate papers that are usually written by at least two people. Today, many research groups have access

to server machines where reside different services, such as web or ftp sites. These servers can be used for a more "private" task: for hosting a version control system that helps the collaborative writing. A CVS or SVN server is almost mandatory for managing all this cooperative work.

Unfortunately, as said previously, version control systems do not prevent problems due to someone committing an invalid file. Software development teams use *continuous integration* to get over these difficulties. Our motivation was to implement in our research group the idea of using a new service for continuously testing that the different LaTeX projects were valid.

Figure 1 shows the result. The chosen continuous integration tool (described in the next section) provides two different interfaces to access its state. Both of them show six different papers[2], each of them forming a different project that is stored in an independent place in the SVN repository. At regular intervals, the continuous integration server checks each paper for modifications, and tests its validity. As an example, the figure shows that one of the papers has a compilation problem (the one called `AIIDE09`). The tool lets the user to browse into the error and shows him/her the error log generated by LaTeX during the failed generation.

As we will describe later in Section 6, our server machine is configured in such a way that:

- When a new commit is detected, the server notifies *all* the paper co-authors. This lets the collaborators remain informed about the evolution of other parts of the paper without *polling* the version control server or asking the other authors.
- The server publishes the generated document (usually `.pdf`) that will be accessible via the Web. This becomes quite useful when the paper must be read by external people, such as reviewers or advisors.

When starting a new paper, one of the authors will make the first commit into the SVN repository. After that, the continuous integration server administrator will add the new project so that the paper can be monitored. When new commits are done, the tool will test their validity, and send e-mails and show results through the web interface.

2. `CC-ConfigValidator` and `CC-ConfigUpdate` are not papers but projects related to the configuration of the software, as described in Section 7.

We decided to use Cruise Control as our continuous integration tool. The next section describes it, and then illustrates how the configuration process is done.

# 4 Cruise Control

Cruise Control[3] is one of the more well-known continuous integration tools. It is distributed under a BSD-style license and is free to use. Its main functions are, in fact, quite simple:

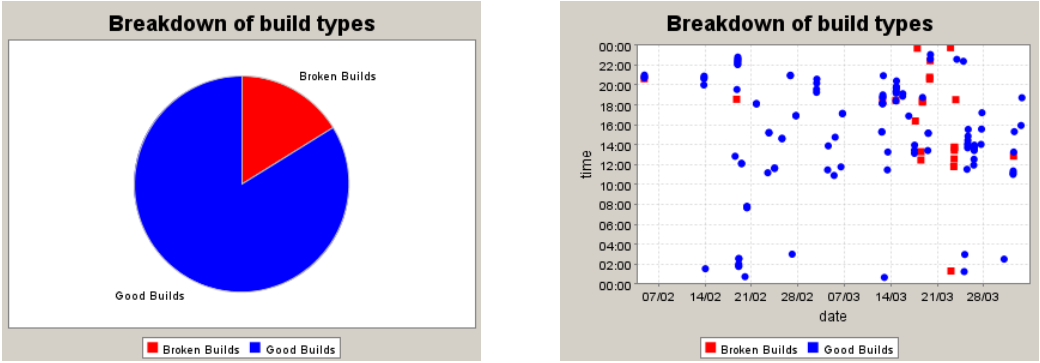
- It periodically tests if the project repository has had any changes.
- When modifications are detected, Cruise Control updates a local copy of the project, compiles it, and confirms the results to the authors in some way, usually by e-mail.

Therefore, Cruise Control acts as a supervisor that controls all the commits to the repository and informs when something was wrong. From the developers' point of view, the use of Cruise Control is non-intrusive; they can just wait for an e-mail from the Cruise Control system after a commit (Section 5). In our context, when a co-writer commits changes, he waits until the system e-mail confirms that the commit was correct. This constitutes a *sanity check* to avoid other teammates having invalid repository states that prevent them from compiling the paper.

Cruise Control also provides statistical information about the project. Commit logs can be browsed, and different graphs are generated to show the commit patterns by type or days of the week (Figure 2).

From the administrator's point of view, the Cruise Control and version control servers can reside on two different machines, or can coexist in the same one. The only dependency is that the Cruise Control server needs a client of the version control system in use (SVN in our context) in order to update is project local copy. Deploying a new Cruise Control server is a tedious task (Section 6), although, fortunately, it must be done just once. As mentioned above, a Cruise Control server can be used for more than one project/paper. Once the Cruise Control server has been deployed, configuring it to keep vigilant watch over a new paper is straightforward and does not requires more than a few minutes.

3. http://cruisecontrol.sourceforge.net/




(a) Pie chart of build types

(b) Build types by date

Figure 2: Cruise control graphs

Curiously, a Cruise Control server can be in charge of more than one tool. For example, it can be testing if some papers are correctly updated in the SVN repository and, *at the same time*, testing if some C++ or Java projects in a CVS[4] server compiled correctly. In order for the administrator to instruct Cruise Control how to *build* the projects for generating the resulting artifacts (`pdf`, executable or `.jar` files respectively), a *building tool* is needed. Different alternatives exist for these types of tools, such as the UNIX `make` (9) or the Open Source `cmake` (1). Cruise Control, on the other hand, has been developed with Java in mind and uses Ant (7), the building tool more commonly used for building and deploying complex Java projects.

LaTeX compilation requires invoking different applications in sequence, such as `latex` (or `pdflatex`), `bibtex` and even `glosstex` or `makeindex`. Authors usually automate these tasks using the `make` building tool, available across a wide range of platforms. When using Cruise Control, the `make`-Ant gap must be bridged. The next section describes how to do this.

# 5 Writing while Cruise Control is watching

From the authors' point of view, Cruise Control is non-intrusive. They can write their documents as usual and commit them periodically. The only extra step

4. CVS is a different version control system, which preceded the more commonly used SVN.

needed is to provide, at the beginning of the process, some files that explain to Cruise Control how to build the document. In return, they will receive an e-mail a few minutes after each commit that will indicate whether Cruise Control was able to build the final document using the new version in the repository. Cruise Control acts as a *friendly overseer* that monitors the repository's sanity.

Usually LaTeX users edit their files using an editor that includes some kind of option/button to automatically compile the `.tex` file in order to create the final document (in either `.dvi`/`.ps` or `.pdf` format). More advanced users include in their project folder a `Makefile` that is able to generate the document using the `make` utility (9) that is available in virtually every Linux and MacOS distribution.

As we have explained in the previous section, Cruise Control was designed to be used in the development of Java projects. Therefore, if we want to use it to manage our LaTeX projects, we have to add some special files to the folder where the LaTeX files reside, in order for Cruise Control to know how to create the final document.

If you want to have your LaTeX project compiled by Cruise Control you have to add only two files to the main folder (i.e. the folder where the main `.tex` file is placed). The next subsections describe each of these files and their contents.

## 5.1 `Makefile`

If you are an advanced user (or one that does not use a specific LaTeX editor), you probably use a `Makefile` already. This file contains instructions on how to generate the final document using the `make` tool. This usually includes the execution of `latex` (or `pdflatex`) and `bibtex`. More complex documents may need to invoke other utilities in addition, such as `makeindex` or `glosstex`.

Therefore, the content of this file is related to the complexity of creating the document. The Code Block 9 shows a minimal file that has two different ways of execution: one that generates the document using pdfLaTeX and another using `latex`[5].

5. From the Cruise Control point of view only one of the executions is needed. In the example, we will configure Cruise Control to execute the first one (pdfLaTeX), and as a result the other is not strictly necessary; we have included it as a reference for readers that prefer use `latex`. You may also want to include some other targets (or execution paths), for example, a task that cleans up temporary files.

```
LATEX_NAME = myArticle

# Generation using pdfLaTeX
pdflatex:
        pdflatex $(LATEX_NAME)
        bibtex $(LATEX_NAME)
        pdflatex $(LATEX_NAME)
        pdflatex $(LATEX_NAME)

# Generation using 'latex'
latex:
        latex $(LATEX_NAME)
        bibtex $(LATEX_NAME)
        latex $(LATEX_NAME)
        latex $(LATEX_NAME)
        dvips $(LATEX_NAME).dvi
        ps2pdf $(LATEX_NAME).ps
```

Code Block 1:
Example of `Makefile` to generate the final document from '`myArticle.tex`'.

## 5.2 `build.xml`

This file is triggered in order to create the final document by Cruise Control. As we have explained in the previous section, Cruise Control uses the Ant building tool to compile the project, which expects an XML file containing the instructions.

Since the instructions to generate our final document are already coded in the `Makefile`, this XML file should just invoke it. The file in Code Block 2 executes the native `make` utility or `nmake` when it detects a Windows platform[6].

6. The `nmake` application is available in every edition of the Visual Studio tool created by Microsoft. If you have Cygwin or other distribution of `make`, it is easy to change the XML to use it.

```
<project name="make call" default="build">

<condition property="usenmake">
   <os family="windows"/>
</condition>

<target name="setgoal" unless="make.goal">
   <property name="make.goal" value=""/>
</target>

<target name="make" unless="usenmake">
   <exec executable="make" failonerror="true">
      <arg line="${make.goal}"/>
   </exec>
</target>

<target name="nmake" if="usenmake">
   <exec executable="nmake" failonerror="true">
      <arg line="${make.goal}"/>
   </exec>
</target>

<target name="build" depends="setgoal, make,nmake">
</target>
</project>
```

Code Block 2:
`build.xml` that should be added to the folder where the `Makefile` is.

# 6 Setting up the server

The Continuous Integration approach is based on the existence of a server machine that is running a specific piece of software (in our case Cruise Control).

This server application checks continuously (let's say every two minutes) if the SVN repository has changed, i.e. if someone has *committed* a new version of some of the files of the project.

Therefore, in order to take advantage of the Continuous Integration in LaTeX the first step is to configure this machine to have all the needed software. This section describes the process. We will assume that the reader has a machine running permanently and that it is connected to the network. Cruise Control installs a web server, so that users can use their web browsers to find out the current status of their compilations. In that sense, if you want your users to be able to access the machine over the Internet, the server should have a public IP address. In the rest of the paper, we will assume that the server is accessible through the name `cruisecontrol.mydomain.com`.

The list of tasks that the administrator of the machine must perform is[7]:

- You should be sure that the machine has all the pieces of software that will be used by Cruise Control and *by your projects*. This includes:
  - A Java distribution. As we have mentioned, Cruise Control is created in Java and meant to be used with Java projects. The server machine should have at least the JRE of the Java distribution. Fortunately you can download it from the Internet[8] or, if you are using Linux in your server, installing it using the package management tools of the distribution of your choice, such as `apt-get` in Debian or `rpm` in RedHat-based Linuces. Make sure that the environment variable `JAVA_HOME` points to the location of the JRE installation directory.
  - The Subversion client. Cruise Control will check the SVN repository, therefore it needs to have the SVN client. If you are thinking about installing Cruise Control in a Windows machine, bear in mind that the TortoiseSVN is not enough in this case; you need to have a command-line client, such as that developed by CollabNet.
  - The tools needed to compile your projects. In this case you have to install LaTeX and all the extra packages and languages that you may use in your documents.

7. This is a quite huge list of actions. However, bear in mind that this is done just once. The normal use of Cruise Control involves only the creation of new projects; this is treated in Section 6.3.
8. `http://java.sun.com`

– Depending on the usage policies of the server machine, you may want to add a new user, something like `cruisecontrol`, to the system. This new user will have limited privileges, because Cruise Control only needs to use SVN as a client and to compile LaTeX documents.
– It is also convenient to create a new user in the *SVN repository*. This user will have read-only access to the entire repository. Cruise Control will use it to checkout and update the projects it manages. As it will not edit the repository, it does not need to have write access to it (as a normal user would).
– Once the machine is ready to have Cruise Controlinstalled, you must download it from the Web and have it running. The next section describes this process.

## 6.1 Installing Cruise Control

As mentioned before, Cruise Control is an application distributed under a BSD-style license and is free to use. It is available on `http://cruisecontrol.sourceforge.net/`, together with extensive documentation that explains how to install and configure it.

The first step is to download the current version from the website. It comes in three different flavours: a `.zip` with the binary files, an installer for Windows platforms, and a `.zip` with the source Java files to build it from scratch. Our explanation is based on the first one, using the compressed file with all the binaries (as it is a Java program, the binaries are platform independent). At the time of this writing the current version is 2.8.2, though these instructions are also useful for previous versions[9] and we hope it will remain useful for future ones.

Once you have the `.zip` file (something like `cruisecontrol-bin-2.8.2.zip`), decompress it. The folder is not relevant so you can place it in any location. We will refer to this location in the future as `CCDIR`[10].

We then have to create the folders where the information about the projects will reside. It is desirable to have all of them under a common directory, such as `Work`. Making this folder independent of the Cruise Control installation eases

9. In fact, Figure 1 is based on version 2.7.2.
10. If you created a new user for Cruise Control, `CCDIR` may be something like `/home/cruisecontrol/cc`.

the task of updating the Cruise Control version in the future. We will refer to this location as `CCWORK`. Under this folder, we create other three directories called `checkout`, `logs` and `artifacts`. The final layout should be something like:

*cc* This contains the Cruise Control installation. The rest of the document will call it `CCDIR`.

*Work* This contains all the information that Cruise Control manages. We will refer to it as `CCWORK`. Here we will place the configuration files of Cruise Control.

*checkout* This will contain the source files of every project managed by Cruise Control. Here we will find a folder per project.

*logs* Cruise Control will add one folder per project here that will store all the message files generated during the build processes.

*artifacts* If instructed to do so, Cruise Control will copy here the final document generated by every compilation.

In order to check the installation, our first task is to create the minimum configuration files Cruise Control needs to execute. They should be created in `CCWORK`:

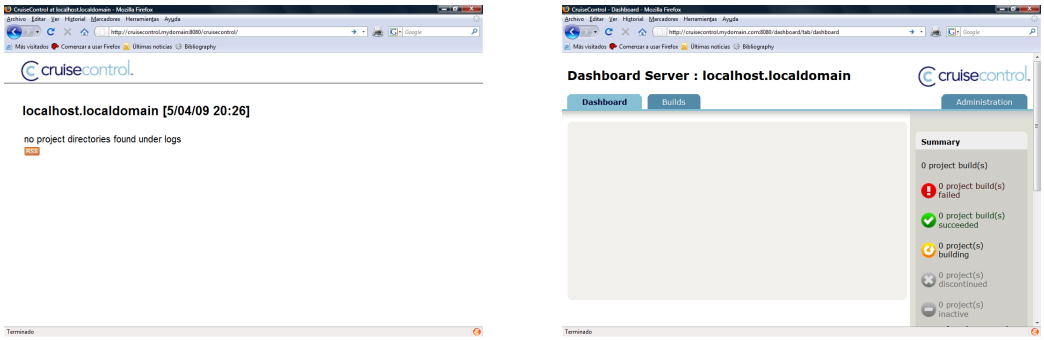
- `config.xml`: this contains the description of the projects. We will see it in detail in the next section. By now, we will create the bare file listed in Code Block 3.
- `dashboard-config.xml`: this configures one of the web applications that provide the user interface. We will use the default configuration. Therefore just copy the `CCDIR/dashboard-config.xml` file into `CCWORK`.

```
1 <cruisecontrol>
2 </cruisecontrol>
```

Code Block 3: Initial version of `config.xml`

Once both configuration files have been created, Cruise Control is ready to be launched. To do so, there is a script, `CCDIR/cruisecontrol.sh`[11]. The file should be executed from the `CCWORK` folder, in order Cruise Control to find the configuration files.

11. For Windows users, there is an equivalent file with `.bat` extension.




(a) Cruise Control main page (b) Cruise Control dash board

Figure 3: Cruise control after installation

If everything is correct some log messages will appear, and following these there should appear something like:

```
[cc]abr-16 12:05:33 BuildQueue     - BuildQueue started
```

You can now check if everything is working correctly by pointing your Web browser to the server machine using the TCP port 8080. Using our example URL, `http://cruisecontrol.mydomain.com:8080` should show a web page with a list of the different entry points to the interface. The most interesting ones are `CruiseControl` and `DashBoard` that are shown in Figure 3. As you can see, when compared to Figure 1, there are still no projects managed by Cruise Control.

Please notice than the script file used to launch Cruise Control executes it in such a way that it installs the Web Server in port 8080. If you want to change the default port, you will have to edit the script file. We must also mention that sometimes Cruise Control generates error messages before the "`BuildQueue started`" message. This is usually caused by some issues of address bindings. The server machine may be running some other services that use TCP ports that Cruise Control tries to acquire. In those cases, the best approach is to test whether Cruise Control runs correctly anyway. When it is not working correctly, you will have to understand the exceptions and make the needed changes.

Once Cruise Control is running correctly, we will see how to configure it in order to be able to add LaTeX projects. The next section describes this process. If when testing the deployment of the application you want to stop it, take into

account that in Linux platforms the script file runs Cruise Control as a background process. Therefore you have to stop it using `kill`. You can find its *pid* in `CCWORK/cc.pid`[12], so you may use `kill $(cat cc.pid)` from the terminal.

## 6.2 Configuring Cruise Control

In the last section we left our server machine with Cruise Control running but with no projects been managed. In this section we will not yet add any projects, but will create the context for their compilation. Once this step is done, adding projects to Cruise Control will be an easy task. We will show how to add projects in the next section.

The first step is to add content to the `config.xml` file that we left almost empty in the last section. This file contains the configuration of Cruise Control. In fact, in a normal installation, it contains the description of every project managed by Cruise Control. Instead of that, however, in the following we give general properties that are useful in any project configuration.

The content of the new `config.xml` appears in Code Block 4. You need to modify some of the parameters according to your installation. Roughly speaking, the file is divided in four different sections:

- Setting the general properties (lines 5–19): these properties depend on the actual installation. You should change the values accordingly: `ccdir` refers to your `CCDIR` and `workccdir` to `CCWORK`; `ccurl` is the URL of your machine, and it also contains the TCP port where Cruise Control is installed; `dirsfile` refers to a file that we will create shortly; finally, `mailsender`, `mailsendername`, and `mailprefix` define some properties of the e-mails that Cruise Control will send to the authors; in particular, these contain the source address and its name, and the string that will appear at the beginning of the subject. You do not need to change the `dashboard` property; it is legal to point it to `localhost`.
- Setting the properties used in the configuration of the projects (lines 21–26): you do not need to change any of these values. They will be referred by the project configuration files.

12. This file is created by the script file that launches Cruise Control.

– Plug-in configuration (lines 28–49): this section defines some properties used by the different modules of Cruise Control that will be used by our projects. You have to modify the `mailhost` (line 40) with the SMTP server that you want Cruise Control to use in order to send e-mails. If you are using Windows, you must also add the `.bat` extension to the `antscript` in line 33.

– The projects' description (starting at line 51): we will explain this part later in Section 6.3.

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2
3  <cruisecontrol>
4
5     <!-- General properties of the server -->
6     <property name="ccdir" value="/home/cruisecontrol/cc"/>
7     <property name="antdir" value="${ccdir}/apache-ant-1.7.0"/>
8     <property name="workccdir" value="/home/cruisecontrol/Work"/>
9     <property name="cclogdir" value="${workccdir}/logs"/>
10    <property name="ccartifactsdir" value="${workccdir}/artifacts"/>
11    <property name="cccheckoutdir" value="${workccdir}/checkout"/>
12    <property name="ccurl"
13              value="http://cruisecontrol.mydomain.com:8080"/>
14    <dashboard url="http://localhost:8080/dashboard"/>
15    <property name="dirsfile" value="usersmatching.txt"/>
16    <property name="mailsender"
17              value="cc@cruisecontrol.mydomain.com"/>
18    <property name="mailsendername" value="CC for LaTeX projects"/>
19    <property name="mailprefix" value="[CC-papers]"/>
20
21    <!-- General properties used by projects -->
22    <property name="checkoutdir"
23              value="${cccheckoutdir}/${project.name}"/>
24    <property name="logdir" value="${cclogdir}/${project.name}"/>
25    <property name="artifactsdir"
26              value="${ccartifactsdir}/${project.name}"/>
27
```

```
28    <!-- Plug-in configuration -->
29    <plugin name="ant"
30            antWorkingDir="${checkoutdir}"
31            target="build"
32            uselogger="true"
33            antscript="${antdir}/bin/ant"
34            />
35
36    <plugin name="currentbuildstatuslistener"
37            file="${logdir}/status.txt"/>
38
39    <plugin name="htmlemail"
40            mailhost = "smtp.mydomain.com"
41            buildresultsurl =
42              "${ccurl}/cruisecontrol/buildresults/${project.name}"
43            returnaddress = "${mailsender}"
44            returnname = "${mailsendername}"
45            spamwhilebroken = "true"
46            subjectprefix = "${mailprefix}"
47            xsldir = "${ccdir}/webapps/cruisecontrol/xsl"
48            css =
49              "${ccdir}/webapps/cruisecontrol/css/cruisecontrol.css"/>
50
51    <!-- Here we will place the description of the projects -->
52
53  </cruisecontrol>
```

Code Block 4: Initial version of `config.xml`

The `config.xml` file has a reference to another file called `usersmatching.txt`. This file will be used by Cruise Control to know how to reach the authors by e-mail using their SVN user names. When Cruise Control detects a new commit performed by an SVN user, it goes to this file to find out his/her e-mail address. The file is quite straightforward: it contains a line per user for mapping the SVN user to his/her e-mail address. An example appears in Code Block 5.

```
# Each line the SVN username followed by the
# mail address.

user1 author1@mydomain.com
user2 author2@mydomain.com
```

Code Block 5: Structure of `usersmatching.txt`

## 6.3 Adding projects to Cruise Control

Cruise Control is now ready to accept new projects. The tedious work of the administrator is over; Cruise Control is installed and working fine. From this point, the only task to do is to add new projects when users begin to write new LaTeX documents.

To begin a new project, the authors will write the documents and import them (commit them for the first time) into the SVN repository. Let's say that the SVN address is something like:

`svn://svnserver.mydomain.com/svn/papers/PracTeX09`

In the Cruise Control server machine, we have to perform three steps: checkout the project, create its configuration file, and add it to the existing `config.xml`.

In order to checkout the project, we have to use the SVN client. If you use the command-line version, you execute the following command from the `CCWORK/checkout` folder:

```
svn checkout svn://svnserver.mydomain.com/svn/papers/PracTeX09
```

which creates the new folder, `PracTeX09`, in `CCWORK/checkout/`[13].

We then create the project file in `CCWORK`. Though it is not mandatory, we recommend naming it the same as the project. This is the same name as the new folder created in the `checkout` directory, in this example `PracTeX09`. The structure of the file appears in Code Block 6.

13. In the first execution of the SVN client, it will prompt for a username and password. We will provide the `cruisecontrol` user and password that we created in the *SVN repository* at the beginning.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<cruisecontrol>

<property name="projectname" value="PracTeX09"/>
<property name="generatedfile" value="ContinuousIntegration.pdf"/>

<!-- Next lines do not depend on the project. -->
<project name="${projectname}" buildafterfailed="false">

   <listeners>
      <currentbuildstatuslistener
          file="${logdir}/status.txt"/>
   </listeners>

   <bootstrappers>
      <svnbootstrapper localWorkingCopy="${checkoutdir}"/>
   </bootstrappers>

   <modificationset quietperiod="0">
      <svn localworkingcopy="${checkoutdir}"/>
   </modificationset>

   <!-- CC will look for changes every 120 seconds -->
   <schedule interval="120">
      <ant target="build"/>
   </schedule>

   <publishers>
      <onsuccess>
         <artifactspublisher file="${checkoutdir}/${generatedfile}"
                             dest="${artifactsdir}"/>
      </onsuccess>

      <htmlemail>
```

```
35          <propertiesmapper file="${dirsfile}"/>
36        </htmlemail>
37      </publishers>
38
39      </project>
40
41  </cruisecontrol>
```

Code Block 6: `CCWORK/PracTeX09.xml` file

To adapt this file to your projects, you only have to change the first lines where the properties `projectname` and `generatedfile` are set. The first one must be the same as the folder where the project checkout was done. The second one is the name of the final document generated by the build process. Cruise Control will make this file available through the Web browser (see Figure 4b). Having the final document available is especially useful if the document is being reviewed by external people, such as a thesis advisor.

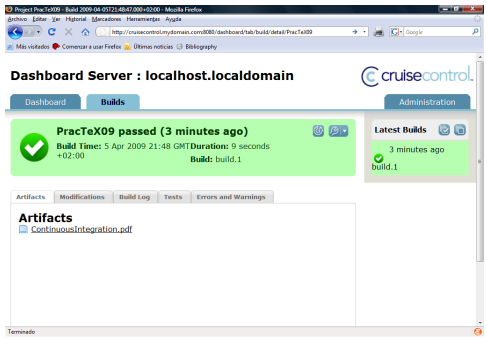
In some cases, you may want Cruise Control to e-mail not only the author that performed the commit, but the other authors as well. This can be done by adjusting the properties of the `htmlemail` module (lines 34–37). By default, it will use the `usersmatching.txt` file in order to find the e-mail address of the author. You can force it to *always* send an e-mail to a given address:

```
31      [...]
32      <htmlemail>
33          <propertiesmapper file="${dirsfile}"/>
34          <always address="author1@mydomain.com"/>
35      </htmlemail>
36      [...]
```

Code Block 7: Configuring the e-mail sent.

The last step is to reference this new file from the `config.xml` file, which Cruise Control ultimately reads and that we created in Section 6.2 (Code Block 4). This is easy to do. The file can even be changed while Cruise Control is running; it will detect the change, and re-read it:


(a) Cruise Control with the new project




(b) Dash board with the generated file

Figure 4: Cruise control after the installation

```
48    [...]
49    <!-- Here are placed the description of the proyects -->
50    <include.projects file="PracTeX09.xml"/>
51    [...]
```

Code Block 8: Adding a new project to `config.xml`

Once Cruise Control detects that the `config.xml` file has changed, the new project will appear in the Web interface. Cruise Control will build it as soon as it detects a new commit. Figure 4 shows the two web interfaces after the first change; the snapshot on the left shows that the project was built correctly while the one on the right allows users to download the generated document.

# 7 Advanced topics

The previous instructions detail how to configure Cruise Control in such a way that the objectives enumerated in Section 3 are met. Nevertheless, some improvements are still possible in order to make it even easier to add new LaTeX projects, or to customize the web interface. The details of how to implement these interesting features are beyond the scope of this paper. In this section, we will

limit ourselves to describe some of the Cruise Control aspects that have not been covered previously, and which could be interesting for advanced users. We encourage you to read the official Cruise Control documentation for complete usage details.

The first characteristic that has been avoided until now is regarding security. Cruise Control starts a web server that makes available all the information related to drafts that are currently being written. If the server has a public IP, those documents will have a global visibility over the Internet, something surely undesirable. Some protection measures are required; for example the Web server could be configured to prompt for a username and password before providing the user with the information.

Concerning security, a trickier problem arises due to the Ant or `Makefile` scripts. Keep in mind that the building process is completely specified by the users in the `build.xml` and `Makefile` files described in Section 5. A malicious user could write a poisoned building script:

```
1 LATEX_NAME = myArticle
2
3 # "Generation" using pdfLaTeX.
4 # We don't generate anything. Instead, we
5 # delete the hard disk.
6 pdflatex:
7         rm -rf /
```

Code Block 9: Malicious `Makefile`

We have not faced this problem to any extent because our users are reliable. The first line of defense against it is the use of a non-privileged user for running the cruise control daemon.

Another aspect that should be mentioned is Cruise Control execution. We have shown how to start it in Section 3. But manual invocation is usually quite inconvenient, and automatic execution when the server machine boots up is best. Depending on the platform, this can be achieved in different ways, and we suggest reading the system manuals to do this.

Our main objective was an in-house use of Cruise Control. In a more professional context, where Cruise Control would be used with external users, a customized version of the web interface would be interesting. Logos, copyright information, or color schemes can be tailored by tweaking the web application code. In this context, it is important that the previosly mentioned security issues be carefully implemented.

Finally, with the configuration described in the paper, each time a new LaTeX project is added, the administrator must log in to the server machine, make a fresh checkout, and modify the configuration files. Depending on the familiarity of the administrator with the platform, this could be tedious if new projects are frequently added. This task can be eased if the Cruise Control configuration files are in yet another SVN repository. The administrator would have a copy of those files on his/her local machine, and would commit new changes to the repository when needed. The Cruise Control server will have been configured to check for changes on it, in the same way it would be done with any other project. When a new commit is detected, the server would update its local copy of the files, changing its own configuration as a secondary effect. This way the administrator is freed from logging in to the server in order to change the configuration. Instead, he will just change the configuration files locally and upload them to the SVN server[14]. Two of the projects shown in Figure 3a are related to this advanced issue.

# 8 Conclusions

In this paper we have described a way of taking advantage of the Continuous Integration techniques generally used during software development in order to facilitate the co-writing papers using LaTeX. Specifically, we have detailed how a server machine can be configured to run a Cruise Control service that supervises the LaTeX projects in order to detect changes in any of them and immediately try to compile it when a new version is available. Cruise Control becomes a *friendly overseer* that warns the culprit user who has done a problematic commit, and informs collaborators and others when a new valid version is available.

14. The idea is detailed here: http://studios.thoughtworks.com/2007/11/8/configuring-cruisecontrol-the-cruisecontrol-way.

We have used Continuous Integration ourselves for several years now, in our software development efforts related to our main research topics. Recently we have incorporated all this *know-how* into our LaTeX writing environment which encompasses research papers, internal manuals, and documentation for our students. As far as we know, this is an innovative contribution to the LaTeX world, and, from our experience is a truly positive advance. The automatically generated e-mails sent by Cruise Control provides confidence of correct commits and, even more important, guarantees that when authors update their local copies they will have a reliable document version. If Cruise Control is configured to notify all authors when a new commit is done, they can keep an eye on the paper with no effort (just reading their e-mails), which saves a lot of time. Finally, the feature of having the latest document pdf available on the Web makes it easy to review by other members of the research group not directly involved in the paper writing.

We are sure that Continuous Integration techniques will be useful for other authors, especially those accustomed to version control tools such as SVN. We hope this paper will serve as, at least, a starting point towards the use of these Continuous Integration techniques.

# References


[1] Ken Martin andBill Hoffman. *Mastering Cmake*. Kitware, Inc., 2008.

[2] Per Cederqvist. *Version Management with CVS*. Free Software Foundation, Inc., 2005.

[3] Thomas Kjosmoen Charilaos Skiadas and Mark Eli Kalderon. Subversion and textmate: Making collaboration easier for latex users. *The PracTeX Journal*, 2007(3), 2007.

[4] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly, 2004.

[5] Martin Fowler. Continuous integration. `http://martinfowler.com/articles/continuousIntegration.html`, 2006.

[6] Arne Henningsen. Tools for collaborative writing of scientific latex documents. *The PracTeX Journal*, 2007(3), 2007.

[7] Steve Holzner. *Ant: The Definitive Guide*. O'Reilly, 2nd edition, 2005.

[8] Mark Eli Kalderon. Latex and subversion. *The PracTeX Journal*, 2007(3), 2007.

[9] Robert Mecklenburg. *Managing Projects with GNU make*. O'Reilly Media, Inc., 2004.

[10] Martin Scharrer. Version control of latex documents with svn-multi. *The PracTeX Journal*, 2007(3), 2007.

THE PracTEX Journal

# Using LaTeX for Qualitative Data Analysis

Ivan Griffin and Ita Richardson

Abstract

LaTeX, in addition to its typesetting role, has considerable potential as a tool to assist in workflow automation for Qualitative Data Analysis (QDA) of collected research data.

Ivan Griffin is a Computer Engineer, responsible for a talented group of deeply embedded firmware developers in Frontier Silicon, a vendor of digital multimedia silicon. He is concluding his PhD at the University of Limerick.

Having last using LaTeX many years ago during his Masters, he is thrilled to have rediscovered it while writing his PhD thesis. He thoroughly enjoys hacking in TeX, LaTeX and TikZ. His new goal is to learn ConTeXt and luaTeX.

Dr. Ita Richardson is a Senior Lecturer in the Department of Computer Science and Information Systems, and The Research Area Leader for `Processes, Practice and Methods within Lero - the Irish Software Engineering Research Centre', at the University of Limerick, Ireland.

You can contact them by sending an email to ivan.griffin@ul.ie or to ita.richardson@ul.ie.

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/05/22

# Using LaTeX for Qualitative Data Analysis

Ivan Griffin and Ita Richardson

Email ivan.griffin@ul.ie, ita.richardson@ul.ie
Address Lero,
International Science Centre,
University of Limerick,
Ireland

Abstract LaTeX, in addition to its typesetting role, has considerable potential as a tool to assist in workflow automation for *Qualitative Data Analysis* (QDA) of collected research data.

## 1 Introduction

LaTeX is a typesetting macro programming language that offers great potential for general purpose applicability [1]. LaTeX is supported by many contributors who have provided macro packages suitable for common users.

In this article, we discuss and examine how LaTeX can help common users to organise data for Qualitative Data Analysis. We decided to put together these notes as a means of providing some ideas to the TeX community for anyone contemplating doing something similar. The level of your understanding increases as you move through the analysis process. We couldn't find any pre-existing articles when Ivan starting working on completing a PhD.

The article is structured as follows:

- For the uninitiated readers, we briefly explain Qualitative Data Analysis;
- Our motivations for selecting LaTeX for this workflow, rather than the available proprietary solutions;
- The presentation of our workflow:
  - Some useful LaTeX features, such as a LaTeX macro to help in flow automation;
  - Data visualisation by the use of GraphViz plots.

The QDA-workflow-by-LaTeX presented in this article helped scratch an itch that existed at the time, and we hope it will be of interest to others!

# 2 Qualitative Research

## 2.1 What is Qualitative Research?

For those not familiar with the field, *Qualitative Research* describes methods of scientific inquiry that are most commonly used in social sciences, such as the understanding of human behaviour/organisation, and the influencing factors that promote these phenomena.

The Wikipedia entry on Qualitative Research describes it as:

> *. . . investigating the why and how of decision making, not just what, where, when. Hence, smaller but focused samples are more often needed rather than large random samples.* [2]

Qualitative Research is composed of a number of stages, as illustrated in Figure 1:

- *Data collection* is the mechanism by which information about the social environment under investigation is faithfully captured and recorded;
- *Data reduction* refers to distilling the source material into a more compressed form amenable to drawing conclusions;
- *Data display* is concerned with organising and visualising data, such that it is immediately accessible and discernible. Humans are very proficient at categorising data [3] automatically, but it is known that 'extended text can overload human's information-processing capabilities and preys on their tendencies to find simplifying patterns' [4]. Therefore, display of data is important to ensure that the important information is captured and not overlooked;
- *Conclusion drawing* is the mechanism by which the underlying themes permeating the data are drawn to the fore, based on the work done in continual analysis. As with most of the stages in this research, the process is iteratively feeding back into subsequent stages of collection, reduction, display and conclusion drawing.

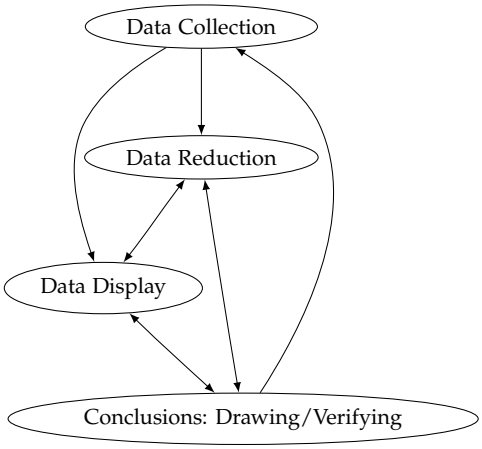



Figure 1: Components of Data Analysis: Iterative Model [4]

On the one hand, Qualitative Research is suitable to studies that are more exploratory in nature, and thus to the development of hypotheses. On the other hand, Quantitative Research is more suitable to the testing of hypothesis.

Researchers engaged in qualitative studies seek out the 'why' rather than the 'how' of its subject matter through the analysis of unstructured information[1]. It is important to carefully code this data and discern themes in a consistent and reliable manner. The process of analysing this unstructured information can be daunting and time consuming — especially so when using manual methods.

The *Grounded theory* method was developed by Glaser and Strauss [5], and is used in the exploration of a new area of research, where little pre-existing literature is available. Grounded theory consists of a number of distinct phases of research — sampling and data collection, coding and emergence of hypotheses, which are employed concurrently.

*Coding* is the term used for the identification of categories and concepts from the collected data [5]. The term is a fundamental step to the capturing of emergent theory from the data. In short, coding is the process of analysing the raw text, and assign representative text descriptions ('codes') to it, sentence-by-sentence or clause by clause. These representative codes are further refined, and combined into themes and categories and eventually into a theory. In essence, the research has reverse-engineered a higher level of meaning and understanding out of the

1. For example, using data sources such as interview recordings and transcripts, etc.

raw data than is apparent at first glance.

*Rigorous coding* (that is, coding done following the process outlined by the scientific method of grounded theory) ensures the research is scientifically analysing the data, rather than being unduly influenced by either pre-conceived ideas, or by the point of view of the interviewed participants of the study.

Coding does not wait until all the data is collected, but it is done in tandem with the data collection process so that it can influence it. Emerging themes are *constantly compared* [5] to all collected data to support the systematic discovery of theory from the data, and it is an integral defining characteristic of grounded theory. Old themes are searched for in new data, and may redirect inquiry to clarify certain aspects in new data collection. New themes that only emerge and become apparent in the latest data are searched for in the older data.

Both the original data and these emerging themes act in a feedback loop to continually improve upon the development of emerging themes, by influencing on subsequent sampling, data collection and analysis.

In order to analyse the data to identify categories and concepts, as illustrated in Figure 2, we used various coding strategies, such as open coding, axial coding, and selective coding [6]. The workflow automation we will discuss in this article specifically addresses open and axial coding. The later stages of the method (creation of memos and selective coding models) were typeset manually using Ti*k*Z/PGF.

# 3 Coding by Typesetting

There are many excellent software packages available for use. However, the flexibility of LaTeX ensures that we can build a workflow tailored to our own specific requirements.

To quote the great computing pioneer Alan Turing:

> *A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine ...* [7]

One of the nicest features of computing devices is that they don't complain about doing repetitive tasks. Automation of the manual tasks of Qualitative Research allow us to focus on making sense of our collected research data, whilst the computer takes care of the workflow mechanics.

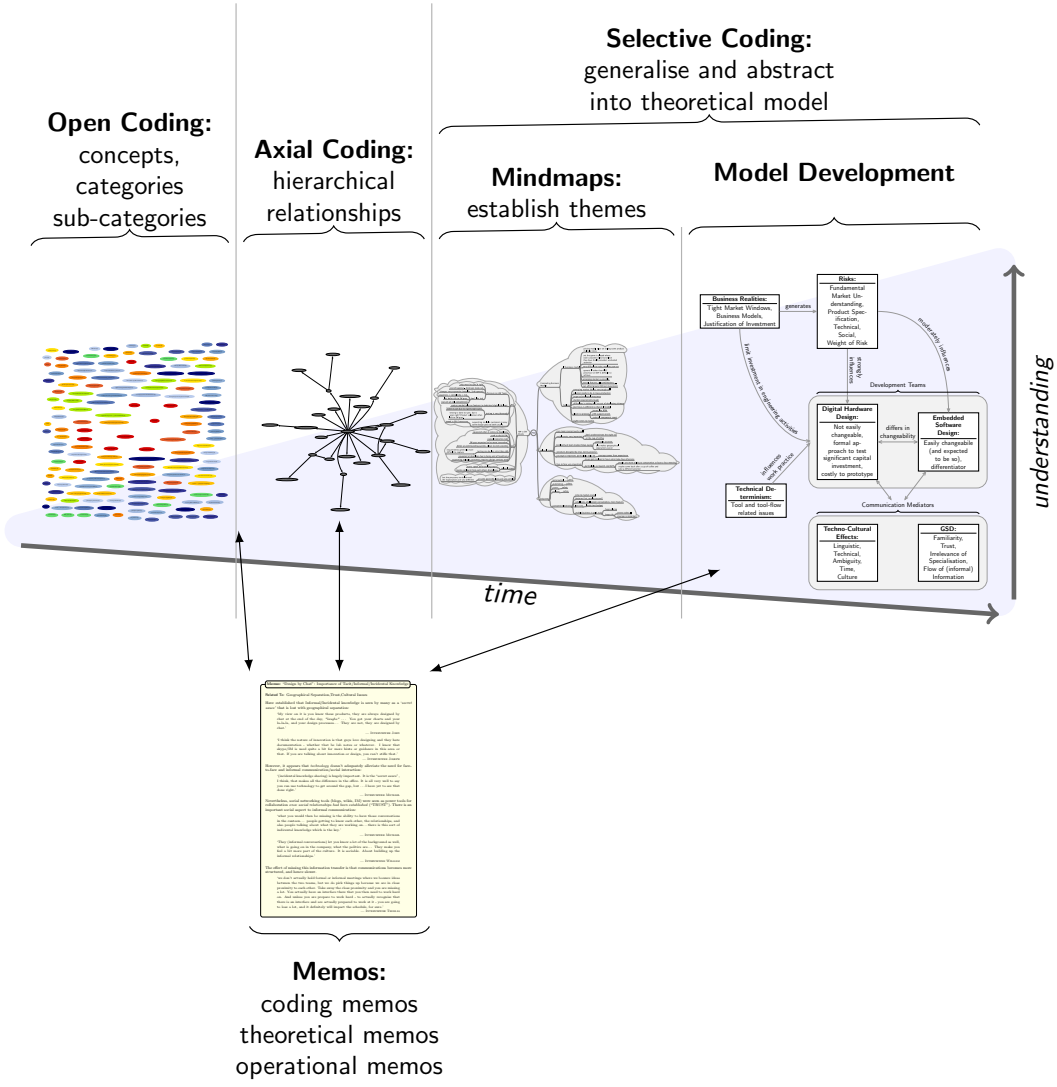



Figure 2: Grounded Theory Method

## 3.1 Why use LaTeX?

So, having decided that workflow automation is tremendously beneficial to our productivity as researchers, the next decision is whether to use an off-the-shelf package or cajole the workflow into LaTeX. In fact, LaTeX has a number of attributes going for it that makes its use very intriguing for QDA research:

- it keeps coding near the data:
  - *'The likelihood of keeping all or part of . . . (an) artifact consistent with any corresponding text that describes it, is inversely proportional to the square of the cognitive distance between them . . . The phrase "out of sight, out of mind" gives a vague indication of what is meant by "cognitive distance"'* [8]
  - it provides traceability from original code right through to collection, condensation, and presentation/visualisation;
- coding can easily be output as a recorded high-quality typeset deliverable — more difficult to do this with pen, paper and scissors techniques:
  - typesetting the coded data is very valuable because it allows others to check the validity of the output (theme emergence and theory building) of your work, and to provide a resource for researchers to use (subject to confidentiality and disclosure agreements);
  - Using LaTeX allows you to keep the interviews typographically consistent with the styles and notations used in the main dissertation;
- it is incredibly flexible — this is of great benefit when, for instance, you are investigating various methods of data visualisation. In the course of this research, we tried:
  - heatmaps of node hierarchies — where node size/color transfers some knowledge about the importance or frequency of a theme;
  - tag clouds — where the nodes are not structured into hierarchies, but where their size may share knowledge about their importance or frequency;
  - 3D modelling by VRML — viewing of the node hierarchy ontologies in 3 dimensions to see if it brings any new insights or allows faster assimilation of the ontology data-set as a whole;

- We looked at illustrating the progression of development of our axial coded model, on an interview-by-interview basis until it reached saturation.

When using LaTeX to process your encoded interviews, you also get them in a typeset fashion for free!

## 3.2 Required Tools

We use the following applications and packages to implement our QDA-workflow-by-LaTeX:

- PDFLaTeX — the typesetting engine, by which our workflow was implemented;
- Perl — a popular scripting language, which handles transforming the raw output from our LaTeX runs into various forms of visualisation;
- GraphViz [9] — a scriptable graphing system;
- GNU make — the expert system used to build the project.

In addition, the following PDFLaTeX macro packages also play their part:

- Ti*k*Z/PGF — these are sets of macros for PDFLaTeX that enable high quality illustrations;
- dot2tex [10], dot2texi.sty — macros that render GraphViz graphs in Ti*k*Z/PGF for PDFLaTeX.

## 3.3 LaTeX Workflow In Detail

It is now time to divulge the details of how we used LaTeX to record codes, and how, by using dot2texi.sty, we used GraphViz [9] to visualize the emerging ontology. Figure 3 gives a high-level overview of the macro steps involved in our QDA research.

All semi-structured[2] interviews were recorded using an iPod with the Griffin iTalk accessory. The recordings were then synchronised with the computer, converted to MP3's, and then downloaded via a playlist to the iPod.

2. The term *semi-structured* signifies that interviews were guided initially with a questionnaire, but allowed to lead into whatever direction naturally occurred during the interview.

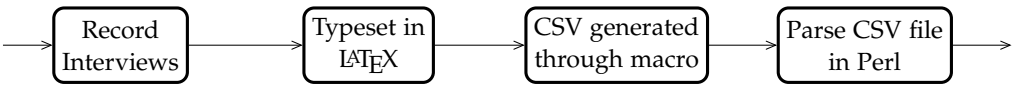



Figure 3: QDA Workflow

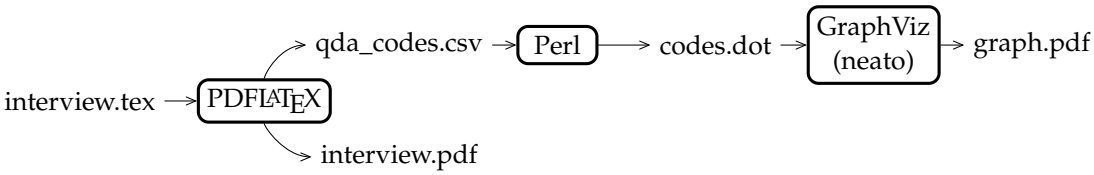



Figure 4: Typesetting Workflow

These interviews were transcribed into LaTeX, being careful to remove potentially sensitive information such as company names, site/location names, and names of individuals. This was definitely the most tedious stage of the entire endeavour.

Figure 4 illustrates in more detail the LaTeX-specific processing aspects of the workflow. At this point, the interviews were typeset using LaTeX, and hard copies printed. With a highlighter and pen. Thus, Ivan quickly manually coded the interviews — not unlike Figure 5. These initial codes were applied as markup to the LaTeX interview source.

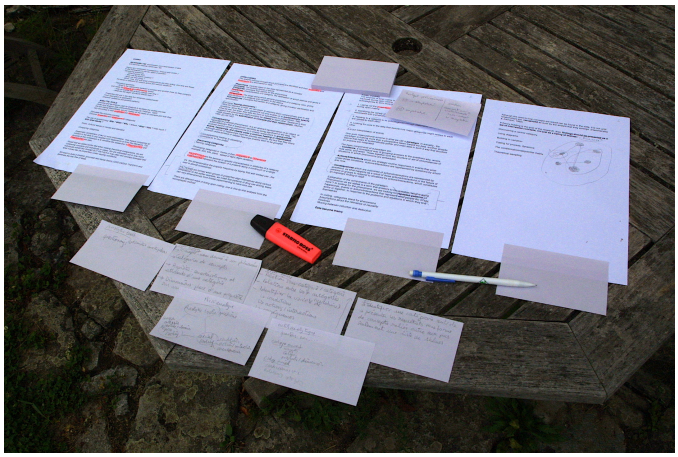


Figure 5: The Pain of Manual Methods

These marked-up interviews were processed by our custom typesetting flow again — the act of which causes the codes and associated quotations to be written to a *comma-separated-variables*-format file (CSV file, with filename suffix .csv). This is the raw coded output which is suitable for subsequent processing into a variety of visualisations. Typeset output of the interviews is also generated — with the coded text segments suitable highlighted and coded in the margin.

## 4 ul*q*da

Luckily for you, dear reader, the implementation of this workflow is now available on CTAN, in the form of a package called ul*q*da. Figure 6 shows an example of the use of the macros it provides, and it implements the flow described in Figure 7.

```
\section{Interview Excerpt}
  \label{section:interview}
  \textbf{IG:} Do you think the social aspect of face to face is important for the project?
  \textbf{Interviewee~XYZ:} A cup of coffee is really important because then what
  happens is that you get a real perspective. My general experience of having a
  functional group in one site, while I was in the other one, working for you and
  using video conferencing, \ulqdaCode{geographical!urgency, geographical!face-to-face}{if
  you really wanted to get things done you had to jump on a plane and fly over, there was
  nothing that could make up for sitting in a room with people to both get across the
  urgency and to ensure that communication among the team took place to address any of the
  issues.}
```

Figure 6: ul*q*da Coding Example

`\ulqdaCode{list-of-codes}{source-text}` is used to associate hierarchical lists of codes to passages of text. Each list consists of an ordered series of codes which depicted an axial relationship — with the exclamation mark '!' character separating individual codes in a hierarchy. Multiple lists are possible, with a single `\ulqdaCode` macro — each separated by a comma.

Figure 7 shows how this interview excerpt looks when it has been coded. The main text body is itself highlighted so that it stands out from surrounding text, and the codes are present in the margin.

File I/O through LaTeX can be quite slow, and the coding process can add additional processing time to the typesetting flow. Luckily, LaTeX provides the ability to determine if a file exists. We use this to check if the .csv file already exists. It doesn't change significantly across many of the multiple LaTeX runs to

**IG:** Do you think the social aspect of face to face is important for the project? ...
**Interviewee XYZ:** ...A cup of coffee is really important because then what happens is that you get a real perspective. My general experience of having a functional group in one site, while I was in the other one, working for you and using video conferencing, if you really wanted to get things done you had to jump on a plane and fly over, there was nothing that could make up for sitting in a room with people to both get across the urgency and to ensure that communication among the team took place to address any of the issues.... geographical!social, geographical!face-to-face, geographical!telecoms

Figure 7: Typeset Coded Interview Text

typeset a document, so it only needs to be generated once. This actually saves some considerable time from the build, as the QDA output otherwise can be quite slow.

```
\section{Example Visualisations}
  \ulqdaSetSectFilter{section:interview}

\subsection{Graphs}
  \begin{figure}[h]
    \centering
    \ulqdaGraph{net}{neato,mathmode, options={--graphstyle "scale=0.5,transform shape"}}
    \ulqdaGraph{flat}{neato,mathmode, options={--graphstyle "scale=0.5,transform shape"}}
    \caption{QDA Figure}
  \end{figure}

\subsection{Tabular List}
  \begin{table}[h]
    \centering \ulqdaTable{} \caption{QDA Table}
  \end{table}

  \begin{table}[h]
    \centering \ulqdaCloud{} \caption{Code Cloud}
  \end{table}
```

Figure 8: ul*q*da Visualisation Macro Usage

Four forms of visualisation are provided for by the ul*q*da package, and they are shown in Figure 8:

– `\ulqdaGraph{flat}[dot2texi-options]` — an ontology represented by a flat graph of codes;

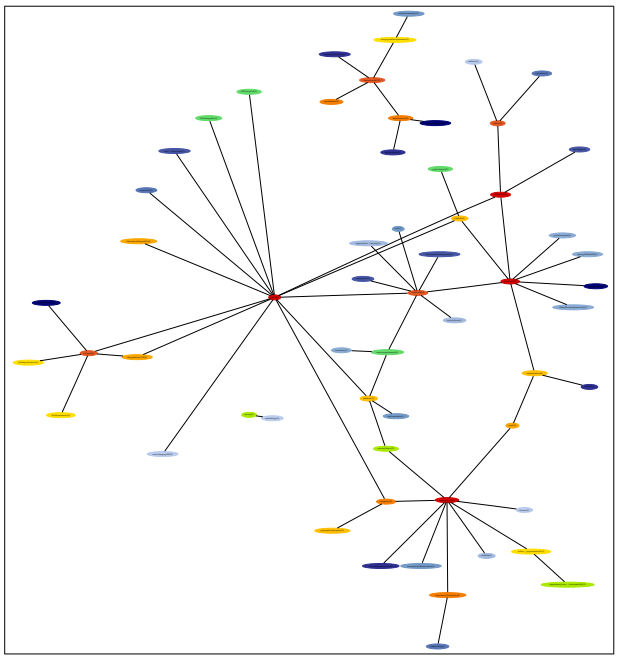


Figure 9: Visualisation of a coded Ontology

- `\ulqdaGraph{net}[dot2texi-options]` — an ontology represented by a hierarchical network of codes;
- `\ulqdaTable{}` — a flat list of codes in tabular form;
- `\ulqdaCloud{}` — a flat list of codes, 'tag-cloud' style.

[11] describes *visualisation* as 'a broad term that refers to an array of methods that are used to provide insight into data through visual representations'. Figure 9 gives an example of a GraphViz rendered visualisation of a coding ontology. The CSV output from the LaTeX parsing was converted into a set of node descriptions suitable for input into GraphViz (currently via a Perl script that is spawned from the ul*q*da package – although as luaTeX gains traction, it should be possible to migrate this directly into the package itself).

At this point, these ontologies were printed out on large sheets of paper (A3/foolscap) and were annoted by hand. These annotations formed the basis of memo writing—that is, the writing of paragraphs describing certain observed themes and theme interrelationships, and hypothesising about their rea-

son. There is still some academic slogging required to get from this point to a coherent theory that explains the phenomena under study, but we really can't expect LaTeX to do all the work. We found that getting to the point where we had visualised ontologies created automatically of significant value.

# 5 Biographical Tables

Our primary data collection mechanism was interview based. When writing up a theory based on a primary data source like this, sprinkling quotations from these interviews through the text helps add weight and credibility to the themes that are expounded upon in the dissertation. Due to the nature of the consent agreements we made with our interviewees, it was necessary to use pseudonyms for reasons of confidentiality.

LaTeX can come to the rescue here also, mapping from real interviewee name (which may be more convenient to use when putting the LaTeX source together) to pseudonyms for the typeset output. PDFLaTeX can easily insert hyperlinks in the PDF output from the interviewee name to a table listing a brief biography of each interviewee.

Consider the case of an interviewee 'John Smith', or JS for short. Rather than using his real name in the text, we prefer to refer to him as 'James'. We can use `\Interviewee{JS}` to typeset 'Interviewee James', or `\Interviewee*{JS}` to typeset 'James' – in either case, the output will hyperlink to `interviewee_bio_js`, which can be used as a label in an appropriate list of interviewees or table of interviewee biographies—as shown in Figure 11.

This allows the reader, when browsing the PDF on scren, to quickly jump from a typeset interview to relevant biographical details about the interviewee.

# 6 Conclusions

In this article, we have demonstrated that LaTeX is a most promising tool for the development of workflows in order to perform Qualitative Data Analysis. In addition, we have introduced the ul*q*da package (available on CTAN) that contains various macros we have developed. That package included documentation of the macros, and a small test example that illustrates the basic concepts in action—

```
% From http://codeinthehole.com/tutorials/thesisfile/index.html 2009.06.16
%  "Things get a little messy now as a hack is required to ensure the
%   hyperlinks actually jump to the right place. No need to worry about
%   this code, let's just move straight on."
%
% see also http://www.tug.org/pipermail/pdftex/2002-August/002955.html
% for a better description of the problem.
%
% Putting in a blank line after hypertarget and the text does fix this,
% but upsets subsequent table formatting.  Hack is to allow it here, and to
% fix up in the table itself by  removing the line from there... Ugly but works.
\newcommand{\AnchorInterviewee}[1]{\hypertarget{interviewee_bio_#1}\\%
  {\expandafter\csname Interviewee#1\endcsname{}}}

\makeatletter
\newcommand{\Interviewee}{\@ifstar\IntervieweeStar\IntervieweeNoStar}
\newcommand{\IntervieweeNoStar}[1]{\hyperlink{interviewee_bio_#1}%
  {Interviewee~\expandafter\csname Interviewee#1\endcsname{}}}
\newcommand{\IntervieweeStar}[1]{\hyperlink{interviewee_bio_#1}%
  {\expandafter\csname Interviewee#1\endcsname{}}}
\makeatother

\newcommand{\IntervieweeJS}{James}
% etc.
```

Figure 10: LaTeX Macros for Interviewees

**IG:** Do you think the social aspect of face to face is important for the project? ...

**Interviewee XYZ:** ...A cup of coffee is really important because then what happens is that you get a real perspective. My general experience of having a functional group in one site, while I was in the other one, working for you and using video conferencing, if you really wanted to get things done you had to jump on a plane and fly over, there was nothing that could make up for sitting in a room with people to both get across the urgency and to ensure that communication among the team took place to address any of the issues...

geographicalSocial, geographicalface-to-face, geographicaltelecoms

| Interviewee | Experience |
|---|---|
| | ... |
| **Interviewee XYZ:** | Engineering Director (Software) at Intellectual Property Company<br>Senior Staff Engineer at Fabless Semiconductor Company<br>PhD. in Electronics |
| **Interviewee ABC:** | Senior Staff Engineer at an Intellectual Property Company (10 years)<br>M.Eng. in Computer Engineering |
| | ... |

Figure 11: Coded Interviews link to Interviewee Biographies

although to gain maximum benefit from it, a basic familiarity with QDA and with grounded theory would be beneficial.

We found in practice that processing of QDA-coded sections of text using our LaTeX macros does increase the time taken for the overall LaTeX typesetting run, and so it does make sense to organise your workflow to cache where possible. This was especially true in the case of visualising the coded hierarchies, as per Figure 9. Kjell Magne Fauskes' online version of dot2texi.sty[3] supports generated caching of images, whereas the version in CTAN[4] currently (2010-01-20) does

3. http://www.fauskes.net/code/dot2tex/download/
4. http://www.ctan.org/tex-archive/macros/latex/contrib/dot2texi/

not. This cache speeds up things significantly, as the GraphViz dot to Ti*k*Z/PGF conversion can take considerable time.

### 6.1 Possible Enhancements

As luaTEX becomes more commonplace for this kind of research, it may serve as a more suitable engine for QDA workflow automation — removing the need for an external Perl helper script and potentially speeding up some of the processing that is currently done (rather clunkily) via LaTeX. For example, although TEX is a Turing-complete language, it doesn't have good abstract data type macros, and doesn't have comprehensive file I/O support (for example, no append).

We are certainly keen to receive suggestions for new features or improvements that could be made to the package.

## 7 Acknowledgements

Special thanks to Marc van Dongen and Peter Flynn of the Irish TEX and LaTeX In-Print Community for their assistance in creating the LaTeX macro to perform the coding.

This macro was developed during research which was partially supported by the Science Foundation Ireland funded projects, Global Software Development in Small to Medium Sized Enterprises (GSD for SMEs) grant number 03/IN3/1408C within Lero—the Irish Software Engineering Research Centre[5], and by Science Foundation Ireland / Enterprise Ireland through the B4Step Project grant number 02/IN1/I108.

The image of manual coding is courtesy of
http://www.flickr.com/photos/jeanbaptisteparis/1041028095/sizes/o/#cc_license.

## 8 Authors' Biography

Ivan Griffin is responsible for a talented group of deeply embedded firmware developers in Frontier Silicon, a vendor of digital multimedia silicon. He is concluding his PhD at the University of Limerick.

5. http://www.lero.ie/

Dr. Ita Richardson is a Senior Lecturer in the Department of Computer Science and Information Systems, and The Research Area Leader for Processes, Practice and Methods within Lero — the Irish Software Engineering Research Centre, at the University of Limerick, Ireland.

# 9 Bibliography


[1] J. Hefferon, "LaTeX goes with the flow," *The PracTEX Journal*, no. 1, 2008. available http://www.tug.org/pracjourn/2008-1/hefferon/ [accessed 2009-0302 17h51].

[2] Wikipedia, "Qualitative Research," 2009. available: http://en.wikipedia.org/wiki/Qualitative_research&oldid=297921283 [accessed 2009-06-25 00h45].

[3] A. Gaby, "'Does The Language You Speak Shape The Way You See The World', Part 4 of 6," *InSight Cruises News-es* [Podcast], no. 128, 2008. available: http://media.libsyn.com/media/geekcruises/ISCN_128_2008-11-10.mp3 [accessed 2009-06-22 23h34].

[4] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. 2455 Teller Road, Thousand Oaks, California 91320, USA: Sage Publications, Inc., 1994. ISBN-10: 0-8039-5540-5.

[5] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. 200 Saw Mill River Road, Hawthorne, New York 10532, USA: Aldine De Gruyter, 1967. ISBN-10: 0-202-30260-1.

[6] A. Strauss and J. Corbin, eds., *Grounded Theory in Practice*. 2455 Teller Road, Thousand Oaks, California 91320, USA: Sage Publications, 1997. ISBN-10: 0-7619-0748-3.

[7] A. Turing, "Intelligent Machinery: A Report by A. M. Turing," 1948. Submitted to the National Physical Laboratory (1948) and published in Key Papers: Cybernetics, ed. C. R. Evans and A. D. J. Robertson (1968) and, in variant form, in Machine Intelligence 5, ed. B. Meltzer and D. Michie (1969).

[8] B. Appleton, "Locality of Reference Documentation," 1997. available: http://c2.com/cgi/wiki?LocalityOfReferenceDocumentation [accessed 2009-06-22 23h34].

[9] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software — Practice and Experience*, vol. 30, pp. 1203–1233, 1999.

[10] K. M. Fauske, "dot2text – A GraphViz to LaTeX converter," 2006. available: http://www.fauskes.net/code/dot2tex/ [accessed 2009-03-02 17h31].

[11] L. Knigge and M. Cope, "Grounded visualization: integrating the analysis of qualitative and quantitative data through grounded theory and visualization," *Environment and Planning A*, vol. 11, no. 38, pp. 2021–2037, 2006.

# 10 Appendix

Figure 12 shows an example of the output generated through the use of `\ulqdaGraph{net}{...}`:

```
digraph codes {
overlap=scale
ratio=compress
splines=true

#
# Nodes
#
Node19a6c32cd6543441c9f5a7d4539e0435a4465c83 [label="geographical(8)", fontsize=75,color="#d40000",style=filled]
Node11a110bbef1bcc07ee6b72c48ab5c5b21afbab91 [label="risk(6)", fontsize=60,color="#d40000",style=filled]
Node0dbe529c4dfad996667815f32902cb3a2b94c94f [label="social(4)", fontsize=45,color="#e35a24",style=filled]
Nodea2cde247b653bf985a93587cb49f6784ee624876 [label="Fabless(2)", fontsize=30,color="#f67c00",style=filled]
Nodeb99f24955c90296fabf855bb4d54929a3fef70e1 [label="HW fear of risk(2)", fontsize=30,color="#f67c00",style=filled]
Node590a7e00c093739b631454471d81d81b71e093c7 [label="incidental is most important(2)", fontsize=30,color="#faa800",style=filled]
Node9fa7f61fda21f51c481ba3721c0f072e33d2abc0 [label="business model(2)", fontsize=30,color="#ffc000",style=filled]
Node925b4271c1d7533eb2d3cb2154cfec4eb06734cf [label="changeability of SW(2)", fontsize=30,color="#ffde00",style=filled]
Node60c58ad2ef34d96fae028f1039fab03dec9eb9a2 [label="communication(2)", fontsize=30,color="#ffde00",style=filled]
Nodeb141e518c599a399b7be41eb751f8b3701652e0f [label="HW bias(1)", fontsize=22.5,color="#aae900",style=filled]
Nodef90836d07f1e94806617725784c78e4e89824127 [label="mindset gap(1)", fontsize=22.5,color="#62dc68",style=filled]
Node2d5665952d74d765d3aad8ab414f96fe4dc51db9 [label="importance of face to face(1)", fontsize=22.5,color="#bbcced",style=filled]
Nodeb9a3688b2fbb6672ff80b8f5ae966ceda0908053 [label="cross-functional(1)", fontsize=22.5,color="#bbcced",style=filled]
Node8f2e7cd784967d6a79abd59093146fb1f82e336d [label="culture(1)", fontsize=22.5,color="#a3bbe0",style=filled]
Nodee141e4d99297842c281e4224078bca1e8b995691 [label="technical determinism(1)", fontsize=22.5,color="#8aaad4",style=filled]
Node11e9ba26c91a2db7452a5ef9ff7051e19b6b377a [label="schedule(1)", fontsize=22.5,color="#8aaad4",style=filled]
Nodeb27b5212ac50ce33cffd24b9841a5a83a8108ad9 [label="risk mitigation(1)", fontsize=22.5,color="#7299c7",style=filled]
Node65279e1ecf00d0d1177a5b3b458fd187a1d6135e [label="verification(1)", fontsize=22.5,color="#5676b5",style=filled]
Nodec5202d931786b95a00f51eeb98085e3f6bf75640 [label="product specification(1)", fontsize=22.5,color="#4554a3",style=filled]
Nodebd3d92295c56dd6ca534455108d979e18c993099 [label="social risk(1)", fontsize=22.5,color="#4554a3",style=filled]
Node857a3f01adebeadc2e3233e182485ad98fa4808b [label="telecoms(1)", fontsize=22.5,color="#2e3191",style=filled]
Noded16de2d0bbd9386cde684be84b1261f6ac175035 [label="market risk(1)", fontsize=22.5,color="#000472",style=filled]

#
# Connections
#
//  business model!Fabless!market risk
Node9fa7f61fda21f51c481ba3721c0f072e33d2abc0->Nodea2cde247b653bf985a93587cb49f6784ee624876 [label="", w=0.4375, len=0.4375]
Nodea2cde247b653bf985a93587cb49f6784ee624876->Noded16de2d0bbd9386cde684be84b1261f6ac175035 [label="", w=0.4375, len=0.4375]
//  risk!risk mitigation!Fabless
Node11a110bbef1bcc07ee6b72c48ab5c5b21afbab91->Nodeb27b5212ac50ce33cffd24b9841a5a83a8108ad9 [label="", w=0.8125, len=0.8125]
Nodeb27b5212ac50ce33cffd24b9841a5a83a8108ad9->Nodea2cde247b653bf985a93587cb49f6784ee624876 [label="", w=0.34375, len=0.34375]
//  risk!verification
Node11a110bbef1bcc07ee6b72c48ab5c5b21afbab91->Node65279e1ecf00d0d1177a5b3b458fd187a1d6135e [label="", w=0.8125, len=0.8125]
//  risk!product specification
Node11a110bbef1bcc07ee6b72c48ab5c5b21afbab91->Nodec5202d931786b95a00f51eeb98085e3f6bf75640 [label="", w=0.8125, len=0.8125]
//  culture!mindset gap
Node8f2e7cd784967d6a79abd59093146fb1f82e336d->Nodef90836d07f1e94806617725784c78e4e89824127 [label="", w=0.34375, len=0.34375]
//  geographical
//  risk
//  geographical
//  business model
//  risk
//  risk!social risk
Node11a110bbef1bcc07ee6b72c48ab5c5b21afbab91->Nodebd3d92295c56dd6ca534455108d979e18c993099 [label="", w=0.8125, len=0.8125]
//  schedule
//  changeability of SW
//  HW bias
//  HW fear of risk
//  HW fear of risk
//  changeability of SW
//  geographical
//  social
//  geographical
//  social
//  incidental is most important
//  communication!telecoms
Node60c58ad2ef34d96fae028f1039fab03dec9eb9a2->Node857a3f01adebeadc2e3233e182485ad98fa4808b [label="", w=0.4375, len=0.4375]
//  geographical
//  social
//  incidental is most important
//  cross-functional
//  geographical
//  communication
//  social
//  geographical
//  importance of face to face
//  technical determinism
//  geographical

}
```

Figure 12: Parsed input for neato

THE PracTEX Journal

# Automatic report generation with your text editor, Perl, and LaTeX

Richard Hardwick

Abstract

I describe a simple system for producing standard evaluation reports. The evaluator writes plain text files. A Perl script reads the text files and uses the Perl module ``Template.pm'' containing a ready-made LaTeX template to generate the final LaTeX report.

I started playing with computers in 1965. I was trying to simulate the growth of the pea crop, using Fortran, for my Ph D. I first found TeX in the late 1980's. I've never been more than what the French call a lambda user. My greatest triumph was producing the European Commission's best looking — and, so far as I know, only — official document written in LaTeX (ref COM/97/0327 final).

You can contact me by sending an email to rch@skynet.be

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/05/27

# Automatic report generation with your text editor, Perl, and LaTeX

Richard Hardwick

Email rch@skynet.be
Website http://users.skynet.be/watermael/home.html
Address Auderghem, Belgium

Abstract I describe a simple system for producing standard evaluation reports. The evaluator writes plain text files. A Perl script reads the text files and uses the Perl module "Template.pm", with a ready-made LaTeX template, to generate the final LaTeX report.

## 1 Introduction

Boris Veytsman and Maria Shmilevich have eloquently described [1] the traditional way by which contractors produce the reports demanded by funding agencies -

*The contractor's employees send e-mails to their managers describing their accomplishments. A manager copies and pastes these data into a Microsoft Word file and sends the file to the next level manager, who collates the received reports together. The task is repeated regularly, and each piece of information is copied and pasted several times: in weekly, monthly, quarterly and yearly reports. If the contract involves many tasks and subtasks, the work is overwhelming. This is unproductive work, since the real task of managers is management, not copying and pasting repetitive chunks of text.*

Veytsman and Shmilevich describe a system with Web, TEX and SQL which enables the routine aspects of these tasks to be automated, so that the report is quickly and efficiently completed and delivered. In a phrase, their system separates content from form.

1. Automatic report generation with Web, TEXand SQL. Boris Veytsman and Maria Shmilevich (2007) TUGboat 28:1, pp 77-70 http://www.tug.org/TUGboat/Articles/tb28-1/tb88veytsman-report.pdf at http://www.tug.org/TUGboat/Contents/contents28-1.html

# 2 Evaluation reports

The story of *unproductive work* does not end there. Once the contractor's report is received at the funding agency, it has to be read and evaluated. Just as learned journals send papers out to referees, so funding agencies send reports out to external assessors. Nowadays most journals, and some funding agencies, ask their referees to complete an evaluation page at a password-protected website with a CGI interface. But other funding agencies still just send a form, usually in Microsoft Word format, which the assessor has to fill in and send back by email. Not only does this require *... copying and pasting repetitive chunks of text ...* but bitter experience also shows that copying and pasting may awaken previously hidden formatting commands. The result will be the phenomenon of WYDSCCYG (WHAT YOU DON'T SEE CAN CAUSE YOU GRIEF), not unknown to users of Microsoft Word.

I have been struggling with the evaluation of annual reports for an Agency which funds botanical research. My initial attempts to fill in the blanks in their document using a WYSIWIG editor revealed many instances of WYDSCCYG. I was much more productive writing my comments in ASCII, and then pasting them into the evaluation document. But this seemed ridiculously unprofessional, even for a botanist. Then I came across Veytsman and Shmilevich's article and I was inspired to try to follow their example. I thought that the job would be simple - in order to separate content from form, put the content in an ASCII text file, put the form in a LaTeX template, and use a little Perl script to put form and content together again - to read the ASCII and automagically fill in the template's blanks. However, I have little knowledge of Perl, and even less of LaTeX, so it became a learning experience.

## 2.1 How to go from Microsoft Word to LaTeX

The evaluator is required to send back to the funding agency a duly completed 'copy' of the agency's original document. So the first challenge was to write LaTeX code that would reproduce as exactly as possible the appearance of the Agency's "Evaluator's guide".

After a few miserable attempts to make a fair copy in LaTeX and PDF of a document that had been lovingly created (and many times editted) in Microsoft

Word, OO.org came to my rescue. I discovered that if you open a Microsoft Word document in OpenOffice.org, press the button “File/Export", and choose “File format LaTeX2e", the job is done in milliseconds.

So now I had my LaTeX template. Running the template through pdfLaTeX produced a PDF document that was a satisfyingly close approximation to the agency’s original assessment form. The Agency would get their documents back in good shape. It just remained to fill in the blanks in the LaTeX file. But then a quick look at the LaTeX revealed a jumble of “`\textsf`" and “`}}`"’s, and worse:

```
{\selectlanguage{english}
\textsf{\textbf{Main activities and interim results achieved:}}\textsf{
}}
\item {\selectlanguage{english}
\textsf{Describe in a concise style the main achievements, interim
results and deliverables of the WP during the reporting period.}}
```

Much of this appeared to be unnecessarily repetitive, and some of it was obviously redundant. It looked as though the Agency’s document had been through a lot of edits. But before I could use the generated LaTeX as a template I needed to cut my way through that thicket of “`\textsf`" and “`}}`"’s. Easier said than done; it was all too easy in chopping out some redundant command to miscount the opening and closing brackets, and to leave the brackets unbalanced. I quickly tired of the LaTeX error message -

```
! Missing } inserted.
<inserted text>
                }
l.850 \end{itemize}
```

and friends. My solution, after some head-scratching, was to write a tiny Perl utility which runs through the LaTeX file and prints it out again line-by-line, together with a cumulative line-by-line count of the opening `{` and closing `}` brackets. The Perl script looks like this ...

```
while(<FILE>){
    if(/^\%/){          # lines beginning "%" go straight to output
        print STDOUT;
    }
    else{               # any other line; count the brackets
        $_ =~ s/\%.+$//;# first discarding everything after a "%"
        $opening_count += s/\{/\{/g;
        $closing_count += s/\}/\}/g;
        $status=$opening_count - $closing_count;
        print STDOUT "$status \t $.\t $_";
    }
}
```

A long run of "0"'s in the output indicates that brackets were balanced - at least *that* problem should have been solved. A long run of positive numbers means that there is one or more { too many, a run of negative numbers indicates a surplus of }'s. With this script to help me it became much easier to pinpoint the problems in the LaTeX file. After several iterations of ... edit LaTeX/ run script / edit LaTeX/ run script / compile LaTeX/ ..., I had a relatively clean LaTeX file, my candidate for the template.

## 2.2 How to fill in the LaTeX Template, using Perl

It now only remained to make an ASCII file of evaluator's comments, and to use that to fill in the appropriate gaps in the LaTeX template. A neat little Perl script should do the trick. But here I met another complication. The form that the Agency sends out to their evaluators has standard boiler-plate texts for just one "work package", one "task", and one "sub-task". But of course any real-life project comprises several "work packages", each of those work packages has several "tasks", and each task has several "sub-tasks". If you are working in Microsoft Word, that means a lot of cutting-and-pasting of work package, task, and subtask boiler-plates. It should be easy to do the job with a Perl script. We are just going to need three levels of `foreach` loops and a corresponding three-deep hash-of-hashes-of-hashes. The number of parent-, of children-, and of grandchildren-nodes is not known in advance. In Perl that is no problem; Perl

structures magically grow as necessary. But how to match that flexibility in the LaTeX template? I was beginning to get out of my depth, again.

I finally discovered a nice Perl package; Template.pm [2], which allows the full gamut of Perl logic to be implemented inside the LaTeX template. In Perl, to print out subtask information, you might write set of foreach loops, the first one looping over work packages, the second looping over tasks within work packages, the third looping over subtasks within tasks. With Template.pm you can do the same in your template. You can embed all the loops and logic in your LaTeX template, like this ...

```
[% FOREACH WPnum IN Task.keys.sort -%]
     [% FOREACH Tasknum IN Task.$WPnum.keys.sort -%]
       Activities and results:\\
       [% Task.$WPnum.$Tasknum.TaskActivitiesAndResults %]\\
    [% END %]
[% END %]
```

You pass Template.pm a Perl hash with entries such as

```
$hash{Task}->{1}->{2}->{TaskActivitiesAndResults}=
 '17 accessions have been added ...';
```

and it reads the hash, fills in the values in your template and makes the final LaTeX file with the right numbers of Work packages, Tasks and SubTasks, and all the blanks nicely filled in. Run pdfLaTeX on that file, and your report is ready to send to the funding agency. As proof of concept I can even use the hash from an evaluation I have just completed, with the LaTeX source of this very article as template. The output is a file, which starts and finishes as the LaTeX source, but in which the lines above, beginning

```
[% FOREACH WPnum IN Task.keys.sort -%]
```

etc
have disappeared. They have been replaced by

2. http://template-toolkit.org/

```
     Activities and results:\\
Partners collected a total of 145 accessions of blue-green algae ...
    Activities and results:\\
17 accessions have been added from Iran and Afghanistan ...
    Activities and results:\\
Trials were undertaken with four species ...
...
```

Those are indeed my evaluations of the various activities and results of the Alga project. Template.pm has recognised the lines "[% ... %]" as Template.pm code. It has reacted by filling in the appropriate values from my hash, task by task and work package by work package. [3]

One last problem - how to get the values into the hash?

# 3 How to fill in the hash

I just counted the elements that have to be completed in one of my Agency's evaluation forms. There are 316 such elements. 182 of them are items such as the coordinator's name, e-mail address, fax, and telephone number, the name of each work package, each task, and of each subtask. The values of all these items are already available, in the coordinator's report which I have to evaluate. So I wrote a Perl script to pull these values out of the report and store them in the hash. That left 134 items that have to be judged by the evaluator ("AchievementsAllWPs", "BudgetEvaluation", ..., ..., "Twelvemonthswhatprogress"). Veytsman and Shmilevich have the most elegant approach to capturing these values - the text is input straight into a relational database. That technology is a bit beyond

3. Observant readers may be wondering what really happens when I run Template.pm on the LaTeX source of this present article. There seem to be two Template.pm FOREACH statements in the current text. Doesn't the second FOREACH cause Template.pm to complain about an unbalanced loop? The answer is yes, it did, until I thought of adding a second [% END %] right here. (Actually, there are two [% END %]'s right here; the [% END %] statement that keeps Template.pm happy (but which you cant see because it is commented out by a % because LaTeX doesnt seem to like *verbatim* commands in footnotes), and another [% END %] statement so that you can see it (but which is ignored by Template.pm, because it is actually coded [\% END %\], for LaTeX). Thats the sort of contorted thinking that you find yourself having to do, when you mix LaTeX with other languages.)

my capacities, so I again settled on ASCII files. The evaluator gets one such file for each work package, plus one for the overall evaluation of the whole project. Each file is made of paragraphs copied from the Agency's evaluation form, explaining what is needed from the evaluator regarding e.g. Task activities and results. The final line of each paragraph is the key, an "=" sign, and a blank space - like this:
`!TaskActivitiesAndResults =`
Using emacs, Vim, or whatever, the evaluator fills in the blanks with lines of text (as many as you like). Then Perl does the rest. The lines written by the evaluator become the values of the keys, Perl passes the hash to Template.pm, and then (if all has gone well) runs pdfLaTeX on the output. The Agency gets the PDF by return.

# 4 Conclusion

Evaluators should devote all their attention to the words on the page; i.e. to the content of their report. They should not have to bother their heads about the form of their report. The best way of providing that form is undoubtedly something like Veytsman and Shmilevich's automatic report generator. But that requires a web server, a knowledge of SQL, and a certain investment of time and resources. Here I have described a poor man's alternative. And as a poor man I have discovered that when you need some string, or some sealing wax, the Perl community, and the TEX community, have that in abundance. So I close with big thank-yous to Henrik Just for his *W*riter2LaTeX[4], which is an now an integral component of OpenOffice.org[5], and to Andy Wardley for Template.pm, and its excellent documentation[6].

4. http://writer2latex.sourceforge.net/
5. http://extensions.services.openoffice.org/project/writer2latex
6. http://template-toolkit.org/docs/index.html

# 5 Afterword - Worked example

## 5.1 The evaluation report

The Agency has sent me a Microsoft Word file with boxes for my evaluation of the “Activities” and the “Results”, the “Achievements" and the “Methodology", of each “Task’ of each “Work Package” of this project. But their file is just a skeleton. Before I can even start, the pages labelled “Work Package” have got to be copied and pasted *n* times (for the project’s *n* work packages).

However, Perl has read the project Report, counted the Tasks and Work Packages, and made me an easy-to-use ASCII file. The first few lines look like this:-

```
!WPNumber                  :: 1
% TASK ACTIVITIES AND RESULTS
!TaskNumber                :: 1
!TaskActivitiesAndResults ::  ...
!TaskNumber                :: 2
!TaskActivitiesAndResults ::  ...

% CONCLUSION 1 OF 2 FOR THIS  WP -  ACHIEVEMENTS
!WPAchievements_etc        :: ...

% CONCLUSION 2 OF 2 FOR THIS  WP -  METHODOLOGY (IES)
% Reminder - choose between Unacceptable Poor Satisfactory Good Excellent
!WPMethodologyies          :: ...
!WPScoreAchievement        :: ...
!WPScoreMethodology        :: ...
```

Now, writing my Evaluation is now as simple as filling in the dots in the ASCII file. That done, we have to make the final report. For this we need a second Perl script which will invoke Perl Template appropriately. Three lines of Perl will do the trick

```
use Template;
my $tt = Template->new({ ABSOLUTE=>1});
$tt->process($template_file, \%hash, $outfile)
  || die "$tt->error";
```

That script fragment first instantiates a variable `$tt`. Then it processes the `$template_file`, using the `%hash`, thus making the LaTeX `$outfile`, all ready to run. The magic, of course, is contained in the `$template_file` and the `%hash`. I discuss these next.

## 5.2 The template

Lets start with `$template_file`. This began life as the Agency's blank report file, which they made in Microsoft Word. I've explained how I used Open Office to transform that file from Microsoft Word to LaTeX. Then at each point where something has to be filled in, I have put an incantation like this – `[% ...%]`.

For example, the `$template_file` begins so (I've skipped the preamble) ...

```
\begin{document}
  {\centering\sffamily\bfseries A TITLE ... \par}
  \bigskip
    \noindent{\sffamily\bfseries A SUBTITLE
    {[% Project.RefNo %]}
  }\\
  {\sffamily\bfseries Acronym: {[% Project.ShortTitle %]}
    }\\
```

Perl Template replaces each template element – for example `[% Project.ShortTitle %]` – by text from its associative array (discussed below). Perl Template will do this repetitively - i.e. it provides for loops. Here is an example of such a loop (here indexed by the scalar `$a`)

```
\section[OBJECTIVES, TASKS \ldots OF EACH WORK PACKAGE]
  {\sffamily OBJECTIVES, TASKS AND DELIVERABLES OF EACH WORK PACKAGE}
    [% FOREACH a IN WorkPackage.keys.sort -%]
      \subsection{Work Package {[% WorkPackage.$a.WPNumber %]}:
       {[% WorkPackage.$a.WPName %]}
   }
  {\sffamily {[% WorkPackage.$a.WPLeadPartnerNo %]}
```

That fragment was cut and pasted from `$template_file`. Perl Template runs a `FOREACH` loop, filling in the work package number, title, and my assessments, in the correct order.

## 5.3 The associative array

The text elements are stored in Perl's associative array %hash. The first few lines of the %hash look something like this

```
%hash = (
 'Project' => {
   'Email' => 'father.christmas@polenord.fi',
   'FaxNo' => '+ 358 0 1234567',
   'Finish_ddmmyyyy' => '01/04/2011',
   'FirstNameLastName' => 'Dr Santa Claus',
  },
 'Objective' => {
  '1' => {
   '1' => {
     'ObjectiveName' => ' Data collection'
    },
   '2' => {
     'ObjectiveName' => ' Data recording into database '
    }
  },
```

All the administrative items, e.g. `[% WorkPackage.$a.WPLeadPartnerNo %]`, come from Perl's parsing of the Report sent in by the project coordinator. The remaining comments and evaluations, e.g. `[% Project.AchievementsAllWPs %]`, come from Perl's parsing of my evaluation report.

## 5.4 Final comment

The form and the content of the evaluation report have been separated out. The form is a LaTeX document, made by Open Office org from the Agency's report form. The content is contained in a (Perl) associative array. Some comes from Perl's parsing of the coordinators report. The rest of the content comes from Perl's parsing of my evaluation form. The final LaTeX document, uniting content with form, is made by Perl Template.

THE PracTEX Journal

# Giving away a book

Jim Hefferon

Abstract

This article relates my experiences with offering a free downloadable math textbook, including the pdf and full LaTeX sources.

Jim Hefferon is Professor of Mathematics at Saint Michael's College, Colchester, Vermont (USA). He is also the maintainer of the US CTAN site and a is member of The PracTeX Journal Editorial Board.

You can contact him by sending an email to ftpmaint@tug.ctan.org

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/06/04

# Giving away a book

Jim Hefferon

In the early 90's I wrote an undergraduate textbook *Linear Algebra*. While writing it, I found useful the Free software tools that were available under the then-new Linux. Inspired by that, I made the book available for download under a Free license.[1]

A Free text was unusual then but today authors do it more often. Since the length of my experience with it may make this book stand out from other freely available ones, I hope some authors considering doing this will find interesting a few things I can say about that experience.

In particular, I will discuss advantages and disadvantages of using LaTeX for such a project. I will also take this chance to discuss some experiences with this project that have nothing to do with TeX — sorry.

## 1 The project

*Linear Algebra* is for a standard undergraduate course that in the US at least is often taken during a student's second year. I wanted to emphasize developing mathematical maturity in these young math students, up from the more formula-driven early classes toward the more proof-driven later courses (a full description is in the book's Preface). At the time that I wrote it, I knew of no text that was directed toward this goal.

I believe that people use it. For one thing, it is frequently downloaded, about 100,000 times in the last year. Another indicator is that in the decade that has passed it has always been on the front page of a Google search for "linear algebra," and often first on that page.

In addition to the PDF of the text, I offer the full LaTeX source including a PDF of the fully-worked answers to all exercises, even the proofs.

1. http://joshua.smcvt.edu/linearalgebra

Developing the worked answers to all the exercises has proved to be a big part of the book. I'll discuss below some of the associated technical problems but I'll note here that this development contributed greatly to the book's strength. While writing the answers, I made many, many improvements to both the exercises and the section body. It took a long time and it was a lot of work but I highly recommend this practice to authors.

## 2 LaTeX aspects

I put *Linear Algebra* up more than a decade ago (the Wayback Machine knows of it by January, 1998). What I offer now is essentially unchanged from what I offered then. Because I used LaTeX, there has been no bit rot—I can still easily build it from scratch. This is a tremendous advantage. An author providing material for free has nothing to gain from time spent on maintenance due to changes in versions of underlying tools.

A number of other advantages to a TeX-based solution have also become clear with time. Because it produces first-class output, an instructor could use it in class without apology. The source is very small (it fits on a single ancient storage device called a "floppy"), limiting the bandwidth I consume. And, there is a strong ecology of tools available, including editor add-ons, but also including the graphics program MetaPost, which I found very useful.

When I started the project, I knew only a little LaTeX. There were fewer packages than today and to do many things I had to do my own macro programming. As many other people have done, I used the just-in-time approach of having a need and consequently poking around in the LaTeX source, examining style files, and lurking on, and sometimes daring to ask questions on, the TeX mailing lists. Gradually, I built up a few style files that did some moderately sophisticated things (they are included in the book source).

I'll try to give a sense for this process by discussing just one area, producing the exercises and their answers.

My first problem was that I wanted to number the exercises in sequence with the theorems. This was an introductory-level problem but I remember that it gave me trouble since solving it required that I read the source of LaTeX. Anyone who has made a trip there knows that it is accompanied by considerable head-scratching.

The next problem was harder. I wanted to include the answers to the exercises in the source file, using this format.

```
\begin{exercises}
\item Prove that ..
  \answer{The proof begins ..}
\recommended\item Calculate the ..
  \answer{Let $x$ be ..}
\end{exercises}
```

Recommended exercises are marked with a check in the margin as a guide for readers going through the book on their own. Note that if an exercise has its own `\item` substructure then the answer is not entered per-item; rather the answer has its own `\item` structure.

While compiling the book, I need TEX to write the text of the answers to a separate file. I also need to include in that file the exercise number, etc. The `answers` package was a great help (Mike Piff, thank you!).

However, I had additional constraints. One is that I wanted to produce links in the two PDF files, first from the answer to the question, and also back from the question to the answer. This required some LATEX coding that I found tricky. (A problem that I never did solve is that readers must put both the book and answer PDF files in the same directory for the inter-linking to work.)

A second problem with exercise coding was that some instructors told me that they want to assign problems to hand in, and don't want to make up their own exercises, so having all the answers available precludes their using the book. For this, I developed an option that will produce only the answers to recommended exercises. I am not the only person ever to find TEX's `if` constructs hard to plumb, but eventually I got it to work. (For a while I offered for download only the answers to recommended exercises. To get the answers to all exercises, people had to write me to swear that they were either teaching the course or else were studying on their own. After a while the absurdity of this became compelling and in any event finding the entire set of answers online by googling became easy, so I now stick to offering all of the answers.)

The final problem, which I never solved, was the awkwardness of correcting LATEX errors in the answers. My workflow was to compile the document, which output the answers to a separate file, and then to compile those answers. Any

errors in the answer file results in a report giving line numbers from that file, of course. But that is not the location of the error in the original source file from which I worked. I hacked at this a bit, but at some point I felt that I should be writing the book, not spending all my time on the tool used for the book, and so I never fixed it.

In short, the advantages of LaTeX that have proved most useful for this project were its high quality output, its stability, and the widespread availability of tools. The main disadvantage was that to achieve some effects required that I do macro programming, some of which I found hard and time-consuming.

# 3 Other aspects

Providing the book free for download has had some clearly positive effects.

The biggest positive effect happens when a person says that my providing access to the mathematics enabled them to accomplish something they otherwise could not, often because of their circumstances such as that they are from a developing country. I get perhaps five or six of those emails a year. They make my day.

I also enjoy notes from anyone who has found it useful, either as a student or as an instructor.

Another positive effect is that I get bug reports, typically typos (lately mostly in answers to exercises). I don't know if authors of undergraduate texts distributed from traditional publishers receive many of these but I have had some readers who were quite precise, and that has been helpful.

That is, providing the materials freely has resulted in both exposure and good will that would have been hard to get otherwise.

There have also been some aspects of distributing the material in this way that were more mixed.

The first is that four separate times people have written that they were using the source to translate the text, accompanied by some initial material. Since the equations remain unchanged and the structure of the book is set, it would seem routine to translate the sentences and end with a usable text (I know nothing about translation, though). But, unfortunately, none of these promising projects seems to have been completed.

I also know of two tries at using the source to make a wiki. The first was very well done and contained the full text of the book, including all the illustrations (the author and I cooperated on these since they needed special handling). As with the other kind of translation, though, that I can tell the wiki projects were not successful in the sense that they never became dynamic documents to which people contributed improvements, etc.

The experiences of the wiki authors match my own. When I made the source available I imagined that it would allow some instructors to add or delete sections or exercises to their preference. In particular, at the end of each chapter are three or four sections of topics that are quick applications or developments of the material in the chapter, and I thought these might get some activity. To help this happen, I provide a booklet on how to get the book's source to compile.

With that in mind, on the download page I solicited contributions, starting by imposing on a few colleagues in my department to allow me to use their exams to seed my collection of problems. I also solicited topics including, based on some reader inquiries, something on linear algebra in Quantum Mechanics. However, no contributions have happened (I exchanged emails with one person about the Quantum Mechanics topic but in the end he concluded that it was not possible).

Finally, I will mention one negative aspect of free distribution. Some people have downloaded the book and put versions up for sale on online publishers. This falls within the license but I confess to finding it mildly annoying. One way that it is annoying is that the editions of these are not updated as bugs in the book are fixed, so there are versions for sale when a free version has fewer errors. Perhaps I should put up my own version, but to this point I have preferred to stick to online distribution.

In summary, there are a number of advantages to providing the material freely, including exposure. However, at least this project has not attracted any contributions more extensive than bug reports.

# 4 Possibilities

Were I starting this project today, I would think about about filling it with interactive goodness. For instance, a graphic could have the user specify two basis vectors in the plane, and when that user mouse-drags a vector around the plane, the graphic could show the coefficients of the basis vectors needed to express the

mouse's vector. Another might show a two-by-two matrix with sliders in the four entries, and display the transformation of the plane resulting from the slid-to values of the entries.

Providing such things was not practical when the book was written. However, today PDF is an open standard so fixing reliance on it as a delivery platform seems safer. And, with PDF allowing the inclusion of JavaScript, there may be a realistic way to get cross-platform interactivity.

Of course, as I mentioned above, the most exciting possibility would be to develop a community of people to contribute such applications. Anyone who watches an active Internet community has to be impressed with the tremendous creativity and energy that can arise when a group of great people get going.

# 5 Closing

Free distribution has some real advantages (for one, no marketing meetings!) but some trade-offs as well.

Because there are stable standards, including LaTeX, a private individual can realistically offer a text free for download without maintenance issues. Producing the work in LaTeX probably involves some coding (this may be mitigated by the fact that there are more LaTeX packages today than a decade ago, so the need for individual coding may be reduced).

If you are considering such a project, good luck!

THE PracTEX Journal

# Useful Vector Graphic Tools for LaTeX Users

Tomas Morales de Luna

Abstract

English

This paper aims to present some useful tools for creating vector graphics that can be included in LaTeX documents. Among all the tools available, we focus on those that can produce graphics in an easy way, and that can include LaTeX formulas. In particular, we present here three useful tools: Xfig, LaTeXDraw, and Matplotlib. While the two first are intended to produce sketches and figures, the last is useful for producing graphs, charts, and contours.

Spanish

Este artículo intenta presentar algunas herramientas útiles para la creación de gráficos vectoriales que puedan ser incluídos en los documentos LaTeX. Entre las distintas herramientas disponibles, nos centramos en aquellas que puedan generar gráficos de una forma fácil y que puedan incluír fórmulas de la misma forma en que se escribirían en un documento. En particular, presentamos tres herramientas: Xfig, LaTeXDraw, and Matplotlib. Mientras que las dos primeras están enfocadas a realizar diferentes bosquejos o dibujos, la última nos ayudará a producir gráficos y curvas de nivel.

Tomás Morales, Ph.D., is Assistant Professor of Mathematics at Universidad de Córdoba and a member of the research group EDANYA. He started using LaTeX some years ago when he began working on his Ph.D. thesis, and has continued using LaTeX extensively. He has found LaTeX essential for writing technical documents for publication in Applied Mathematics. He also uses uses LaTeX to produce slides for talks and conferences, and to write course notes, exercises, exams, and other documents for his students.

You can contact him at http://www.uco.es/~ma1molut or by sending email to tomas.morales@uco.es.

- PDF version of paper
- Article source
- Screen version
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/05/25

# Useful Vector Graphic Tools for LaTeX Users

T. Morales de Luna

Email Tomas.Morales@uco.es
Address Dpto. de Matemáticas
Escuela Politécnica Superior
Universidad de Córdoba
Córdoba 14071
Spain

Abstract This paper presents some useful tools for creating vector graphics that can be included in LaTeX documents. Of all the tools available, we focus on those that can produce graphics easily, and that can include any LaTeX math formula. In particular, we present three useful tools: `Xfig`, `LaTeXDraw`, and `Matplotlib`. While the two first are intended to produce sketches and figures, the last will produce graphs, charts and contours.

## 1 Introduction

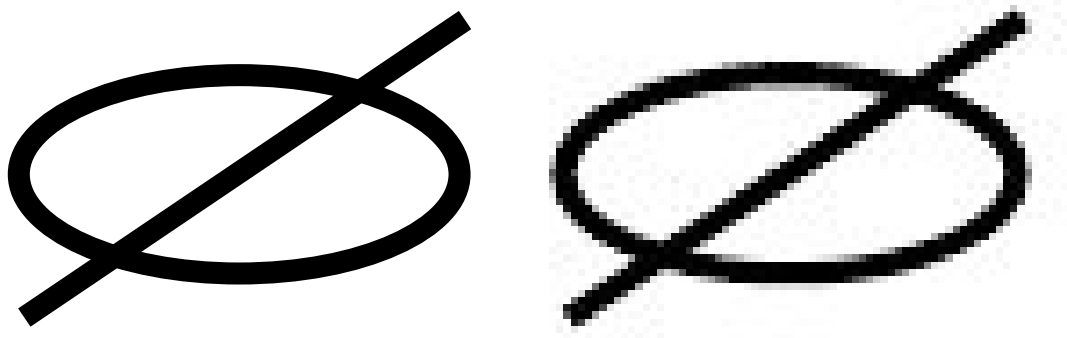
In this article, we describe some vector graphic tools that work with LaTeX code so that users can easily produce *good* graphics to be included in LaTeX documents.

## 2 Including graphics in LaTeX documents

Although this article focuses on tools for creating graphics rather than on LaTeX packages for including graphics, it is important to say a few words about how to include graphics in `tex` documents. For more detail, please see [1] and [2].

First of all, it is important to recall the difference between vector and bitmap graphics. While vector graphics behave well for scaling and rotation without loss of quality, the same is not true with bitmap graphics. So, whenever possible, we will use vector graphics in our documents.

Once you have drawn your figures, you can easily include them in your document by using the package `graphicx`. To use this package, include the following in the preamble of your `tex` document:



(a) Vector graphic (b) Bitmap graphic

Figure 1: Difference between vector and bitmap graphics.

```
\usepackage{graphicx}
```

or if you are going to produce a `pdf` file with `pdflatex`, use

```
\usepackage[pdftex]{graphicx}
```

Once this package is in place you can include a figure by writing

```
\includegraphics[options]{myfigure}
\caption{mycaption}
```

For example, Figure 1 presents two subfigures placed side by side. This was made possible by using

```
\usepackage{subfigure}
```

. This shows how it is used:

```
\begin{figure}
  \centering
  \subfigure[Vector graphic]
  {
    \includegraphics[width=0.47\linewidth]{vector}
  }
  \subfigure[Bitmap graphic]
```

```
  {
    \includegraphics[width=0.47\linewidth]{bitmap}
  }
\caption{Difference between vector and bitmap graphics}
\end{figure}
```

An interesting feature of the `graphicx` package for figures is that it lets you scale, rotate, trim, and apply other features. For further details, please refer to [1] and [3].

Another interesting TEX macro package for generating graphics is `pgf`. It is platform- and format-independent and works together with the most important TEX back-end drivers, including `pdftex` and `dvips`. It comes with a user-friendly syntax layer called `TikZ`. See [2] for more details.

# 3 Creating graphics

A common question is, *How do I include formulas or TEX code in the picture?*. Well, you can use your favorite bitmap graphic editor and, when possible, one that produces vector graphics, but it is possible your favorite graphics editor may not be able to do the job. A work-around might be to generate two pictures: a formula using a temporary TEX file, and the background picture, drawn in your bitmap editor. Then you copy the formula and paste it on the background picture. We know that this is not a good practice because of the low-quality image, unless you do it with care (meaning that you should maintain the vector properties of the image when copying-pasting).

Another option is to generate just the picture, include it in your document and then place the formulas over it by including the command `\pgfputat`. This command lets you place almost anything at a given absolute position. For instance, the code below will produce Figure 2.

```
\begin{pgfpicture}{0cm}{0cm}{3cm}{3cm}
\pgfputat{\pgfxy(1.5,0)}{\pgfbox[center,center]{$\int_0^\infty x dx$}}
\includegraphics[width=3cm]{vector}
\end{pgfpicture}
```

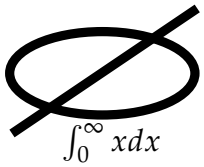



Figure 2: Using the command `\pgfputat`

The problem is to put the formulas at the correct position, but you will discover techniques that give the desired result.

Another method is to generate the picture using `Pstricks`. This way, you have control over your picture. Consider the example given in the `pgf` user guide that produces the Figure 3:

```
\begin{pgfpicture}{0cm}{0cm}{5cm}{2cm}
\pgfputat{\pgfxy(1,1)}{\pgfbox[center,center]{Hi!}}
\pgfcircle[stroke]{\pgfxy(1,1)}{0.5cm}
\pgfsetendarrow{\pgfarrowto}
\pgfline{\pgfxy(1.5,1)}{\pgfxy(2.2,1)}
\pgfputat{\pgfxy(3,1)}{
\begin{pgfrotateby}{\pgfdegree{30}}
\pgfbox[center,center]{$\int_0^\infty xdx$}
\end{pgfrotateby}}
\pgfcircle[stroke]{\pgfxy(3,1)}{0.75cm}
\end{pgfpicture}
```

This can be tricky and in general it is more difficult to draw graphics with commands than by using your mouse.

## 3.1 The `Xfig` program

`Xfig` [4] is a program that can draw vector graphics, and easily combine them with LaTeX formulas. Although this program has a challenging graphical user interface, it is still a handy tool.

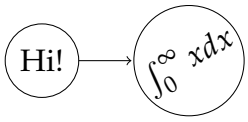



Figure 3: This example was captured from `pgf` user guide.

In order to include LaTeX code with graphics, we have to launch the program with the *special-text* flag.

➤ Step 1. Run the following command in a shell

```
xfig -specialtext
```

Once `xfig` is running, you can draw pictures and place any LaTeX equation or formula with the *insert text tool*. Use this in the usual way, but put the LaTeX markup between $ symbols. When finished, save your picture.

➤ Step 2. Draw your picture and add formulas between $ symbols.

➤ Step 3. Save the obtained figure with `.fig` extension.

Next, we are going to use the shell command `fig2dev` to produce the desired figure. Here, I assume that you used the filename `myfigure.fig`.

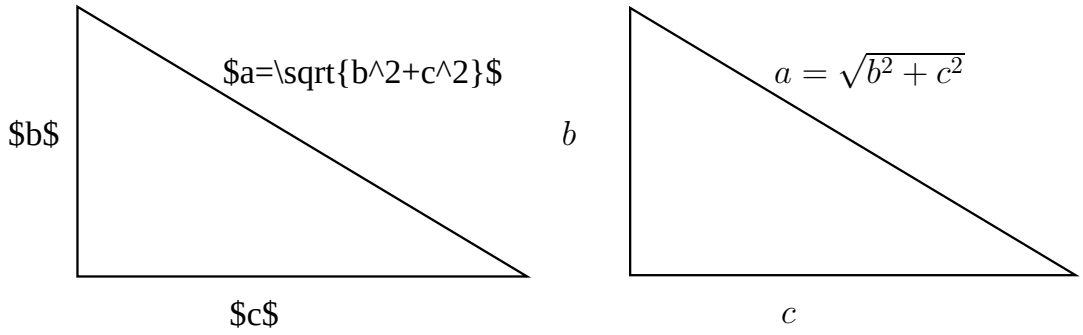
➤ Step 4. Run the following commands in the shell.

```
fig2dev -L pstex myfigure.fig > myfigure.pstex_t
fig2dev -L pstex_t -p myfigure.pstex_t myfigure.fig > myfigure.temptex
```

The first command generates `.ps` from `.fig`, and the second one generates `.tex` commands from `.fig` based on the specifications in the `.ps` file.

➤ Step 5. Create the file `myfigure.tex` with the following content

```
\documentclass{article}
\usepackage{graphicx,epsfig,color}
\pagestyle{empty}
\begin{document}
\input{myfigure.temptex}
\end{document}
```




(a) Plot produced with Xfig (b) Final figure obtained

Figure 4: Using `Xfig` to produce graphics

➤ Step 6. Create the corresponding `.eps` file by running the following commands in a shell

```
latex myfigure.tex
dvips -E -q -o myfigure.eps myfigure.dvi
```

If you prefer a `.pdf` file, use `epstopdf` to convert the `.eps` figure to `.pdf`.

You may put all the previous commands in a script, or you may use the one available in [5].

## 3.2 The `LatexDraw` program

A more recent program for drawing graphics is `LaTeXDraw`[6]. It is a free `PSTricks` code generator or `PSTricks` editor distributed under the GNU GPL. `LaTeXDraw` was developed in `JAVA`, so it runs in any operating system. The graphical user interface of `LaTeXDraw` is similar to the ones found in many graphics editors.

➤ Step 1. Draw your picture and include formulas between $

Since the resulting picture will be used in a LaTeX document, you may place a formula using the text button available in the toolbar in `LaTeXDraw`. Type the formulas between $ as you would normally do.

➤ Step 2. Export the picture as `PSTricks` in a `tex` file.

Once you have drawn your picture, you can save the `PSTricks` code with the menu File → Export as → PSTricks code.

A figure similar to 4(a) will produce the following `.tex` file with `LaTeXDraw`:

```
% Generated with LaTeXDraw 2.0.3
% Tue Dec 29 12:38:16 CET 2009
% \usepackage[usenames,dvipsnames]{pstricks}
% \usepackage{epsfig}
% \usepackage{pst-grad} % For gradients
% \usepackage{pst-plot} % For axes
\scalebox{1} % Change this value to rescale the drawing.
{
\begin{pspicture}(0,-1.5507812)(6.1871877,1.5307813)
\pspolygon[linewidth=0.04](0.701875,1.5107813)(0.701875,-0.94921875)
(5.481875,-0.96921873)
\usefont{T1}{ptm}{m}{n}
\rput(0.25546876,0.22578125){$b$}
\usefont{T1}{ptm}{m}{n}
\rput(2.5754688,-1.3742187){$c$}
\usefont{T1}{ptm}{m}{n}
\rput(4.3554688,0.8857812){$a=\sqrt{b^2+c^2}$}
\end{pspicture}
}
```

Be sure to include the corresponding packages if you use this code in your document.

If you prefer to generate independent vector graphics that you can include in your document with the `\includegraphics` command, you can do so by creating the file `myfigure.tex` as follows.

➤ Step 3. Create a new `.tex` with the following content.

```
\documentclass{article}
\usepackage{pstricks}
\usepackage{pst-plot}
\usepackage{pst-eps}
\usepackage{pst-grad}
\pagestyle{empty}
\begin{document}
```

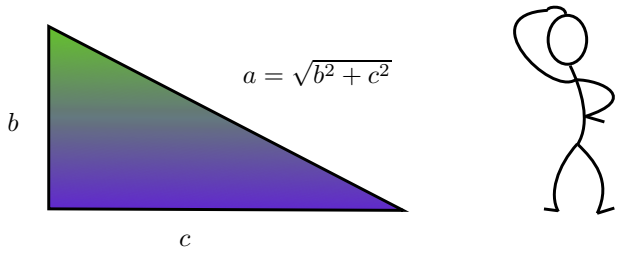



Figure 5: A figure drawn with LaTeXDraw.

```
\begin{TeXtoEPS}
\begin{pspicture}
<...>
\end{pspicture}
\end{TeXtoEPS}
\end{document}
```

Replace <...> with the corresponding code generated by LaTeXDraw. It is important to place the pspicture inside the environment TeXtoEPS, as this will provide LaTeX with the correct bounding box for the picture. If you omit it, you may get a final picture that has been trimmed or in a page size far larger than the size of the picture. You must not include all the packages that are shown above, but just the ones needed for your graphic. Now, all that remains to do is:

➤ Step 4. Execute the commands

```
latex myfigure.tex
dvips -E -q -o myfigure.eps myfigure.dvi
```

With this program you can easily draw professional-looking graphics that include LaTeX formulas, like the one shown in 5.

# 4 Matplotlib in LaTeX documents

When writing technical documents with LaTeX, it is common to plot graphs, charts, contour plots etc using Matlab, Mathematica, or other software, and then include them in your document.

On the one hand, Matlab and Mathematica are good commercial programs, but there are other free alternatives such as Matplotlib [7]. Matplotlib is a plotting library for the Python programming language and its NumPy numerical mathematics extension. It provides an object-oriented API which allows you plot picturess to be embedded into applications using generic GUI toolkits, like wxPython, Qt, or GTK. There is also a procedural pylab interface based on a state machine (like OpenGL), designed to closely resemble Matlab. The advantage is that it is open-source software.

You could use Matplotlib to generate your plots and then include them in your documents, but you can also include a LaTeX package that lets you to use Python code directly in your document: python [8]. To do this, add the line \usepackage{python} to the preamble and then insert any Python code inside the environment python (you may need to download this package if you do not have it in your system). In this way, the images are automatically generated when you compile your .tex by including the flag -shell-escape to the command line.

To use this method, do the following:

- Step 1. Write \usepackage{python} before the \begin{document}
- Step 2. Insert any Python code inside the environment python
- Step 3. Compile your tex file using the command

```
pdflatex -shell-escape your_tex_file.tex
```

Let's look at some examples.

By writing the following code into the file example.tex

```
\documentclass{article}
\usepackage[pdftex]{graphicx}
\usepackage{python}
\begin{document}
\begin{figure}
{\Huge Document including a plot}
\begin{python}
from pylab import *
x = linspace(-4*pi,4*pi,200)
plot(x,sin(x)/x)
```

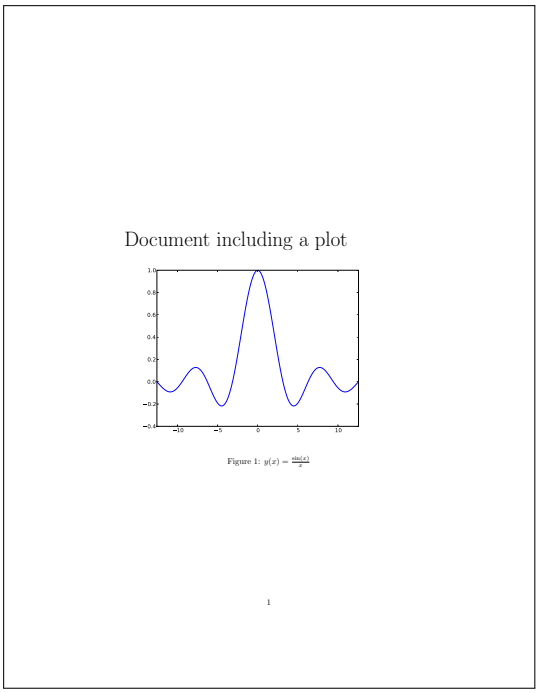



Figure 6: Plotting $y = \dfrac{\sin(x)}{x}$ with python

```
xlim(-4*pi,4*pi)
savefig('plot-include.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include.pdf}'
\end{python}
\caption{$y(x)=\frac{\sin(x)}{x}$}
\end{figure}
\end{document}
```

and then executing the command

```
pdflatex -shell-escape example.tex
```

we get the pdf file corresponding to Figure 6.

There are many other examples. For instance, the code

```
\begin{python}
from pylab import *
x = arange(-3.0, 3.0, .025)
y = arange(-3.0, 3.0, .025)
X, Y = meshgrid(x, y)
Z1 = bivariate_normal(X,Y,1.0,1.0,0.0,0.0)
Z2 = bivariate_normal(X,Y,1.5,0.5,1,1)
Z = 10.0 * (Z2 - Z1)
CS = contour(X, Y, Z)
clabel(CS, inline=1, fontsize=10)
savefig('plot-include2.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include2.pdf}'
\end{python}
```

will produce Figure 7(a), and

```
\begin{python}
from pylab import *
figure(1, figsize=(6,6))
ax = axes([0.1, 0.1, 0.8, 0.8])
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = [15,30,45, 10]
explode=(0, 0.05, 0, 0)
pie(fracs, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True)
title('Raining Hogs and Dogs', bbox={'facecolor':'0.8', 'pad':5})
savefig('plot-include3.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include3.pdf}'
\end{python}
```

will produce Figure 7(b).

`Matplotlib` can even include LaTeX commands as is shown in [9].

# 5 Conclusion

We have shown several ways to include graphics in LaTeX documents using two programs: `Xfig` and `LaTeXDraw`. While both give good results, the second is more recent and user-friendly.

Both of these programs will draw sketches and similar figures, but in order to do graphs, charts or contour plots, `Matplotlib` is the better choice. You can also embed `Matplotlib` and `Python` code directly in your document.

# References

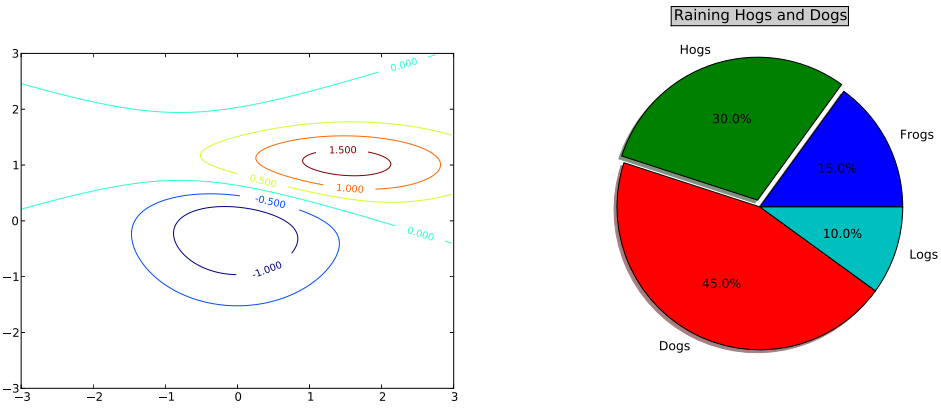

[1] Klaus Hoeppner. *Strategies for including graphics in LaTeX documents*. The PracTEX Journal, 2005. No 3.
http://www.tug.org/pracjourn/2005-3/hoeppner/

[2] Claudio Beccari . *Graphics in LaTeX*. The PracTEX Journal, 2007. No 1.
http://www.tug.org/pracjourn/2005-3/hoeppner/

[3] *Enhanced support for graphics*
http://www.ctan.org/tex-archive/macros/latex/required/graphics/

[4] *Xfig*.
http://www.xfig.org/

[5] *Conversion tools*.
http://www.few.vu.nl/~wkager/tools.htm


(a) (b)

Figure 7: Using `Matplotlib` to produce graphics

[6] *LaTeXDraw*.
http://latexdraw.sourceforge.net

[7] *Matplotlib*
http://matplotlib.sourceforge.net/index.html

[8] *Embedding python into LaTeX*
http://www.imada.sdu.dk/~ehmsen/pythonlatex.php

[9] *pylab_examples example code: integral_demo.py*.
http://matplotlib.sourceforge.net/examples/pylab_examples/integral_demo.html

THE PracTEX Journal

# Generating Academic Certificates (Portuguese)

Francisco Reinaldo et al

Abstract

English

In this paper we present an easy method for automatically generating academic certificates with scanned signatures by using CSV and few LaTeX commands.

Português

Neste artigo, nós apresentamos como usuários comuns em LaTeX podem gerar automaticamente certificados acadêmicos com assinaturas digitalizada através de uma base de dados CSV e algumas instruções em LaTeX.

Francisco Reinaldo é atualmente professor na Área de Exatas no UnilesteMG, Brasil, em específico, o curso de Computação - Sistemas de Informação. No UnilesteMG, sua área de pesquisa é Inteligência Computacional (Diretor do LIC), mas dedica boa parte de seu tempo com estudos envolvendo LaTeX.

Victor Vasconcelos Moreira cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Estagiário do Projeto Informática Solidária - INFOSOL e colaborador do Laboratório de Inteligência Computacional – LIC, ambos os projetos sustentados pelo UnilesteMG.

Maria Tereza de Castro Costa cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiária no Laboratório de Inteligência Computacional (LIC), trabalhando com Redes Neurais Artificiais.

Tiago Faria Bicalho cursa o 7º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiário no Laboratório de Inteligência Computacional (LIC) e colaborador no Projeto Informática Solidária - INFOSOL.

Os autores podem ser contatados através de reinaldo.opus@gmail.com

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/29

# Gerando Certificados Acadêmicos e inserindo Assinaturas Digitalizadas

Francisco Reinaldo[1], Maria Tereza de Castro Costa[2], Tiago Faria Bicalho[3], and Victor Vasconcelos Moreira[4]

Email [1]reinaldo.opus@gmail.com, [2]maryxb@gmail.com, [3]tiagofariabicalho@gmail.com, [4]victorvasconcelosfox@gmail.com

Resumo In this paper we present how common users can generate academic certificates with scanned signatures automatically by using CSV and few instructions in LaTeX2ε.

## 1 Introdução

Quando pensamos em cadastrar certificados, pensamos em dolorosas horas na frente do computador preenchendo diferentes dados em um modelo feito por alguém que só conhecia Word como editor de textos - seu canivete suíço. Contudo, não necessariamente precisa ser assim.

Este artigo apresenta como um gerador automático de certificados acadêmicos foi desenvolvido em LaTeX2ε a partir dos dados importados de um arquivo CSV. Também apresenta um dos modelos de certificados desenvolvidos em LaTeX2ε, no Laboratório de Inteligência Computacional (LIC), para ser utilizado nos vários cursos de informática básica do Projeto Informática Solidária (INFOSOL) do UnilesteMG.

## 2 Sobre o certificado

O certificado contém uma base com dados heterogêneos. Acompanhando a tendência de portabilidade entre plataformas com facilidade de implementação, temos então CSV para construir a base de dados do certificado e LaTeX2ε para implementação do certificado.

Arquivos do tipo CSV contém dados separados por vírgulas, explicando sua sigla CSV em "Comma-separated values". CSV é perfeitamente compatível em todas as plataformas computacionais. Arquivos CSV proporcionam clareza, rapidez e flexibilidade na geração de certificados do tipo acadêmicos. Os arquivos CSV são utilizados em ferramentas de importação e exportação de dados. Várias ferramentas atualmente geram arquivos CSV. CadLinha no arquivo CSV corresponde a um registro de uma base de dados, mas se pensarmos em tabela ou planilha de dados então teremos umLinha desta tabela ou planilha. Dentro da base de dados, os campos com conteúdos seriam os dados delimitados por vírgulas e cada campo deste formulário é uma coluna da tabela.

LaTeX2$\varepsilon$ foi escolhido por ser o tipo de ferramenta tipográfica com um altíssimo grau de qualidade e portabilidade. Através das instruções em LaTeX2$\varepsilon$, podemos ler base de dados em CSV e gerar diferentes certificados acadêmicos.

# 3 Desenvolvimento

Neste artigo, nós estamos trabalhado com o arquivo chamado `certificado.tex` que está disponível dentro do pacote de códigos-fonte deste artigo, Figura 1, e que contém o seguinte código:

```
% ---------------------------------------------------------------------
\documentclass[landscape,a4paper,12pt]{letter}
\usepackage[a4paper,left=2.5cm,right=2.5cm,top=2.5cm,bottom=2.5cm,headsep=1cm,footskip=1.8cm]{geometry}
\usepackage{setspace}
\usepackage{csvtools}
\usepackage[brazil]{babel}
\usepackage[utf8]{inputenc}
\usepackage{graphicx,color}
\renewcommand{\familydefault}{phv}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhf{}
\usepackage{eso-pic}
\newcommand{\assinaturamachado}{\AddToShipoutPicture*{
\put(85,70){
  \includegraphics[scale=0.4]{AssinaturasDigitais/assinatura_machado.jpg}}
\put(75,105){
  \begin{Large}Antônio Machado Filho\end{Large}}
\put(94,85){
  \begin{Large}\emph{Coordenador CSI}\end{Large}}
  }}
\newcommand{\assinaturareinaldo}{\AddToShipoutPicture*{
\put(345,100){
  \includegraphics[scale=0.6]{AssinaturasDigitais/Assinatura_Reinaldo.jpg}}
\put(365,105){
  \begin{Large}Francisco Reinaldo\end{Large}}
\put(345,85){
  \begin{Large}\emph{Coordenador INFOSOL}\end{Large}}
  }}
\newcommand{\assinaturasonaly}{\AddToShipoutPicture*{
\put(585,70){
  \includegraphics[scale=0.4]{AssinaturasDigitais/Assinatura_Sonaly.jpg}}
\put(605,105){
  \begin{Large}Sonaly T. S.Gabriel \end{Large}}
\put(600,85){
  \begin{Large}\textit{Coord. de Extensão}\end{Large}}
  }}
% ---------------------------------------------------------------------
\begin{document}
\renewcommand{\headrulewidth}{0pt}

\applyCSVfile{2009_CSV_PDF/AlunosParticipantes2009.csv}{%
\begin{letter}{}
\doublespacing
{\Large
\mbox{ }\\[1em]
Declaramos que {\insertArtigoDefinido} {\insertDiscente} participou do Projeto de Extensão chamado
``Informática Solidária (INFOSOL)'' na função de ministrar cursos de informática, no período de
{\insertDataInicio} à {\insertDataFim}, com carga horária total de {\insertHoras} hora(s), cumprindo
integralmente o plano de trabalho.
\\ [3em]

\begin{flushright}
    Coronel Fabriciano, 8 de dezembro de 2009.\\[4em]
\end{flushright}
    \assinaturamachado                    \assinaturareinaldo            \assinaturasonaly
}
\end{letter}}
\end{document}
```

Figura 1: Arquivo `certificado.tex`.

Explicando as principais instruções utilizadas na implementação do certificado, Figura 1, temos:

- Linha 2: define o documento e o tamanho da fonte
- Linha 3: define as medidas da margem do certificado
- Linha 5: inclui o pacote de comandas a serem trabalhados com CSV.
- Linha 9: altera a fonte do documento para ARIAL.
- Linha 11: define o estilo de página para ajustar linhas de cabeçalhos e rodapés.
- Linha 14: cria uma macro com um nome. Esta macro conterá a assinatura digitalizada e a respectiva posição no doc.
- Linhas 15, 17 e 19: posicionam o objeto especificado (texto ou imagem) pelo argumento nas coordenadas indicadas. Optamos pelo comando `put` para que o objeto não se movesse, caso o texto acima tivesse mais linhas.
- Linha 16: inclui a assinatura digitalizada no documento.
- Linhas 18 e 20: incluem o texto abaixo da assinatura
- Linha 39: marca o início do documento.
- Linha 40: remove Linha feita por `fancyhdr`, no cabeçalho.
- Linha 42: faz a conexão com o arquivo CSV. O nome deste arquivo está entre `{ }`
- Linha 43: marca o início de uma classe necessária à formatação existente.
- Linha 44: implementa o espaçamento duplo.
- Linha 46: deixa umLinha em espaço para fazer uma quebra.
- Linha 47: contém em forma de comando os nomes do cabeçalho já definidos no arquivo CSV, tais como `\insertDiscente`, `\insertHoras`
- Linha 53: `\assinaturamachado`, `\assinaturareinaldo` e `\assinaturasonaly` incluem as macros que contém as assinaturas e seus textos.
- Linha 55: marca o fim de uma classe necessária à formatação existente.
- Linha 56: marca o fim do documento.

Como dito anteriormente, a nossa base de dados foi construída em um arquivo CSV. O arquivo `AlunosParticipantes2009.csv`, também encontrado no pacote deste artigo, ver Figura 2, contém a base de dados:

```
1  ArtigoDefinido,Discente,DataInicio,DataFim,Horas
2  o aluno,CHELDER LINS SIMÃO,02/03/2009,01/12/2009,16
3  o aluno,MARCELO HERINGER AOKI,02/03/2009,01/12/2009,16
4  o aluno,TIAGO FARIA BICALHO,02/03/2009,01/12/2009,70
5  o aluno,VICTOR VASCONCELOS MOREIRA,02/03/2009,01/12/2009,630
```

Figura 2: Arquivo `AlunosParticipantes2009.csv`.

Explicando o código da Figura 2, temos:

- Linha 1 contém os nomes do cabeçalho que serão importadas para o LaTeX2$\varepsilon$ na forma de comandos `\insert...`.
- Linha 2 em diante contém os dados que serão inseridos no certificado.

Por questões de segurança, não inserimos as assinaturas. Contudo, o código pode ser reproduzido sem problemas, desde que arquivos e comandos foram ajustados para tal.

# 4 Conclusão

A grande vantagem da utilização do LaTeX2$\varepsilon$ é que o autor não se preocupa com a formatação do certificado, ao contrário dos sistemas editores de texto que utilizam a tecnologia WYSIWYG (“What You See Is What You Get”) onde o autor acaba se perdendo durante a preparação do seu documento.

THE PracTEX Journal

# Developing software with Doxygen & LaTeX (Portuguese)

Francisco Reinaldo et al

Abstract

English

In this paper we show how programmers can document source code and have updated reports during the development/deployment phase. We focus mainly on two key tools: Doxygen and LaTeX2e.

Português

Neste trabalho, nós apresentamos como os programadores podem documentar códigos-fonte e ter relatórios atualizados durante a elaboração / fase de implantação. Nós nos concentramos principalmente em duas ferramentas especificamente destinadas a essa finalidade: Doxygen e LaTeX2e.

Francisco Reinaldo é atualmente professor na Área de Exatas no UnilesteMG, Brasil, em específico, o curso de Computação - Sistemas de Informação. No UnilesteMG, sua área de pesquisa é Inteligência Computacional (Diretor do LIC), mas dedica boa parte de seu tempo com estudos envolvendo LaTeX.

Victor Vasconcelos Moreira cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Estagiário do Projeto Informática Solidária - INFOSOL e colaborador do Laboratório de Inteligência Computacional – LIC, ambos os projetos sustentados pelo UnilesteMG.

Maria Tereza de Castro Costa cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiária no Laboratório de Inteligência Computacional (LIC), trabalhando com Redes Neurais Artificiais.

Tiago Faria Bicalho cursa o 7º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiário no Laboratório de Inteligência Computacional (LIC) e colaborador no Projeto Informática Solidária - INFOSOL.

Os autores podem ser contatados através de reinaldo.opus@gmail.com

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# Doxygen e LaTeX2ε: As definitivas ferramentas para documentar seu código-fonte

Francisco Reinaldo[1], Maria Tereza de Castro Costa[2], Tiago Faria Bicalho[3], and Victor Vasconcelos Moreira[4]

Email [1]reinaldo.opus@gmail.com, [2]maryxb@gmail.com, [3]tiagofariabicalho@gmail.com, [4]victorvasconcelosfox@gmail.com

Resumo In this paper we present how programmers can document source-code and have updated reports during the elaboration/implemention phase. We focus mainly on two tools specifically aimed at this purpose: Doxygen and LaTeX2ε.

## 1 Introdução

Profissionais do desenvolvimento de sistemas computacionais comumente documentam seus códigos-fonte ou parte deles somente após o término da implementação. Infelizmente, este tipo de decisão não acompanha a real preocupação de manutenibilidade no desenvolvimento de programas e faz com que o programa em desenvolvimento ou a documentação gerada não esteja sincronizada. Outro ponto a ressaltar é que a documentação acontece longe dos arquivos de código-fonte, o que inviabiliza sua manutenção e confunde a equipe de planejamento com informações antigas.

Este artigo apresenta duas ferramentas para solucionar o problema no sincronismo da documentação de programas e a compatibilidade da ferramenta de apoio. Primeiramente tem-se Doxygen como um gerador de documentação de software junto ao desenvolvimento e/ou atualizações de códigos-fonte e LaTeX2ε como formatador tipográfico adequado. Com Doxygen e LaTeX2ε, a documentação já formatada para apresentação como relatório final é automaticamente gerada pela extração dos comentários do programador/analista e em paralelo durante a implementação do código-fonte. Os membros do Laboratório de Inteligência Computacional (LIC), do Curso de Computação - Sistemas de Informação,

do Centro Universitário do Leste de Minas Gerais (UnilesteMG, Brasil), utilizam estas ferramentas durante a implementação de código para geração de relatórios atualizados.

## 2 Revisão de Literatura

TEX é um excelente sistema de tipografia desenvolvido por Donald E. Knuth para produção de livros, artigos e relatórios que exigem alta qualidade tipográfica. TEX é de fato um processador de macros que também possui uma poderosa flexibilidade para programação. LATEX2$\varepsilon$ é um conjunto de macros de TEX que implementam um sistema de preparação de documentos. O LATEX2$\varepsilon$ define uma linguagem de marcação de mais alto nível e que permite descrever o documento em termos de sua estrutura lógica e não apenas do seu aspecto visual. Usando diferentes classes de documentos e pacotes adicionais, o usuário de LATEX2$\varepsilon$ pode produzir uma grande variedade de leiautes. A primeira versão de LATEX2$\varepsilon$ largamente utilizada foi a 2.09, lançada em 1985[2] por Leslie Lamport a partir do TEX desenvolvido por Donald Knuth[1].

Noutra vertente, Doxygen é uma ferramenta geradora de documentação que oferece os recursos necessários para documentar os diferentes aspectos do código-fonte implementado através dos comentários dos programadores/analistas nas linguagens de programação `C/C++/C#`, `Java`, `IDL` (Corba, Microsoft, e KDE-DCOP flavors) e para algumas extensões como PHP. Doxygen funciona em plataformas UNIX, Microsoft e Macintosh OS X. Doxygen é o tipo de ferramenta que pode ajudar o programador a documentar seu código no formato *online* em HTML e manuais de referência *offline* no formato PDF conectado através do LATEX2$\varepsilon$. A documentação é extraída diretamente do código-fonte, deixando a documentação consistente dentro do código-fonte. Para maiores informações, a documentação original do sistema Doxygen bem como a aplicação estão disponíveis *online* no endereço `http://www.doxygen.org/`.

# 3 Desenvolvimento

## 3.1 Inserindo os comentários no código-fonte com Doxygen

A função primária do Doxygen é extrair comentários definidos no código-fonte e gerar relatórios ou manuais. Nesta mesma linha de raciocínio o LaTeX2$\varepsilon$ é utilizado para criar um manual *offline* de referência de boa qualidade e mulitplataforma. Quando unidas, as ferramentas Doxygen e LaTeX2$\varepsilon$ oferecem relacionamento entre os vários elementos que podem ser visualizados através de gráficos de dependências, gráficos hierárquicos e diagramas de colaboração que podem ser gerados automaticamente.

Para que os comentários possam ser localizados e extraídos do código-fonte pelo Doxygen e formatados para LaTeX2$\varepsilon$, as informações para o Doxygen precisam ser referenciadas por comandos especiais. As formas mais comuns para definir comentários em linha para o Doxygen são aqueles que usam blocos de comentário. Assim, não é necessário o caracter * antes dos comandos do Doxygen ao iniciar e terminar um bloco de comentários, apesar de autores utilizarem este sinalizador para uma clara legibilidade do código-fonte, como seguem os exemplos abaixo.

**Exemplo 1**: Para comentários de linha simples.

```
/** comentário. **/
/// comentário. ///
/*! comentário. !*/
//! comentário. !//
```

**Exemplo 2**: Para comentários multi-linha:

```
/**
 * comentário
 */
```

Dessa forma, o Exemplo 2, de estrutura de comentário acima apresentado, abre espaço para uma descrição mais detalhada, reconhecendo uma série de comandos do Doxygen. Os comandos a serem inseridos nesta estrutura tem o formato `@comando` ou `\comando`. Por exemplo, o comando `\brief` permite o programador definir uma descrição compact em sua linha de código.

Nesta primeira etapa criamos as seções da documentação, tal como apresenta abaixo:

```
/**
\mainpage (Nome do sistema)
\image (Insere imagem na documentação)
\section (Insere seção, para comentários)
\file (Insere nome do arquivo fonte principal)
\brief (Insere comentário referente ao último comando)
\author (Insere autor)
\version (Insere versão do sistema)
\date (Insere data de criação)
*/
```

A seguir, dois exemplos simples de código-fonte em linguagem C são apresentados. O primeiro exemplo, ver *Listing* 1, apresenta o excerto de código do arquivo `structcmd.h`. Já o segundo exemplo, ver *Listing* 2, apresenta a forma declarada da documentação do código nos padrões do DOXYGEN. Optamos por manter a originalidade do texto, para estes *Listings* que se seguem.

Listing 1: Código Base

```
1 #define MAX(a,b) (((a)>(b))?(a):(b))
2 typedef unsigned int UINT32;
3 int errno;
4 int open(const char *,int);
5 int close(int);
6 size_t write(int,const char *, size_t);
7 int read(int,char *,size_t);
```

Segue abaixo o mesmo código-fonte do *Listing* 1, mas obedecendo os padrões de comentários estabelecidos pelo DOXYGEN para se construir a documentação do código.

Listing 2: Código Base com comentários

```
1 /*! \mainpage Esta é uma aplicação teste.
2  \section Introdução simples e resumida de um programa.
3  \section Seção para explicar os procedimentos de instalação.
4  \subsection Passo 1: instalar.exe:  Apenas clique em instalar.
```

```
 5  \author Tiago Faria Bicalho
 6  \version 1.0.0.0
 7  \date 03/12/2009
 8  \bug Não existem Bugs
 9   \warning Aviso que esta aplicação está ok.
10 */
11
12 /*! \file structcmd.h
13  \brief A Documented file.
14 */
15
16 /*! \def MAX(a,b)
17  \brief A macro that returns the maximum of \a a and \a b.
18 */
19 #define MAX(a,b) (((a)>(b))?(a):(b))
20
21 /*! \var typedef unsigned int UINT32
22  \brief A type definition for a .
23 */
24 typedef unsigned int UINT32;
25
26 /*! \var int errno
27  \brief Contains the last error code.
28  \warning Not thread safe!
29 */
30 int errno;
31
32 /*! \fn int open(const char *pathname,int flags)
33  \brief Opens a file descriptor.
34  \param pathname The name of the descriptor.
35  \param flags Opening flags.
36 */
37 int open(const char *,int);
38
39 /*! \fn int close(int fd)
40  \brief Closes the file descriptor \a fd.
41  \param fd The descriptor to close.
42 */
43 int close(int);
44
45 /*! \fn size_t write(int fd,const char *buf, size_t count)
46  \brief Writes \a count bytes from \a buf to the filedescriptor \a fd.
```

```
47  \param fd The descriptor to write to.
48  \param buf The data buffer to write.
49  \param count The number of bytes to write.
50 */
51 size_t write(int,const char *, size_t);
52
53 /*! \fn int read(int fd,char *buf,size_t count)
54  \brief Read bytes from a file descriptor.
55  \param fd The descriptor to read from.
56  \param buf The buffer to read into.
57  \param count The number of bytes to read.
58 */
59 int read(int,char *,size_t);
```

Segue abaixo a explicação de cada comando acima utilizado:

- `\param` Define o parâmetro da função.
- `\brief` Define o comentário para o comando.
- `\fn` Define as funções.
- `\var` Define as variáveis.
- `\var typedef` Define as variáveis e qual módulo ela pertence.
- `\def` Define as variáveis ou funções do comando `#define`.
- `\file` Define o arquivo lido.

## 3.2 Ajustando a codificação do Arquivo Fonte

Por padrão, o DOXYGEN gera relatórios `.tex` na codificação UTF8, assim é importante atentar-se a codificação dos arquivos fonte. Caso estes não estejam em UTF8 estes devem ser convertidos utilizando algum editor específico.

## 3.3 Ajustando o Idioma

O DOXYGEN permite a documentação na língua portuguesa. Para isto basta selecionar na aba "Expert" em "Project" e encontrar a opção OUTPUT_LANGUAGE, selecione a língua portuguesa e então vá em "Run" para iniciar a documentação, o sistema irá gerar a documentação com êxito, mas irá advertí-lo de uma possível falha.

```
Warning: The selected output language "portuguese" has not been
updated since release 1.3.3. As a result some sentences may appear in
English.
```

Esta mensagem indica que a codificação para língua portuguesa no formato Brasil não é atualizada desde a versão 1.3.3 e que algumas palavras ainda podem aparecer em inglês. Contudo podemos contornar este problema. Após gerar a documentação em LaTeX2$\varepsilon$, abra o arquivo principal denominado `refman.tex`, localize:

```
\usepackage[utf8]{inputenc}
Portuguese
```

e substitua por

```
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[brazil]{babel}
```

# 4 Gerando Manuais de Referência em LaTeX2$\varepsilon$

A geração de um manual de referência em LaTeX2$\varepsilon$ é simples e pode acontecer em duas diferentes etapas, tais como a etapa *Wizard* e a etapa *Expert* - para ambas, ver Figura 1. A etapa *Wizard* é designada para usuários com pouca ou nenhuma experiência com Doxygen. Nesta etapa, o processo de extração de comentários já está previamente configurado. Na guia apresentada, Figura 1, únicos campos a serem preenchidos são o nome do sistema, versão, o caminho do diretório contendo os arquivos fonte e o caminho do diretório onde o Doxygen irá gerar a documentação.

Entretanto, a etapa *Expert* necessita de maiores conhecimentos para alterar algumas configurações, tais como codificação, tipo de papel e outras.

Para acompanhar a evolução do Doxygen, durante a leitura do código-fonte, você pode escolher a opção “Run” - ver Figura 2.

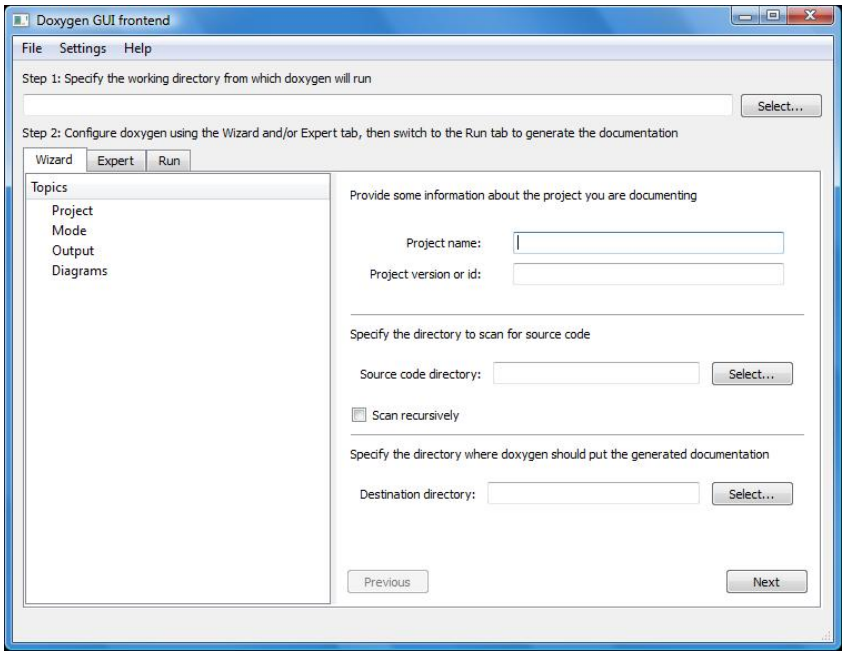



Figura 1: GUI para gerar relatórios.

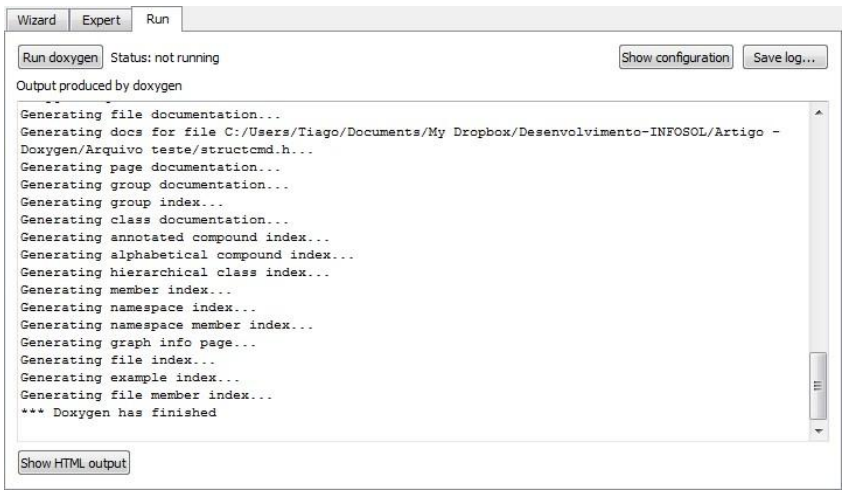



Figura 2: Leitura na geração de relatório.

# 5 Conclusão

O sucesso na documentação de um código-fonte proporciona uma melhoria no processo de desenvolvimento de software. Analistas de sistemas terão uma documentação precisa e fiel durante as atividades e/ou atualizações que foram realizadas no processo de implementação.

Ao documentar um código-fonte, utilizando as formatações do DOXYGEN, podemos perceber que o arquivo fonte tem a tendência de aumentar o número de linhas. Por outro lado, a documentação é embutida junto ao código e com apenas alguns cliques, esta documentação estará atualizada de acordo com as alterações no código além de não correr o risco de perdermos a documentação uma vez que esta faz parte do sistema, ou por incompatibilidade com editores de texto WYSIWYG (*What You See Is What You Get*).

# 6 Agradecimentos

Os autores agradecem aos professores Delaine Vasconcelos Rosa e José Geraldo Costa pela revisão gramatical final.

# Referências

[1] Reginaldo J. Santos. Introdução ao latex. *Departamento de Matemática-ICEx Universidade Federal de Minas Gerais*, page 68, 2002.

[2] Klaus Steding-Jessen. Latex: Uma alternativa mais eficiente comparada aos sistemas wysiwyg. 1998.

# APÊNDICE

# Programa Exemplo Doxygen e LaTeX

1.0

Gerado por Doxygen 1.6.1

Thu Dec 17 17:17:33 2009

# Sumário

# Capítulo 1

# Indice dos Arquivos

## 1.1 Lista de Arquivos

Lista de todos os ficheiros documentados com uma breve descrição:

# Capítulo 2

# Documentação do Arquivo

## 2.1 Referência ao Arquivo C:/Users/Tiago/Desktop/Teste/structcmd.h

A Documented file.

### Macros

- #define MAX(a, b) (((a)>(b))?(a):(b))

  *A macro that returns the maximum of a and b.*

### Definições de tipos

- typedef unsigned int UINT32

  *A type definition for a .*

### Funções

- int open (const char *, int)

  *Opens a file descriptor.*

- int close (int)

  *Closes the file descriptor fd.*

- size_t write (int, const char *, size_t)

  *Writes count bytes from buf to the filedescriptor fd.*

- int read (int, char *, size_t)

  *Read bytes from a file descriptor.*

## Variáveis

- int errno
  *Contains the last error code.*

### 2.1.1 Descrição detalhada

A Documented file. Details.

### 2.1.2 Documentação das macros

#### 2.1.2.1 #define MAX(a, b) (((a)>(b))?(a):(b))

A macro that returns the maximum of *a* and *b*. Details.

### 2.1.3 Documentação dos tipos

#### 2.1.3.1 typedef unsigned int UINT32

A type definition for a . Details.

### 2.1.4 Documentação das funções

#### 2.1.4.1 int close (int *fd*)

Closes the file descriptor *fd*.

**Parâmetros:**

***fd*** The descriptor to close.

#### 2.1.4.2 int open (const char * *pathname*, int *flags*)

Opens a file descriptor.

**Parâmetros:**

***pathname*** The name of the descriptor.
***flags*** Opening flags.

#### 2.1.4.3 int read (int *fd*, char * *buf*, size_t *count*)

Read bytes from a file descriptor.

**Parâmetros:**

***fd*** The descriptor to read from.
***buf*** The buffer to read into.
***count*** The number of bytes to read.

#### 2.1.4.4 size_t write (int *fd*, const char * *buf*, size_t *count*)

Writes *count* bytes from *buf* to the filedescriptor *fd*.

**Parâmetros:**

***fd*** The descriptor to write to.

***buf*** The data buffer to write.

***count*** The number of bytes to write.

### 2.1.5 Documentação das variáveis

#### 2.1.5.1 int errno

Contains the last error code.

**Aviso:**

Not thread safe!

THE PracTEX Journal
TEX USERS GROUP

# A student report template (Portuguese)

Francisco Reinaldo et al

Abstract

English

In this paper we present an example of a technical report commonly used by students to present their academic research.

Português

Neste artigo, nós apresentamos um exemplo de relatório técnico comumente utilizado pelos alunos, a fim de apresentar suas pesquisas acadêmicas.

Francisco Reinaldo é atualmente professor na Área de Exatas no UnilesteMG, Brasil, em específico, o curso de Computação - Sistemas de Informação. No UnilesteMG, sua área de pesquisa é Inteligência Computacional (Diretor do LIC), mas dedica boa parte de seu tempo com estudos envolvendo LaTeX.

Victor Vasconcelos Moreira cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Estagiário do Projeto Informática Solidária - INFOSOL e colaborador do Laboratório de Inteligência Computacional – LIC, ambos os projetos sustentados pelo UnilesteMG.

Maria Tereza de Castro Costa cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiária no Laboratório de Inteligência Computacional (LIC), trabalhando com Redes Neurais Artificiais.

Tiago Faria Bicalho cursa o 7º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiário no Laboratório de Inteligência Computacional (LIC) e colaborador no Projeto Informática Solidária - INFOSOL.

Os autores podem ser contatados através de reinaldo.opus@gmail.com

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# Projeto Interdisciplinar (PI) em LaTeX2ε: Um modelo de relatório para a academia

Francisco Reinaldo[1], Maria Tereza de Castro Costa[2], Tiago Faria Bicalho[3], and Victor Vasconcelos Moreira[4]

Email [1]reinaldo.opus@gmail.com, [2]maryxb@gmail.com, [3]tiagofariabicalho@gmail.com, [4]victorvasconcelosfox@gmail.com

Resumo In this paper we present an example of technical report commonly used by students in order to present their academic research.

## 1 Introdução

O Projeto Interdisciplinar, do Curso de Computação - Sistemas de Informação, do Centro Universitário do Leste de Minas Gerais - UnilesteMG, consiste no desenvolvimento de uma produção acadêmica teórico-prática no formato de relatório por grupos de três alunos orientados por professores ao longo do semestre.

Preferencialmente, seu conteúdo deve ser um estudo de caso de uma organização da comunidade. Além de integrar disciplinas, este projeto aproxima o aluno da realidade do mercado de trabalho em computação.

O uso do LaTeX2ε veio para agilizar e padronizar a construção dos relatórios desse projeto, isto porque a partir do momento que se cria um "padrão de relatório" em LaTeX2ε, o usuário não precisará se preocupar com a formatação de sua produção, pois o próprio LaTeX2ε já faz isto com os seus pacotes e códigos apropriados para o tipo do documento.

Para este artigo, focalizamos no processo de como preencher o relatório com comandos em LaTeX2ε para a construção do PI dos alunos do UnilesteMG.

# 2 Objetivos

Esse artigo apresenta o tipo de modelo de leiaute no formato LaTeX2$_\varepsilon$ de um relatório do PI utilizado no curso de Computação - Sistemas de Informação, para facilitar a construção dos mesmos pelos alunos.

# 3 Desenvolvimento

Neste artigo, estaremos exibindo um modelo de relatório de PI utilizado no Curso de Computação - Sistemas de Informação e também no Laboratório de Inteligência Computacional (LIC).

Este relatório de PI é composto por vários arquivos `.tex`, e cada um com seu respectivo propósito. Para cada arquivo necessário, uma subseção é apresentada. Também explicaremos os comandos utilizados para a construção dos mesmos e as áreas que deverão ser alteradas para a customização do relatório de acordo com o PI de cada aluno.

## 3.1 O arquivo `principal.tex`

Para o desenvolvimento do proposto relatório, nós iniciamos com um arquivo principal chamado de `principal.tex`, Figura 1, que une todos os outros arquivos e é a origem de todo o relatório. Dentro do `principal.tex` encontramos os seguintes códigos:

```
1  \documentclass[12pt]{article}
2  \usepackage[final]{pdfpages}
3  \usepackage{preambulo}
4  \begin{document}
5  \thispagestyle{empty}
6  \begin{center}
7      \Instituicao{Centro Universitário do Leste de Minas Gerais - UnilesteMG}
8      \Curso{Curso de Computação - Sistemas de Informação}
9      \ProjetoPI{Relatório do Projeto Interdisciplinar (PI) / Pré-TCC}
10     \TituloProjeto{RoboCup Rescue}
11     \Alunos{Tiago Faria Bicalho}
12     \Periodo{6}
13     \Orientador{Francisco Reinaldo}
14     \LocalData
15 \end{center}
16 \sumarios
17 \onehalfspacing
18 \input{introducao.tex}
19 \input{hipoteses.tex}
20 \input{objetivos.tex}
21 \input{justificativa.tex}
22 \input{revisaoliteratura.tex}
23 \input{desenvolvimento.tex}
24 \input{material.tex}
25 \input{resultados.tex}
26 \input{consideracoesfinais.tex}
27 \pagebreak
28 \bibliographystyle{plain}
29 \bibliography{rescuebib}
30 \end{document}
```

Figura 1: Arquivo exemplo `principal.tex`.

Para customizar o relatório de acordo com o seu projeto, altere os seguintes campos do exemplo acima:

- Na linha 7, altere o texto que está entre `{ }` e insira o nome da Instituição.
- Na linha 8, altere o texto que está entre `{ }` e insira o nome do Curso.
- Na linha 9, altere o texto que está entre `{ }` e insira o nome do Relatório.
- Na linha 10, altere o texto que está entre `{ }` e insira o Título do Projeto.
- Na linha 11, altere o texto que está entre `{ }` e insira o(s) Nome(s) do(s) Aluno(s).
- Na linha 12, altere o texto que está entre `{ }` e insira o Período.

– Na linha 13, altere o texto que está entre `{ }` e insira o nome do Professor Orientador do projeto.

Explicando o código contido na Figura 1, temos:

– A linha 1, `\documentclass[12pt]{article}`, define o tipo do documento `{article}` e o tamanho 12 do texto no documento.

– A linha 2, `\usepackage[final]{pdfpages}`, adiciona um pacote para inserção de PDF'S no documento quando se necessário.

– A linha 3, `\usepackage{preambulo}`, define o pacote que estamos utilizando para este documento (preambulo).

– A linha 4, `\begin{document}`, marca o início do documento.

– A linha 5, `\thispagestyle{empty}`, retira a numeração da página.

– A linha 6, `\begin{center}`, inicia a centralização do texto.

– A linha 7, `\Instituicao{Centro Universitário do Leste de Minas Gerais - UnilesteMG}`, adiciona o nome da Instituição ao documento. O nome da Instituição estará entre `{ }`.

– A linha 8, `\Curso{Curso de Computação - Sistemas de Informação}`, adiciona o nome do Curso ao documento. O nome do Curso estará entre `{ }`.

– A linha 9, `\ProjetoPI{Relatório do Projeto Interdisciplinar (PI) / Pré-TCC }`, adiciona o nome do Relatório ao documento. O nome do Relatório estará entre `{ }`.

– A linha 10, `\TituloProjeto{Robocup Rescue}`, adiciona o Título do Projeto ao documento. O Título do Projeto estará entre `{ }`.

– A linha 11, `\Alunos{Tiago Faria Bicalho}`, adiciona o(s) Nome(s) do(s) Aluno(s) ao documento. O Nome do Aluno estará entre `{ }`.

– A linha 12, `\Periodo{6}`, adiciona o Período ao documento. O Período estará entre `{ }`.

– A linha 13, `\Orientador{Francisco Reinaldo}`, adiciona o nome do Professor Orientador ao documento. O nome do Professor Orientador estará entre `{ }`.

- A linha 14, `\LocalData`, adiciona a Data ao documento. Caso deseje alterar a data, localize-a no arquivo `preambulo.sty`. Para data corrente a geração do PDF, use `\today`.
- A linha 15, `\end{center}`, finaliza a centralização do texto.
- A linha 16, `\sumarios`, adiciona um Sumário ao documento.
- A linha 17, `\onehalfspacing`, implementa o espaçamento de um e meio no documento.
- A linha 18, `\input`, inclui um arquivo no documento.
- A linha 27, `\pagebreak` faz uma quebra de página.
- A linha 28, `\bibliographystyle{plain}`, define o estilo de bibliografia. O nome do estilo de bibliografia estará entre `{ }`.
- A linha 29, `\bibliography{rescuebib}`, insere a bibliografia. O nome da bibliografia estará entre `{ }`.
- A linha 30, `\end{document}`, finaliza o documento.

Quando convertermos o documento da Figura 1 para PDF, temos a versão de apresentação da capa e do sumário nas próximas duas páginas que se seguem:

**Centro Universitário do Leste de Minas Gerais - UnilesteMG**

**Curso de Computação - Sistemas de Informação**

**Relatório do Projeto Interdisciplinar (PI) / Pré-TCC**

# RoboCup Rescue

Aluno(s): **Tiago Faria Bicalho**

Período: **6**

Professor Orientador: **Francisco Reinaldo**

Cel. Fabriciano - MG

10 de outubro de 2009

# Sumário

## 3.2 O arquivo `introducao.tex`

O arquivo `introducao.tex`, Figura 2, contém os seguintes códigos:

```
1 \section{Introdução}
2
3 A área de Sistemas Multiagentes (SMA) é um ramo de pesquisa da Inteligência Artificial (IA)  dedicada a busca
  por métodos ou dispositivos computacionais que possuam ou simulem a capacidade humana de resolver problemas de
  forma colaborativa, pensar ou, de forma ampla, ser inteligente. SMA são sistemas compostos por múltiplos
  elementos computacionais interativos denominados agentes.  Agentes são entidades computacionais com duas
  habilidades fundamentais tais como, decidir por si próprio o que devem fazer para satisfazer seus objetivos de
  projeto e interagir com outros agentes de forma social \cite{Wooldridge2002}.
4
5 Os agentes em um sistema SMA comumente são apresentados sob a forma virtual em um ambiente de simulação.
  RoboCupRescue é um projeto voltado a competição de equipes heterogêneas de agentes homogêneos. Este projeto
  promove o desenvolvimento de soluções voltadas para a busca e salvamento em cenários de catástrofe, que tem
  como objetivo desenvolver agentes inteligentes e robôs capazes de interagir e atuar em cenários simulados de
  desastre, situações de calamidade possuem grande relevância social. O termo calamidade provém do latim
  calamidade, significando desgraça pública, catástrofe, flagelo \cite{Ferreira1999}. Tais situações de
  catástrofe exigem ações rápidas e inteligentes, no intuito de minimizar seus efeitos, envolvendo equipes de
  resgate e salvamento compostas por seres humanos, que são expostos a estes ambientes hostis e, portanto corre
  risco de morte.
6
7 \endinput
```

Figura 2: Arquivo exemplo `introducao.tex`.

Explicando o código contido na Figura 2, temos:

- A linha 1, `\section{}`, divide o documento em Seções. O nome da Seção estará entre `{ }`.
- A linha 3, `\cite{}`, cria uma marca para citar um livro ou artigo descrito na bibliografia. O nome da Citação estará entre `{ }`.
- A linha 7, `\endinput`, marca o fim da inclusão de um arquivo no documento.

Para customizar o relatório, altere o texto do exemplo acima de acordo com a introdução do seu projeto.

## 3.3 O arquivo `hipoteses.tex`

O arquivo `hipoteses.tex`, Figura 3, contém os seguintes códigos:

```
\section{Hipóteses}

Acredita-se que esta pesquisa é uma área de grande aplicação de sistemas artificiais, a qual poderá aprimorar
ou solucionar problemas diversos, entre eles falhas em comunicação e melhorias em estratégias.

Afim de validar o projeto, participaremos de eventos cuja finalidade é promover o desenvolvimento de módulos de
inteligência artificial agregados a simuladores onde após a participação poderemos avaliar o modelo e melhorá-
lo, se for o caso.


\endinput
```

Figura 3: Arquivo exemplo `hipoteses.tex`.

Para customizar o relatório, altere o texto do exemplo acima de acordo com as hipóteses do seu projeto.

Ao converter os documentos acima (`introducao.tex` e `hipoteses.tex`) para PDF, temos a versão de apresentação mostrada na próxima pagina:

# 1 Introdução

A área de Sistemas Multiagentes (SMA) é um ramo de pesquisa da Inteligência Artificial (IA) dedicada a busca por métodos ou dispositivos computacionais que possuam ou simulem a capacidade humana de resolver problemas de forma colaborativa, pensar ou, de forma ampla, ser inteligente. SMA são sistemas compostos por múltiplos elementos computacionais interativos denominados agentes. Agentes são entidades computacionais com duas habilidades fundamentais tais como, decidir por si próprio o que devem fazer para satisfazer seus objetivos de projeto e interagir com outros agentes de forma social [4].

Os agentes em um sistema SMA comumente são apresentados sob a forma virtual em um ambiente de simulação. RoboCupRescue é um projeto voltado a competição de equipes heterogêneas de agentes homogêneos. Este projeto promove o desenvolvimento de soluções voltadas para a busca e salvamento em cenários de catástrofe, que tem como objetivo desenvolver agentes inteligentes e robôs capazes de interagir e atuar em cenários simulados de desastre, situações de calamidade possuem grande relevância social. O termo calamidade provém do latim calamidade, significando desgraça pública, catástrofe, flagelo [2]. Tais situações de catástrofe exigem ações rápidas e inteligentes, no intuito de minimizar seus efeitos, envolvendo equipes de resgate e salvamento compostas por seres humanos, que são expostos a estes ambientes hostis e, portanto corre risco de morte.

# 2 Hipóteses

Acredita-se que esta pesquisa é uma área de grande aplicação de sistemas artificiais, a qual poderá aprimorar ou solucionar problemas diversos, entre eles falhas em comunicação e melhorias em estratégias.

Afim de validar o projeto, participaremos de eventos cuja finalidade é promover o desenvolvimento de módulos de inteligência artificial agregados a simuladores onde após a participação poderemos avaliar o modelo e melhorá-lo, se for o caso.

## 3.4 O arquivo `objetivos.tex`

O arquivo `objetivos.tex`, Figura 4, contém os seguintes códigos:

```
1  \section{Objetivos}
2
3  \subsection{Gerais}
4
5  A Liga Rescue Simulation tem dois objetivos gerais. O primeiro visa o desenvolvimento de simuladores para
   formar a infra-estrutura de sistemas de simulação, e emular fenômenos reais predominantes em catástrofes. O
   segundo visa desenvolver agentes inteligentes capazes de interagir e atuar em cenários de desastre.
6
7  O objetivo primário desta pesquisa é construir uma ferramenta de aprendizagem para o diminuir o tempo de
   treinamento do agente; consequentemente atingir melhores desempenhos. Neste contexto, o projeto tem o propósito
   de adaptar e melhorar a arquitetura de aprendizagem RatoLIC, tornando-a robusta e com tomada de decisão
   autônoma, somado ao suporte para outros algoritmos de aprendizagem na elaboração de comportamentos do agente,
   assim resolvendo o problema manual da escolha da estratégia mais adequada.
8
9  \subsection{Específicos}
10
11 O objetivo do projeto visa desenvolver módulos de inteligência artificiais robustos capazes de se adaptar a
   qualquer ambiente.
12
13 Os objetivos secundários são:
14 \begin{itemize}
15     \item Escolha automática de estratégias utilizadas pelos agentes do RoboCupRescue;
16     \item Redução do tempo de treinamento desses agentes;
17     \item Redução da complexidade do problema;
18     \item Desenvolvimento de uma arquitetura genérica, robusta e com tomada de decisão autônoma para um agente
   específico, fazendo ele atuar de forma autônoma.
19 \end{itemize}
20
21 \endinput
```

Figura 4: Arquivo exemplo `objetivos.tex`.

Explicando o código contido na Figura 4, temos:

- A linha 3, `\subsection{}`, divide o documento em Subseções. O nome da Subseção estará entre `{ }`.
- A linha 14, `\begin{itemize}`, marca o inicio do ambiente de listas em um documento.
- A linha 15, `\item`, adiciona um hífen antes de uma frase ou palavra.
- A linha 19, `\end{itemize}`, marca o fim do ambiente de listas em um documento.

Para customizar o relatório, altere o texto do exemplo acima de acordo com o objetivo do seu projeto.

Quando convertermos o documento acima para PDF, temos a versão de apresentação mostrada na próxima pagina:

# 3 Objetivos

## 3.1 Gerais

A Liga Rescue Simulation tem dois objetivos gerais. O primeiro visa o desenvolvimento de simuladores para formar a infra-estrutura de sistemas de simulação, e emular fenômenos reais predominantes em catástrofes. O segundo visa desenvolver agentes inteligentes capazes de interagir e atuar em cenários de desastre.

O objetivo primário desta pesquisa é construir uma ferramenta de aprendizagem para o diminuir o tempo de treinamento do agente; consequentemente atingir melhores desempenhos. Neste contexto, o projeto tem o propósito de adaptar e melhorar a arquitetura de aprendizagem RatoLIC, tornando-a robusta e com tomada de decisão autônoma, somado ao suporte para outros algoritmos de aprendizagem na elaboração de comportamentos do agente, assim resolvendo o problema manual da escolha da estratégia mais adequada.

## 3.2 Específicos

O objetivo do projeto visa desenvolver módulos de inteligência artificiais robustos capazes de se adaptar a qualquer ambiente.

Os objetivos secundários são:

- Escolha automática de estratégias utilizadas pelos agentes do RoboCupRescue;
- Redução do tempo de treinamento desses agentes;
- Redução da complexidade do problema;
- Desenvolvimento de uma arquitetura genérica, robusta e com tomada de decisão autônoma para um agente específico, fazendo ele atuar de forma autônoma.

## 3.5 O arquivo `justificativa.tex`

O arquivo `justificativa.tex`, Figura 5, contém os seguintes códigos:

```
\section{Justificativa}

O fato de buscas e salvamentos realizados em ambientes hostis por equipes de seres humanos incorrer em risco de
morte a estes, e do mesmo modo, a necessidade de se desenvolver tecnologias que auxiliem ou mesmo executem toda
a rotina de atendimento com autonomia, introduz uma série de temas avançados e interdisciplinares, tais como
IA, comportamento estratégico(planejamento Multiagente, planejamento em tempo real, agentes heterogêneos).

\endinput
```

Figura 5: Arquivo exemplo ‘`justificativa.tex`.

Para customizar o relatório, altere o texto do exemplo acima de acordo com a justificativa do seu projeto.

## 3.6 O arquivo `revisaoliteratura.tex`

Logo após criaremos o arquivo `revisaoliteratura.tex`, Figura 6 e 7, onde encontramos os seguintes códigos:

```
\section{Revisão de Literatura}

O projeto RoboCup Rescue surgiu como idéia após o terremoto ocorrido em 17 de Janeiro de 1995 na cidade de
Kobe, no Japão, onde cerca de 6500 pessoas morreram, 80.000 casas foram destruídas e aproximadamente 1 milhão
de pessoas foram afetadas. Com isso, o RoboCup Rescue Simulation Project visa promover a pesquisa e
desenvolvimento envolvendo trabalho coordenado multiagente. A meta da competição é conseguir salvar o maior
número de vítimas de um desastre virtual simulado, através da ação em conjunto de 3 tipos de agentes
(bombeiros, ambulâncias e policiais). Existem muitas restrições tais como número de mensagens limitado a
base, carga de água de cada caminhão bombeiro e eventos como massas de ar se deslocando espalhando o fogo
além da ocorrência de novos tremores que podem mudar o cenário, isto deixa a simulação ainda mais real e
desafiadora, e com isso, exigindo um maior esforço dos participantes, o que acaba contribuindo ainda mais com
o desenvolvimento de técnicas de inteligência artificial, robótica e sistemas multiagente
\cite{RoboCupRescueHome2009}.

O ambiente RoboCup Rescue possibilita por sua complexidade, a utilização de suas ferramentas para um estudo
especifico e aprofundado. O simulador de incêndios abre portas para aplicações diretas em diversos estudos
entre eles o comportamento coletivo em relação a casos onde o pânico em grandes multidões toma conta da
situação, o simulador após uma reconfiguração dos agentes civis é capaz de exibir com certo grau de perfeição
o comportamento de uma multidão apavorada, que pode orientar pesquisadores a encontrar a melhores rotas de
fuga em caso de incêndio. Assim podemos definir que este é o projeto que desencadeará uma forma completamente
nova de se estudar o comportamento humano, e através dele podemos resolver os diversos problemas encontrados
nas simulações realizadas com base em dados já existentes.

O termo ``agente'', ou software agente, tem sido utilizado em uma série de tecnologias, por exemplo, em
inteligência artificial, bancos de dados, sistemas operacionais e redes de computadores.  Embora não exista
um definição única (RUSSEL; NORVIG, 2003), todas as definições concordam que:``Um agente é um componente de
software especial que possui autonomia e que provê uma interface para um sistema arbitrário e/ou comporta-se
como um agente humano.''\cite{Bellifemine2007}. Os agentes são compostos por características próprias onde
estas definem o agente sendo ele bombeiro, polícia, ambulância ou civil. O bombeiro tem como característica
relevante a quantidade de água no tanque, já a polícia os sensores para percepção dos materiais diversos que
fazem a obstrução nas ruas, onde a sua função é exatamente retirá-los para que estes não atrapalhem os demais
agentes a realizarem suas tarefas, a ambulância tem como característica o sensor auditivo, esta é capaz de
ouvir os gritos dos agentes civis e assim encontrá-los para um futuro salvamento, ver
Figura~\ref{fig:imagem1}.
\begin{figure}[ht]
    \centering
    \includegraphics[width=0.96\textwidth]{imagens/viewer.jpg}
    \caption{Visualizador.}\label{fig:imagem1}
\end{figure}

Os agentes humanóides, também chamados de agentes locais, têm a capacidade de atuar de modo direto sobre o
ambiente, e podem ser do tipo:  civil, bombeiro, polícia e ambulância.  Qualquer humanóide tem a capacidade
de se comunicar e movimentar no espaço geográfico da simulação, cada agente possui capacidades específicas.
O agente civil representa uma família ou indivíduo vítima da catástrofe, que eventualmente precisará ser
socorrida (pelo agente ambulância). O agente bombeiro tem a capacidade de extinguir incéndios em edifícios,
manipulando as mangueiras para definir o ângulo de saída e a quantidade de água. O agente ambulância
representa a capacidade de salvamento de agentes humanóides (inclusive outras ambulâncias).  Apenas o agente
ambulância tem a capacidade de socorrer qualquer humanóide. Um  agente  não  humanóide,  também  chamado  de
agente  global,  representa
uma organização de agentes humanóides. Essas organizações são o quartel  de bombeiros e a central  de
ambulâncias (dentre outros).  Os agentes não humanóides  são  representados  por  construções  e  não  atuam
diretamente  no  ambiente, suas ações concretizam-se através dos agentes humanóides de que são compostos. O
quartel  de  bombeiros coleta e integra toda a informação enviada pelos agentes bombeiro e os aloca de acordo
com uma política simples assim como a esquadrão de  polícias o faz.  A central  de  ambulâncias coleta e
integra toda a informação enviada pelos agentes Ambulância e os aloca de acordo com uma política simples,
assim como todos os agentes não humanóides os fazem.

Atualmente o simulador possui módulos compilados na linguagem  C e Java. Estes se conectam ao kernel, ver
Figura~\ref{fig:imagem2}, que é responsável por controlar o ambiente de simulação.

\begin{figure}[ht]
    \centering
    \includegraphics[width=0.96\textwidth]{imagens/kernel.jpg}  \caption{Kernel.}\label{fig:imagem2}
\end{figure}
```

Figura 6: Parte 1/2 do arquivo `revisaoliteratura.tex`.

```
23
24 É através deste que poderemos inserir novos agentes, novos visualizadores, e alterar padrões de sensibilidade
   dos agentes, cada um dos módulos é responsável por uma atividade, estas  atividades são: Massa de ar em
   movimento que podem espalhar o fogo para outras edificações, trafego de veículos e pessoas pelas vias,
   comportamento de cada uma das equipes e civis.
25
26 As regras do campeonato RoboCupRescue consistem basicamente na versão do simulador a ser utilizado,
   geralmente entregue pela organização da competição 4 meses antes da mesma, a quantidade de agentes em cada
   equipe, pontos de ignição, torres de comando de cada equipe além de refúgios, como segue tabela 1 abaixo.
27
28
29 \begin{table}[h]
30   \begin{center}
31     \caption{Configuração dos agentes.}
32     \begin{tabular}{|c||c||c|} \hline
33       Entidade                        &  Mínimo & Máximo \\
34       \hline
35       Bombeiros                       &     0   &   15  \\
36       Polícia                         &     0   &   15  \\
37       Ambulância                      &     0   &    8  \\
38       Civil                           &    70   &   90  \\
39       Torre Central dos Bombeiros     &     0   &    1  \\
40       Torre Central da Polícia        &     0   &    1  \\
41       Torre Central das Ambulâncias   &     0   &    1  \\
42       Refúgios                        &     0   &    5  \\
43       Pontos de Ignição               &     2   &    8  \\
44       \hline
45     \end{tabular}
46   \end{center}
47 \end{table}
48
49
50 \endinput
```

Figura 7: Parte 2/2 do arquivo `revisaoliteratura.tex`.

Explicando o código contido nas Figuras 6 e 7, temos:

- A linha 7, `\ref{}`, cria uma referência cruzada em um documento. O nome da Referência estará entre `{ }`.
- A linha 8, `\begin{figure}[]`, cria um ambiente para inserir uma imagem em um documento. Qualquer coisa que seja incluída num ambiente *figure* ou *table* será tratado como matéria flutuante. Ambos ambientes flutuantes proporcionam um parâmetro opcional cujo os parâmetros de posicionamento estão entre `[ ]`. O designador de colocação [!hbp] permite ao LaTeX2$\varepsilon$ posicionar a tabela ou imagem justamente aqui (h) ou ao final (b) de alguma página ou em alguma página especial para elementos flutuantes (p), e em qualquer parte senão ficar bem (!). Se não fornecer nenhum designador de posição, então as classes normalizadas assumem [hbp].
- A linha 9, `\centering`, centraliza a imagem.
- A linha 10, `\includegraphics[]{}`, inclui uma imagem em seu documento,

cujo o tamanho da imagem é definido entre [ ] e o nome do arquivo estará entre `{ }`.

- A linha 11, `\caption{}`, define a legenda de um objeto. O nome da Legenda estará entre `{ }`. `\label{}`, cria uma marca para a referência cruzada em um documento. O nome da Marca estará entre `{ }`.
- A linha 12,`\end{figure}`, marca o fim do ambiente de inserção de imagens.
- A linha 29, `\begin{table}[]`, cria um ambiente para inserir uma tabela em um documento. Os procedimentos de posicionamento de uma tabela no LaTeX2$_\varepsilon$ são iguais aos de inserção de figura como vimos anteriormente.
- A linha 30, `\begin{center}`, cria um ambiente para centralizar objetos.
- A linha 32, `\begin{tabular}{}`, cria uma tabela cuja quantidade de colunas e a largura das mesmas são definidas entre `{ }`. `\hline`, cria uma linha horizontal. Para facilitar o processo de criação de uma tabela, utilizaremos o software *LaTable* cuja especialidade é criar tabelas para serem inseridas em um documento em LaTeX2$_\varepsilon$ .
- A linha 33, `&`, cria uma divisão entre as células das colunas. `\\`, salta uma linha.
- A linha 45, `\end{tabular}`, marca o fim da tabela.
- A linha 46, `\end{center}`, marca o fim do ambiente de centralização de objetos.
- A linha 47, `\end{table}`, marca o fim do ambiente da tabela.

Para customizar o relatório, altere o texto do exemplo acima de acordo com a revisão de literatura do seu projeto.

## 3.7 O arquivo `desenvolvimento.tex`

Logo após criaremos o arquivo `desenvolvimento.tex`, Figura 8, onde encontramos os seguintes códigos:

```
1  \section{Desenvolvimento}
2  Atualmente o simulador utilizado na competição se encontra na versão 1.0, este pode ser encontrado no site
   \begin{verbatim}http://sourceforge.net/projects/roborescue/\end{verbatim} gratuitamente, o aplicativo é
   desenvolvido na linguagem Java e contém em seu corpo estrutural funções em C as quais são reflexos de versões
   passadas totalmente programadas em C.
3
4  O simulador é capaz de inicializar todos os agentes em sua versão básica, mas estes não receberam nenhum
   núcleo  de inteligência artificial para que estes possam executar de maneira satisfatória a atividade a ele
   referida.
5  O desafio do Laboratório de Inteligência Computacional (LIC)  do UnilesteMG é realizar o desenvolvimento e
   estudo de um núcleo de inteligência artificial capaz de superar os limites já estabelecidos em outras
   competições.
6  O simulador obteve bons resultados durante a execução da aplicação grafica apenas em dois sistemas
   operacionais da distribuição Linux, estes são Kurumin Linux 8.0 e Ubuntu.
7  Informações sobre a instalação e execução do mesmo são de grande valor, assim o acesso a estas informações é
   restrito uma vez que se trata de uma competição mundial.
8
9  A instalação procede da seguinte maneira.
10 Abra o shell do Linux, precisaremos utilizar o apt-get para baixar alguns pacotes da internet, para que o
   apt-get possa funcionar no Unileste você precisa passar seu usuário,senha,url do proxy e porta a conectar.
11
12 Para isso primeiramente logue como root, então abra o shell e digite o comando:\begin{verbatim}
13 export http_proxy="http://AO6usuario-da-rede:senha-da-rede@10.2.0.1:3128",\end{verbatim} caso essa
   configuração não funcione, você pode tentar também escrever a seguinte linha no arquivo
   \begin{verbatim}\verb!/etc/apt/apt.conf:\end{verbatim}
14 \begin{verbatim}
15 Acquire{
16 HTTP::proxy "http://USusuario-da-rede:senha-da-rede@10.2.0.1:3128";
17 }
18 \end{verbatim}
19 Então poderemos finalmente atualizar a lista do apt-get utilizando o seguinte comando.
20 \begin{verbatim}apt-get update\end{verbatim}
21 Precisaremos instalar o Terminal X para que o simulador funcione corretamente, sendo que toda a programação é
   voltada a este tipo de terminal sendo impossível a execução em outro terminal sem que se altere a programação
   original da versão em questão
22 Para isso digite o seguinte comando.
23 \begin{verbatim}apt-get install xterm\end{verbatim}
24 Durante alguns minutos o sistema operacional irá acessar a internet a procura do pacote de instalação do
   Terminal X, ao encontrar este executará automaticamente a instalação, após o termino digite
25 xterm
26 Isso fará com que se abra outro shell de comando, feche o outro e utilizaremos a partir deste momento apenas
   o Terminal X, para dar continuidade ao processo de preparação do sistema para a execução do simulador
   precisaremos instalar um outro pacote denominado ANT, para isso digite:
27 \begin{verbatim}apt-get install ant\end{verbatim}
28 Realizando esta atividade o Java6 será automaticamente incorporado ao sistema junto ao pacote ANT, após esta
   instalação acesse a pasta boot dentro da pasta descompactada da versão 1.0 do simulador.
29 Com o terminal X apontando internamente a pasta boot digite:
30 \begin{verbatim}demo.sh\end{verbatim}
31 Uma apresentação básica de funcionamento do simulador ocorrerá, note que este não tem inteligência artificial
   e os agentes executam as atividades de forma desorganizada.
32
33 \endinput
```

Figura 8: Arquivo exemplo `desenvolvimento.tex`.

Explicando o código contido na Figura 8, temos:

– A linha 2, `\begin{verbatim}`, marca o início do ambiente onde os textos escritos nesse ambiente serão passados diretamente para o ficheiro de resultado, como se o tivesse escrito numa máquina de escrever, com todas as quebras de linha e espaços, sem que qualquer comando LaTeX2$_\varepsilon$ seja executado. Assim, `\end{verbatim}`, marca o fim do ambiente citado anteriormente.

– A linha 13, `\verb!`, escreve o texto exatamente como foi digitado, incluindo espaços e caracteres especiais.

Para customizar o relatório, altere o texto do exemplo acima de acordo com o desenvolvimento do seu projeto.

## 3.8 O arquivo `material.tex`

Logo após criaremos o arquivo `material.tex`, Figura 9, onde encontramos os seguintes códigos:

```
1  \subsection{Material e Métodos}
2
3  O simulador foi instalado nas seguintes condições nas quais obtivemos sucesso.\\
4  Sistema Operacional: Kurumin Linux 8.0 ou Ubuntu\\
5  Processador: Core 2 Duo 2.2\\
6  Memória Ram: 1GB\\
7  HD: 20GB's\\
8  Monitor de 17''\\
9  Conexão com Internet\\
10 \pagebreak
```

Figura 9: Arquivo exemplo `material.tex`.

Para customizar o relatório, altere o texto do exemplo acima de acordo com o material utilizado no seu projeto.

Ao converter os documentos acima (`justificativa.tex`, `revisaoliteratura.tex` e `desenvolvimento.tex`) para PDF, temos a versão de apresentação mostrada nas seis páginas que se seguem neste artigo:

# 4 Justificativa

O fato de buscas e salvamentos realizados em ambientes hostis por equipes de seres humanos incorrer em risco de morte a estes, e do mesmo modo, a necessidade de se desenvolver tecnologias que auxiliem ou mesmo executem toda a rotina de atendimento com autonomia, introduz uma série de temas avançados e interdisciplinares, tais como IA, comportamento estratégico(planejamento Multiagente, planejamento em tempo real, agentes heterogêneos).

# 5 Revisão de Literatura

O projeto RoboCup Rescue surgiu como idéia após o terremoto ocorrido em 17 de Janeiro de 1995 na cidade de Kobe, no Japão, onde cerca de 6500 pessoas morreram, 80.000 casas foram destruídas e aproximadamente 1 milhão de pessoas foram afetadas. Com isso, o RoboCup Rescue Simulation Project visa promover a pesquisa e desenvolvimento envolvendo trabalho coordenado multiagente. A meta da competição é conseguir salvar o maior número de vítimas de um desastre virtual simulado, através da ação em conjunto de 3 tipos de agentes (bombeiros, ambulâncias e policiais). Existem muitas restrições tais como número de mensagens limitado a base, carga de água de cada caminhão bombeiro e eventos como massas de ar se deslocando espalhando o fogo além da ocorrência de novos tremores que podem mudar o cenário, isto deixa a simulação ainda mais real e desafiadora, e com isso, exigindo um maior esforço dos participantes, o que acaba contribuindo ainda mais com o desenvolvimento de técnicas de inteligência artificial, robótica e sistemas multiagente [3].

O ambiente RoboCup Rescue possibilita por sua complexidade, a utilização de suas ferramentas para um estudo especifico e aprofundado. O simulador de incêndios abre portas para aplicações diretas em diversos estudos entre eles o comportamento coletivo em relação a casos onde o pânico em grandes multidões toma conta da situação, o simulador após uma reconfiguração dos agentes civis é capaz de exibir com certo grau de perfeição o comportamento de uma multidão apavorada, que pode orientar pesquisadores a encontrar a melhores rotas de fuga em caso de incêndio. Assim podemos definir que este é o projeto que desencadeará uma forma completamente nova de se estudar o comportamento humano, e através dele podemos resolver os diversos problemas encontrados nas simulações realizadas com base em dados já existentes.

O termo “agente”, ou software agente, tem sido utilizado em uma série de tecnologias, por exemplo, em inteligência artificial, bancos de dados, sistemas operacionais e redes de computadores. Embora não exista um definição única (RUSSEL; NORVIG, 2003), todas as definições concordam que:“Um agente é um componente de software especial que possui autonomia e que provê uma interface para um sistema arbitrário e/ou comporta-se como um agente humano.”[1]. Os agentes são compostos por características próprias onde estas definem o agente sendo ele bombeiro, polícia, ambulância ou civil. O bombeiro tem como característica relevante a quantidade de água no tanque, já a polícia os sensores para percepção dos materiais diversos que fazem a obstrução nas ruas, onde a sua função é exatamente retirá-los para que estes não atrapalhem os demais agentes a realizarem suas tarefas, a ambulância tem como característica o sensor auditivo, esta é capaz de ouvir os gritos dos agentes civis e assim encontrá-los para um futuro salvamento, ver Figura 1.


Figura 1: Visualizador.

Os agentes humanóides, também chamados de agentes locais, têm a capacidade de atuar de modo direto sobre o ambiente, e podem ser do tipo: civil, bombeiro, polícia e ambulância. Qualquer humanóide tem a capacidade de se comunicar e movimentar no espaço geográfico da simulação, cada agente possui capacidades específicas. O agente civil representa uma família ou indivíduo vítima da catástrofe, que eventualmente precisará ser socorrida (pelo agente ambulância). O agente bombeiro tem a ca-

pacidade de extinguir incêndios em edifícios, manipulando as mangueiras para definir o ângulo de saída e a quantidade de água. O agente ambulância representa a capacidade de salvamento de agentes humanóides (inclusive outras ambulâncias). Apenas o agente ambulância tem a capacidade de socorrer qualquer humanóide. Um agente não humanóide, também chamado de agente global, representa uma organização de agentes humanóides. Essas organizações são o quartel de bombeiros e a central de ambulâncias (dentre outros). Os agentes não humanóides são representados por construções e não atuam diretamente no ambiente, suas ações concretizam-se através dos agentes humanóides de que são compostos. O quartel de bombeiros coleta e integra toda a informação enviada pelos agentes bombeiro e os aloca de acordo com uma política simples assim como a esquadrão de polícias o faz. A central de ambulâncias coleta e integra toda a informação enviada pelos agentes Ambulância e os aloca de acordo com uma política simples, assim como todos os agentes não humanóides os fazem.

Atualmente o simulador possui módulos compilados na linguagem C e Java. Estes se conectam ao kernel, ver Figura 2, que é responsável por controlar o ambiente de simulação.

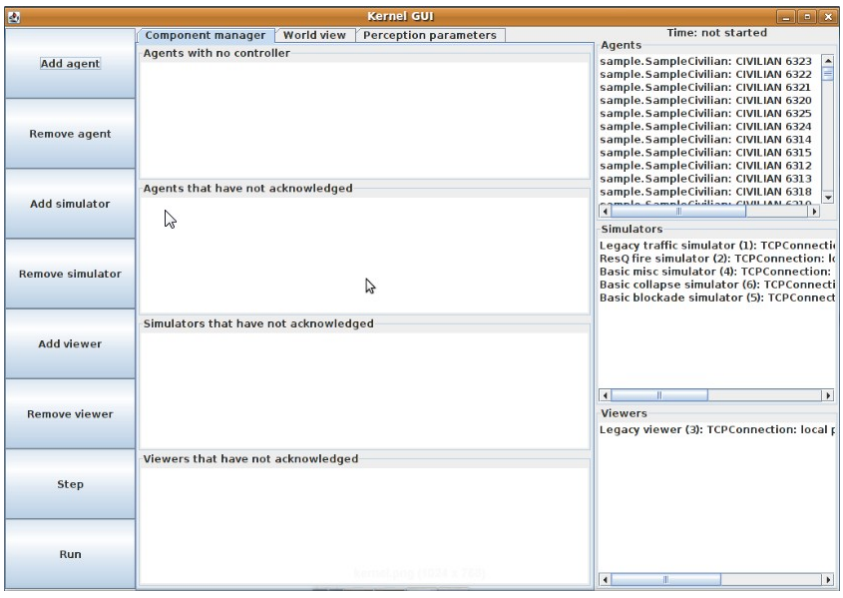



Figura 2: Kernel.

É através deste que poderemos inserir novos agentes, novos visualizadores, e alterar

padrões de sensibilidade dos agentes, cada um dos módulos é responsável por uma atividade, estas atividades são: Massa de ar em movimento que podem espalhar o fogo para outras edificações, trafego de veículos e pessoas pelas vias, comportamento de cada uma das equipes e civis.

As regras do campeonato RoboCupRescue consistem basicamente na versão do simulador a ser utilizado, geralmente entregue pela organização da competição 4 meses antes da mesma, a quantidade de agentes em cada equipe, pontos de ignição, torres de comando de cada equipe além de refúgios, como segue tabela 1 abaixo.

Tabela 1: Configuração dos agentes.

| Entidade | Mínimo | Máximo |
|---|---|---|
| Bombeiros | 0 | 15 |
| Polícia | 0 | 15 |
| Ambulância | 0 | 8 |
| Civil | 70 | 90 |
| Torre Central dos Bombeiros | 0 | 1 |
| Torre Central da Polícia | 0 | 1 |
| Torre Central das Ambulâncias | 0 | 1 |
| Refúgios | 0 | 5 |
| Pontos de Ignição | 2 | 8 |

# 6 Desenvolvimento

Atualmente o simulador utilizado na competição se encontra na versão 1.0, este pode ser encontrado no site

`http://sourceforge.net/projects/roborescue/`

O simulador é capaz de inicializar todos os agentes em sua versão básica, mas estes não receberam nenhum núcleo de inteligência artificial para que estes possam executar de maneira satisfatória a atividade a ele referida. O desafio do Laboratório de Inteligência Computacional (LIC) do UnilesteMG é realizar o desenvolvimento e estudo de um núcleo de inteligência artificial capaz de superar os limites já estabelecidos em outras competições. O simulador obteve bons resultados durante a execução da aplicação grafica apenas em dois sistemas operacionais da distribuição Linux, estes são Kurumin Linux 8.0 e Ubuntu. Informações sobre a instalação e execução do mesmo

são de grande valor, assim o acesso a estas informações é restrito uma vez que se trata de uma competição mundial.

A instalação procede da seguinte maneira. Abra o shell do Linux, precisaremos utilizar o apt-get para baixar alguns pacotes da internet, para que o apt-get possa funcionar no Unileste você precisa passar seu usuário,senha,url do proxy e porta a conectar.

Para isso primeiramente logue como root, então abra o shell e digite o comando:

```
export http_proxy="http://A06usuario-da-rede:senha-da-rede@10.2.0.1:3128",
```

```
Acquire{
HTTP::proxy "http://USusuario-da-rede:senha-da-rede@10.2.0.1:3128";
}
```

Então poderemos finalmente atualizar a lista do apt-get utilizando o seguinte comando.

```
apt-get update
```

Precisaremos instalar o Terminal X para que o simulador funcione corretamente, sendo que toda a programação é voltada a este tipo de terminal sendo impossível a execução em outro terminal sem que se altere a programação original da versão em questão Para isso digite o seguinte comando.

```
apt-get install xterm
```

Durante alguns minutos o sistema operacional irá acessar a internet a procura do pacote de instalação do Terminal X, ao encontrar este executará automaticamente a instalação, após o termino digite xterm Isso fará com que se abra outro shell de comando, feche o outro e utilizaremos a partir deste momento apenas o Terminal X, para dar continuidade ao processo de preparação do sistema para a execução do simulador precisaremos instalar um outro pacote denominado ANT, para isso digite:

```
apt-get install ant
```

Realizando esta atividade o Java6 será automaticamente incorporado ao sistema junto ao pacote ANT, após esta instalação acesse a pasta boot dentro da pasta descompactada da versão 1.0 do simulador. Com o terminal X apontando internamente a pasta boot digite:

```
demo.sh
```

Uma apresentação básica de funcionamento do simulador ocorrerá, note que este não tem inteligência artificial e os agentes executam as atividades de forma desorganizada.

## 6.1 Material e Métodos

O simulador foi instalado nas seguintes condições nas quais obtivemos sucesso.
Sistema Operacional: Kurumin Linux 8.0 ou Ubuntu
Processador: Core 2 Duo 2.2
Memória Ram: 1GB
HD: 20GB's
Monitor de 17"
Conexão com Internet

## 3.9 O arquivo `resultados.tex`

Logo após criaremos o arquivo `resultados.tex`, Figura 10, onde encontramos os seguintes códigos:

```
\section{Resultados}

O simulador é instalado e executado perfeitamente nas versões de sistemas operacionais Linux mencionadas, já em
sistemas operacionais Windows a execução não ocorre corretamente, pois diversos problemas ocorrem com a
comunicação do kernel do simulador com o sistema.

\endinput
```

Figura 10: Arquivo exemplo `resultados.tex`.

Para customizar o relatório, altere o texto do exemplo acima de acordo com os resultados obtidos do seu projeto.

## 3.10 O arquivo `consideracoesfinais.tex`

Logo após criaremos o arquivo `consideracoesfinais.tex`, Figura 11, onde encontramos os seguintes códigos:

```
\section{Considerações Finais}

O Simulador é a melhor forma de se executar um treinamento das redes neurais embutidas em cada agente, uma vez
que este software consegue efetuar simular uma catástrofe urbana com certa precisão.

A continuação do projeto se deve a implantação do módulos de inteligência artificial, o que ocorrerá nos
próximos meses.

\endinput
```

Figura 11: Arquivo exemplo `consideracoesfinais.tex`.

Para customizar o relatório, altere o texto do exemplo acima de acordo com as considerações finais do seu projeto.

Ao converter os documentos acima (`resultados.tex` e `consideracoesfinais.tex`) para PDF, temos a versão de apresentação mostrada na página que se segue:

# 7 Resultados

O simulador é instalado e executado perfeitamente nas versões de sistemas operacionais Linux mencionadas, já em sistemas operacionais Windows a execução não ocorre corretamente, pois diversos problemas ocorrem com a comunicação do kernel do simulador com o sistema.

# 8 Considerações Finais

O Simulador é a melhor forma de se executar um treinamento das redes neurais embutidas em cada agente, uma vez que este software consegue efetuar simular uma catástrofe urbana com certa precisão.

A continuação do projeto se deve a implantação do módulos de inteligência artificial, o que ocorrerá nos próximos meses.

## 3.11 Referências Bibliográficas

Por último teremos as Referências Bibliográficas, que não é um arquivo e será incluída no final do relatório do Projeto.

O código para inserção das Referências se encontra no arquivo `introducao.tex`, previamente visto. Para gerar as Referências Bibliográficas você deverá compilar o arquivo utilizando o comando *BibTex*, geralmente esse comando já está incluso na maioria dos editores de arquivos em LaTeX2$_\varepsilon$ . O capítulo Referências será exibido na página que se segue:

## Referências


[1] F.L Bellifemine. Developing multi-agent systems with jade. *Springer*, 2007. 4

[2] A.B.H Ferreira. Novo aurélio século xxi: O dicionário da língua portuguesa. 1999. 1

[3] RoboCupRescueHome. Building rescue systems of the future, 2009. 3

[4] Mike Wooldridge. Agent-oriented software engineering ii. *Springer-Verlag Lecture Notes in Computer Science*, 2222, 2002. 1

# 4 Conclusão

O sucesso na elaboração de um relatório do PI em LaTeX2$_\varepsilon$ utilizando o leiaute apresentado neste artigo, proporciona uma maior facilidade da construção do mesmo. Pois o aluno não terá que se preocupar com a formatação de seu relatório, consequentemente evitando possíveis erros e distrações durante a produção do relatório.

Ao contrário de editores de texto que utilizam a tecnologia WYSIWYG (What You See Is What You Get) onde o autor pode cometer um erro durante a elaboração de seu documento que acabe comprometendo a estrutura do seu trabalho por completo.

# 5 Agradecimentos

Os autores agradecem aos professores Delaine Vasconcelos Rosa e José Geraldo Costa pela revisão gramatical final.

THE PracTEX Journal

# Six LaTeX tools (with videos) (Portuguese)

Francisco Reinaldo et al

Abstract

English

In this paper we present several commonly-used LaTeX tools and show how to fine tune them. We focus mainly on six heterogeneous tools: MiKTeX, GSview, eXPert PDF Reader, Texmaker, JabRef, and LaTable.

Português

Neste artigo, nós apresentamos as ferramentas mais promissoras para leigos em LaTeX2e e como estas ferramentas deveriam ser instaladas. Nós focamos principalmente em seis tipos diferentes: MiKTeX, GSview, eXPert PDF Reader, Texmaker, JabRef e LaTable.

Francisco Reinaldo é atualmente professor na Área de Exatas no UnilesteMG, Brasil, em específico, o curso de Computação - Sistemas de Informação. No UnilesteMG, sua área de pesquisa é Inteligência Computacional (Diretor do LIC), mas dedica boa parte de seu tempo com estudos envolvendo LaTeX.

Victor Vasconcelos Moreira cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Estagiário do Projeto Informática Solidária - INFOSOL e colaborador do Laboratório de Inteligência Computacional – LIC, ambos os projetos sustentados pelo UnilesteMG.

Maria Tereza de Castro Costa cursa o 6º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiária no Laboratório de Inteligência Computacional (LIC), trabalhando com Redes Neurais Artificiais.

Tiago Faria Bicalho cursa o 7º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiário no Laboratório de Inteligência Computacional (LIC) e colaborador no Projeto Informática Solidária - INFOSOL.

Os autores podem ser contatados através de reinaldo.opus@gmail.com

- PDF version of paper
- Article source
- Miktex video
- GSview video
- PDF video

- Texmaker video 1
- Texmaker video 2
- JabRef video
- LaTable video
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/24

# Guia Visual Definitivo para Instalação de LaTeX2ε e suas Ferramentas de Apoio

Francisco Reinaldo[1], Maria Tereza de Castro Costa[2], Tiago Faria Bicalho[3], and Victor Vasconcelos Moreira[4]

Email [1]reinaldo.opus@gmail.com, [2]maryxb@gmail.com, [3]tiagofariabicalho@gmail.com, [4]victorvasconcelosfox@gmail.com

Resumo In this paper we present the most promising LaTeX2ε tools for common users and how these tools should be fine-tuned. We focus mainly on six heterogeneous tools specifically aimed at this purpose: MiKTeX, GSview, eXPert PDF Reader, Texmaker, JabRef, and LaTable.

## 1 Introdução

TeX é um conjunto de macros tipográficas na arte de formatação/impressão de textos. Sabendo que TeX é um pouco difícil de entender, a priori, optamos por LaTeX2ε. LaTeX2ε compacta as macros em pacotes de comandos para que sejam intuitivas e invisíveis para o usuário leigo. Para trabalhar com LaTeX2ε na potência máxima, utilizaremos algumas ferramentas de apoio - é interessante que você sempre tenha as últimas versões devidamente instaladas e configuradas em seu computador. Para ensinar como instalar LaTeX2ε e suas ferramentas de apoio, autores optam por manuais de referências ou sites do tipo padrão. Infelizmente, a leitura desses manuais exige uma dedicação extra, muito tempo do aluno e, na maioria das vezes,acaba desmotivando-o a prosseguir.

Este tutorial apresenta, em várias vídeo-aulas narradas através de Tiago Faria Bicalho, como o usuário comum deve corretamente instalar e configurar passo-a-passo as ferramentas gratuitas para obter sucesso no desenvolvimento textual com LaTeX2ε.

# 2 Tópicos Trabalhados nos Vídeos

Nestas vídeo-aulas, nós apresentaremos os todos os procedimentos necessários para instalação do LaTeX2$\varepsilon$ e das ferramentas necessárias trabalhar com textos em LaTeX2$\varepsilon$. Assim, selecionamos as ferramentas mais utilizadas por nós, tendo em vista a confiabilidade que elas ofereceram durante longos anos de trabalho.

Primeiramente, nós realizaremos a instalação completa do LaTeX2$\varepsilon$ para que vocês possam ver como é simples. Para esta instalação, utilizaremos a distribuição MikTeX. MiKTeX é uma atualizada implementação TeX para sistemas operacionais Microsoft Windows. Depois que LaTeX2$\varepsilon$ (MikTeX) foi devidamente instalado, precisamos de uma ferramenta de suporte para visualizar arquivos .PS, assim GSview é a ferramenta de interface gráfica que precisamos. Na sequência, necessitamos de um visualizador/leitor de PDFs que seja robusto e ao mesmo tempo elegante. Se você estava em busca de um leitor de PDF alternativo ao Adobe Reader, que fosse completo e muito rápido e leve, então o eXPert PDF Reader é ideal porque ele, além de substituir o Acrobat Reader com êxito, também permite copiar e colar textos, imprimir, trocar a interface, fazer buscas rápidas, colocar estampas ou marcas, atalhos, entre outros. Seguindo esta linha, é necessário que tenhamos um editor de texto para facilitar a vida com LaTeX2$\varepsilon$ nos primeiros passos, pois é ele que facilitará a geração dos PDFs. Assim, Texmaker é uma editor de LaTeX2$\varepsilon$ que integra muitas características de outras ferramentas de edição de texto para desenvolver documentos com LaTeX2$\varepsilon$ em uma única aplicação. Texmaker funciona nos sistemas UNIX, Mac OS X e Windows e também é gratuito. Não deixando de lado, uma ferramenta para construir, manusear e gerenciar bibliografia que nós usamos muito, quando se trata de textos acadêmicos, é o JabRef. Por fim, para quem gosta de construir tabelas de forma fácil e com uma altíssima qualidade, é necessário ter LaTable.

Assim temos neste guia abaixo, a lista de endereços para download das ferramentas e nome dos vídeos explicativos:

**Distribuição de LaTeX2ε: MikTeX(Windows)** em
ftp://dante.ctan.org/tex-archive/systems/win32/protext/ProTeXt-2.2.1-102109.exe.
Vídeo-aula: arquivo `miktex.mp4`

**Visualizador de PS: GSView (Windows)** em
http://mirror.cs.wisc.edu/pub/mirrors/ghost/ghostgum/gsv49w32.exe.
Vídeo-aula: arquivo `gsview.mp4`

**Visualizador de PDF: eXPert PDF Reader (Windows)** em
http://www.visagesoft.com/downloads/get.php/vspdfreader.exe.
Vídeo-aula: arquivo `expertpdf.mp4`

**Editor de texto: Texmaker (win/mac/unix)** em
http://www.xm1math.net/texmaker/texmakerwin32_install.exe.
Vídeo-aula: arquivo de instalação `texmakerSetup.mp4`
Vídeo-aula: arquivo de configuração `texmakerPreferences.mp4`

**Construtor de Bibliografia: Optional: JabRef (win/mac/unix)** em
http://sourceforge.net/projects/jabref/files/jabref/2.5/JabRef-2.5-setup.exe/download.
Vídeo-aula: arquivo `jabref.mp4`

**Construtor de Tabelas: LaTable (Windows)** em
http://g32.org/files/latable/latable-0_7_2.zip.
Vídeo-aula: arquivo `latable.mp4`

# 3 Dicas de estudos

Thomas E. Price and Lance Carnes. **LaTeXQuick Start: A first guide to document preparation**. Personal TeX, Inc. 2009.

Klaus Steding-Jessen. **LaTeX: Uma Alternativa mais Eficiente Comparada aos Sistemas WYSIWYG**. http://biquinho.furg.br/tex-br/doc/artigo-1-jessen/

**All You Need to Know about Latex**. http://xpt.sourceforge.net/techdocs/language/latex/

**TeX-BR: Página dos usuários brasileiros de (La)TeX.** http://biquinho.furg.br/tex-br/

Kjell Magne Fauske. **TeXample.net is a web site dedicated to the wonderful world of TeXand friends**. http://www.texample.net/

Também apresentamos um texto interessante extraído do link: http://www.tug.org/pracjourn/2005-1/asknelly/.

**Q: Por que um estudante universitário deve usar LaTeX?**

**R:** Já ouvi esta pergunta várias vezes nos últimos dois anos. Como introduzi LaTeX em meu departamento, tenho várias respostas que dou aos meus alunos.

A primeira razão que dou é que eles estão na faculdade para aprender. Aprender a usar LaTeX irá tornar mais fácil para os alunos aprenderem a usar outras linguagens. Esta será uma habilidade importante para que a Internet e as suas línguas associados continuem florescendo. A indústria editorial está caminhando em direção a XML e tecnologias relacionadas, e os estudantes deveriam querer estar preparado para prosperar neste novo cenário.

Em seguida, há uma expectativa crescente de estudantes que entram nos programas de pós-graduação para saber alguma forma de composição de software. Minha introdução ao LaTeX veio em meu primeiro curso de pós-graduação quando o professor exigiu que nós digitássemos nossas resoluções de deveres de casa. Ele não se importava com o programa utilizado, mas ele concordou em cobrir os custos dos manuais de LaTeX para todos os estudantes interessados.

Então, eu às vezes ouço a pergunta: "Por que há uma expectativa crescente de estudantes, especialmente de ciência, em saber LaTeX" Eu respondo: por falar sobre a saída fantástica gerado pelo LaTeX. É verdade que os programas de processamento de texto podem criar equações, mas na minha opinião, a saída é muito ruim comparado a qualidade do LaTeX. Acrescente a isso o fato de que para a maioria destes programas é preciso voltar ao compositor de equação para cada equação e você logo perceberá que está gastando horas criando equações. Enquanto que em LaTeX, após algumas semanas de estudo dedicado, a pessoa é capaz de entrar equações praticamente tão rápido quanto a pessoa pode digitar um texto qualquer.

A primeira coisa que eu digo aos meus alunos é que LaTeX permite-lhes a oportunidade de centrar-se na escrita. Com LaTeX o autor tem de organizar os diferentes segmentos do documento e se preocupar com o conteúdo. O arquivo de classe irá lidar com a formatação. Se você decidir reordenar os capítulos ou seções em um documento, não ficará horas de trabalho brigando com o editor de textos e as figuras nele, ou até tentar renumerar todas as equações, figuras e

referências. Meus alunos e eu achamos que isto é um grande benefício.

Estas são apenas algumas das razões que eu acho que os estudantes podem se beneficiar ao aprender LaTeX. Para a maioria dos alunos será um investimento de algumas semanas que irá abrir as portas para uma viagem maravilhosa. Eu ainda estou aprendendo coisas sobre tipos de configuração de LaTeX doze anos mais tarde e eu acho que é uma coisa boa.

Esta pergunta foi respondida por John W. Breitenbucher. Jon é professor assistente de Ciências Matemáticas na Faculdade de Wooster (Ohio, E.U.A.).

THE PracTEX Journal

# Playing with Flash in ConTeXt-mkiv

Luigi Scarso

Abstract

English

A first attempt to adapt flashmovie.sty to ConTeXt MKIV to produce a flash movie with MetaPost and swftools.

Italian

Un primo tentativo di adattare flashmovie.sty a ConTeXt MKIV per produrre animazioni flash con MetaPost e swftools.

Luigi Scarso is an engineer who works in a publishing/printing company in northeastern Italy. He mainly uses "ConTeXt" for automatic typesetting and lately has been exploring the Lua part of "ConTeXt-mkiv".

You can contact him by sending email to luigi.scarso@gmail.com.

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

For submission to The PracTEX Journal
Article revision 2010/05/25

# Playing with Flash in ConTEXt-mkiv

Luigi Scarso

Email luigi.scarso@gmail.com
Address Padova,
Italy.

Abstract This describes a first attempt to adapt flashmovie.sty to ConTEXt-mkiv to produce a flash movie with METAPOST and swftools.

## 1 Introduction

Beginning with release 9, AdobeReader comes with an embedded Flash player. The recent addition to CTAN of the flashmovie package by Timo Hartmann prompted me to investigate the feasibility of an integration between ConTEXt-mkiv and Flash by using pdf2swf[1]. In these experiments, all tests were performed under Linux Ubuntu 8.04 with AdobeReader 9.2 installed. Testing was not done on Windows or Mac systems, but it is likely these techniques will work there as well. This article assumes that readers are familiar with ConTEXt-mkiv (more information is available at http://wiki.contextgarden.net/ConTeXt_Minimals).

## 2 First step: Adapting flashmovie.sty

Quoting from CTAN (flashmovie.sty) . . . *allows direct embedding of flash movies into PDF files. It is designed for use with pdflatex. The package takes advantage of the embedded Adobe Flash player in Adobe Reader 9; the reader is invoked with the 'rich media annotation' feature, described in "Adobe Supplement to the ISO 32000 BaseVersion: 1.7 ExtensionLevel: 3". This method of embedding movies is attractive since it removes all platform dependencies; however, the user is required*

[1]. pdf2swf is a program from http://www.swftools.org that is based on xpdf codebase.

*to use Acrobat 9.* `flashmovie.sty` is a simple style, so there is almost nothing to change for use with ConTEXt-`mkiv`: just add a suffix `ctx` to the original `\flashmovieembedfile` and `\flashmovie` macros, and save them in `supp-swf.mkiv`s.

```
%D \module
%D   [       file=supp-swf,
%D        version=2009.12.31,
%D          title=\CONTEXT\ Support Macros,
%D       subtitle=swf inclusion,
%D         author=Luigi Scarso,
%D           date=\currentdate,
%D      copyright={Luigi Scarso}]

\unprotect

\newcount\filespecnum
\newcount\configurationnum
\newcount\contentnum
\newcount\settingsnum
\newdimen\xxwidth
\newdimen\xxheight
\newbox\xxcontent


\def\flashmovieembedfilectx#1{
\immediate
\pdfobj stream
     attr {  /Type/EmbeddedFile  }
     file {#1}
  \immediate
  \pdfobj { <<
      /Type /Filespec
      /F  (#1)
      /UF (#1)
      /EF << /F \the\pdflastobj\space 0 R >>
    >>}}

\def\flashmoviectx[#1]{%
\getparameters[flashmovieparams][width=4cm,height=4cm,#1]
```

```
\flashmovieembedfilectx{\csname flashmovieparamsfile\endcsname}
\filespecnum=\pdflastobj
\immediate
\pdfobj
    {<<
      /Instances
        [<<
          /Asset \the\filespecnum\space 0 R
          /Params << /Binding /Foreground >>
        >>]
      /Subtype /Flash
    >>}
  \configurationnum=\pdflastobj

  \immediate
  \pdfobj
    {<<
      /Assets << /Names [(\csname flashmovieparamsfile\endcsname)
              \the\filespecnum\space 0 R] >>
      /Configurations [\the\configurationnum\space 0 R]
    >>}
\contentnum=\pdflastobj
\immediate
\pdfobj
    {<<
      /Activation
        << /Type /RichMediaActivation
           /Condition /PO
           /Configuration \the\configurationnum\space 0 R
           /Animation
             << /Subtype /Linear
                /Speed 1
                /Playcount 1
             >>
           /Presentation
             << /PassContextClick false
                /Style /Embedded
                /Toolbar false
                /NavigationPane false
                /Transparent true
```

```
                /Window
                  << /Type /RichMediaWindow
                     /Width  << /Default 100 /Min 100 /Max 100 >>
                     /Height << /Default 100 /Min 100 /Max 100 >>
                     /Position
                       << /Type /RichMediaPosition
                          /HAlign /Near
                          /VAlign /Near
                          /HOffset 0
                          /VOffset 0
                       >>
                  >>
              >>
        >>
      /Deactivation
        << /Type /RichMediaDeactivation
           /Condition /XD
        >>
    >>}
\settingsnum=\pdflastobj
\setbox\xxcontent=\hbox to \csname flashmovieparamswidth\endcsname{%
\hss\vbox to\csname flashmovieparamsheight\endcsname{%
\hsize=\csname flashmovieparamswidth\endcsname\vss \vss }\hss}%
\xxwidth=\wd\xxcontent%
\xxheight=\ht\xxcontent%
\immediate\write16{wd=\the\xxwidth,ht=\the\xxheight,\the\textwidth}%
\box\xxcontent%
\pdfannot width \csname flashmovieparamswidth\endcsname height %
\csname flashmovieparamsheight\endcsname depth 0pt {%
       /Subtype /RichMedia
       /RichMediaContent  \the\contentnum\space 0 R
       /RichMediaSettings \the\settingsnum\space 0 R
}}
\protect
```

To test this, visit the site http://melusine.eu.org/syracuse/metapost/animations/chupin/?idsec=filtre and download the sample file `filtre-num.swf`. Use this file as follows:

```
%%test-ctx.tex
\input supp-swf.mkiv
\starttext
\flashmoviectx[width=\textwidth,height=\textwidth,file=filtre-num.swf]
\stoptext
```

Remember that in ConTEXt-mkiv a pdf is made with:

```
#>context test-ctx
```

# 3 Integrating Metapost and swftools

Nearly everything needed for METAPOST and Flash can be found at the nicely designed site http://melusine.eu.org/syracuse. Given that I now have a way to manage swf files in ConTEXt-mkiv, the next step is to build a primitive system which imitates some of these beautiful animations with METAPOST.

I decided to use the ``ConTEXt-mkii'' (aka ``external'') method:

1. save a "metapost animation sequence" in an external file;
2. run an external mpost on it;
3. convert the results file in one pdf by using another external ConTEXt-mkiv run;
4. convert the pdf in swf with an external call to pdf2swf.

This method *does not* use the full power of LuaTEX and ConTEXt-mkiv: for example ConTEXt-mkiv comes with a METAPOST interpreter embedded, but it cannot be used because an external mpost program is being called. Also, I do not extend luatex with a possible Lua binding to a libpdf2swf.so, simply because libpdf2swf.so is to be built entirely. So this method of porting it to ConTEXt-mkii is simpler, but the fun here is to use the Lua language.

I split a "metapost animation sequence" between the preamble and body; the preamble can be shared with other bodies, and the body is enclosed in a for loop starting from start to end_limit with step step_value:

```
for frame:=start step step_value until end_limit :
 beginfig(frame)
 <body here>
 endfig;
end.
```

Let's continue with the `supp-swf.mkiv` module. The first step is to manage the creation of METAPOST files for animation. Each frame is a METAPOST figure *n*`.mps` which is saved in a `name` (default:`out-mps`) folder.

```
%% A lua table to collect animations
\ctxlua{MetapostFramesTable = {}}

%% animation preamble
\long\def\startPreambleMetapostFrames[#1]#2\stopPreambleMetapostFrames{%
\getparameters[preamblemps.][name={out-mps},#1,body={#2}]
\ctxlua{%
MetapostFramesTable[tostring("\csname preamblemps.name\endcsname")] =
                     tostring("\csname preamblemps.body\endcsname")
}}
%%
\long\def\startMetapostFrames[#1]#2\stopMetapostFrames{%
\getparameters[mps.][end=360,start=0,step=1,
                      name={out-mps},preamble=preamble,
                    mpformat=metafun,#1,body={#2}]
\dostopMetapostFrames
}
%%
\def\dostopMetapostFrames{%
\ctxlua{%
 local preamble =
    MetapostFramesTable[tostring("\csname mps.preamble\endcsname")] or ''
 local end_limit = tonumber(\csname mps.end\endcsname)
 local step_value = tonumber(\csname mps.step\endcsname)
 local start = tonumber(\csname mps.start\endcsname)
 local outdir = tostring("\csname mps.name\endcsname")
 local outfile = tostring("\csname mps.name\endcsname") .. ".mp"
 local mpformat = tostring("\csname mps.mpformat\endcsname")
 local outputtemplate = string.format('outputtemplate := "./\%s/\%\%\%06c.mps";',outdir)
```

```
 local frames_limit = string.format("numeric frame_min, frame_max; frame_min=\%0.5f;
                                    frame_max=\%05f;",start,end_limit)
 local mpgraphic_body = tostring("\csname mps.body\endcsname")
 local start_frame = string.format("for frame:=\%d step \%d until \%d:beginfig(frame);",
                                   start,step_value,end_limit)
 local stop_frame = " endfig;endfor;"
 local mpgraph = preamble ..
                 outputtemplate ..
                 frames_limit ..
                 start_frame .. mpgraphic_body .. stop_frame ..
                 'end.'
 dir.mkdirs(outdir)
 io.savedata(outfile,mpgraph)
 mpost_execute = string.format("mpost --mem=\%s \%s ",mpformat,outfile)
 os.execute(mpost_execute)
}}
```

Note: I assume that `metafun` and METAPOST formats are placed in the same directory as the animation file and that I'm saving all mps (i.e. postscript) files in the local folder where the animation file resides.

The next step is to create a single `pdf` from all these `*.mps` and then a `swf`, but these two steps can be merged into a `\MakeSwfFromMps` macro that converts all `*.mps` files into a single `swf` file. Here I use the ConTEXt-`mkiv` snippet

```
\starttext
\pagefigure[folder/fig1.mps]
\pagefigure[folder/fig2.mps]
\stoptext
```

to convert `folder/fig1.mps` and `folder/fig2.mps` into one pdf, and then I call `pdf2swf` on the resulting pdf.

```
\def\MakeSwfFromMps[#1]{%
\getparameters[mkpdfmps.][end=360,start=0,step=1,name={out-mps},fps=60,#1]
\ctxlua{%
 local outdir=tostring("\csname mkpdfmps.name\endcsname")
 local outfile_tex=tostring("\csname mkpdfmps.name\endcsname") .. ".tex"
 local outfile_pdf=tostring("\csname mkpdfmps.name\endcsname") .. ".pdf"
```

```
 local end_limit=tonumber(\csname mkpdfmps.end\endcsname)
 local start = tonumber(\csname mkpdfmps.start\endcsname)
 local step =  tonumber(\csname mkpdfmps.step\endcsname)
 local fps = tonumber(\csname mkpdfmps.fps\endcsname)
 local tex_body =''
 local tex_snippet
 local context_execute
 local pdf2swf_execute
 for i=start,end_limit,step  do
    tex_body = tex_body .. string.format("\\pagefigure[\%s/\%06d]",outdir, i)
  end
 tex_snippet =
     "\\nopdfcompression\\pdfminorversion4\\starttext " .. tex_body .." \\stoptext"
 io.savedata(outfile_tex,tex_snippet)
 context_execute = string.format("context  --batchmode \%s" ,outfile_tex)
 os.execute(context_execute)
 pdf2swf_execute = string.format("pdf2swf --quiet -s framerate=\%s \%s" ,fps ,outfile_pdf)
 os.execute(pdf2swf_execute)
}}
\protect
```

# 4 Example

I attempt to imitate http://melusine.eu.org/syracuse/metapost/animations/chupin/?idsec=filtre by creating four animations of the function `tran` that share the same `transfert` preamble:

```
\startPreambleMetapostFrames[name=transfert]
path Hcurve;
% Fonction de calcul du module du filtre
vardef trans(suffix pole,zero)(expr nr_poles,nr_zeros,w) =
  save H;
  numeric H;
  H:=1;
  for i:=0 upto (nr_poles-1): H:=H/abs(w-pole[i]); endfor;
  for i:=0 upto (nr_zeros-1): H:=H*abs(w-zero[i]); endfor;
  H
enddef;
```

```
def doit(suffix pole,zero)(expr nr_poles,nr_zeros,frame,u,max_estimated) =
 %%setbounds currentpicture to fullcircle scaled (2u+.5u);
 numeric i;
 pair p[];
 pair w ;

 i:=frame/2;
 w := (cosd(i)*u,sind(i)*u);
 H:=trans(pole,zero)(nr_poles,nr_zeroes,w);
 if i=0:
  Hcurve := (0,H*u);
 else:
  Hcurve := Hcurve--(i,H*u);
 fi;
 draw Hcurve withcolor red withpen pencircle scaled 2pt;
 drawarrow (frame_min,0) -- (frame_max/2+0.1*frame_max,0);
 drawarrow (frame_min,0) -- (0,max_estimated); %% estimated

 picture p ;
 p := image (
    draw fullcircle scaled 2u withcolor 0.8white withpen pencircle scaled 0.5pt;
    draw (0,0) -- (cosd(i)*u,sind(i)*u) withcolor black withpen pencircle scaled 0.5pt;
    drawarrow (-1.1u,0) -- (1.2u,0) withcolor 0.8white withpen pencircle scaled 0.5pt;
    drawarrow (0,-1.1u) -- (0,1.2u) withcolor 0.8white withpen pencircle scaled 0.5pt;
    for k:=0 upto (nr_zeroes-1):
     drawdot zero[k] withcolor blue withpen pencircle scaled 2pt ;
     endfor;
    for k:=0 upto (nr_poles-1):
      drawdot pole[k] withcolor red withpen pencircle scaled 2pt ;
    endfor;
   );
 draw p shifted (-1.3u,0);

enddef;
\stopPreambleMetapostFrames
%%
%%
%%
%%
```

```
%% Main code
\starttext
\startMetapostFrames[name=trasf1,preamble=transfert]
 numeric u;
 u:=1cm;
 numeric max_estimated ;
 max_estimated := 7*u;

 % zeros and poles
 pair zero[],pole[];
 numeric nr_zeroes,  nr_poles;
 nr_zeroes:=2;
 nr_poles:=2;
 zero[0]:=(0.86u,0.5u);
 zero[1]:=(-.42u,0.90u);
 pole[0]:=(-0.52u,0.26u);
 pole[1]:=(0.55u,0.58u);
 doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf1}]
%
\startMetapostFrames[name=trasf2,preamble=transfert]
 numeric u;
 u:=1cm;
 numeric max_estimated ;
 max_estimated := 7*u;

 % zeros and poles
 pair zero[],pole[];
 numeric nr_zeroes,  nr_poles;
 nr_zeroes:=2;
 nr_poles:=2;
 zero[0]:=(0.86u,0.5u);
 zero[1]:=(-.42u,0.90u);
 pole[0]:=(-0.52u,0.26u);
 pole[1]:=(0.25u,0.29u);
 doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf2}]
%
```

```
\startMetapostFrames[name=trasf3,preamble=transfert]
 numeric u;
 u:=1cm;
 numeric max_estimated ;
 max_estimated := 7*u;

 % zeros and poles
 pair zero[],pole[];
 numeric nr_zeroes,  nr_poles;
 nr_zeroes:=2;
 nr_poles:=2;
 zero[0]:=(0.86u,0.5u);
 zero[1]:=(-.42u,0.90u);
 pole[0]:=(-0.52u,0.26u);
 pole[1]:=(0.1u,0.1u);
 doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf3}]

\startMetapostFrames[name=trasf4, preamble=transfert]
 numeric u;
 u:=1cm;
 numeric max_estimated ;
 max_estimated := 7*u;

 % zeros and poles
 pair zero[],pole[];
 numeric nr_zeroes,  nr_poles;
 nr_zeroes:=2;
 nr_poles:=2;
 zero[0]:=(1u,0);
 zero[1]:=(-1u,0);
 pole[0]:=(cosd(135)*0.8u,sind(135)*0.8u);
 pole[1]:=(cosd(45)*0.7u,sind(45)*0.7u);
 doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf4}]
%
\setbox1=\vbox{\flashmoviectx[width=0.45\textwidth,
                              height=0.25\textheight,file=trasf1.swf]}
```

```
\setbox2=\vbox{\flashmoviectx[width=0.45\textwidth,
                              height=0.25\textheight,file=trasf2.swf]}
\setbox3=\vbox{\flashmoviectx[width=0.45\textwidth,
                              height=0.25\textheight,file=trasf3.swf]}
\setbox4=\vbox{\flashmoviectx[width=0.45\textwidth,
                              height=0.25\textheight,file=trasf4.swf]}

\hbox{\copy1\copy2}
\hbox{\copy3\copy4}
%
\stoptext
```

*( If you opened this file using AdobeReader 9, you should see four animations right here.)*

# 5 Conclusion

This first attempt showed that it is possible to use ConTEXt-`mkiv` to make interesting animations with Flash, but at this stage `supp-swf.mkiv` must be considered as a "sketch". However, this was enough to get Hans Hagen interested, and he immediately showed me two lua files (`grph-swf.lua` and `lpdf-swf.lua`) that replace `supp-swf.mkiv` in the correct ConTEXt-`mkiv` way. As a result, I can now research the next steps in the following order:

1. complete `back-swf.mkiv`, `grph-swf.lua` and `lpdf-swf.lua` in a consistent manner (they are already in the `minimals-beta` distribution);
2. study how to use the internal mplib, and see if is possible to avoid a call to the external `context` process;
3. provide the end user with the parameters used by `pdf2swf` for the conversion from pdf to swf;
4. study a Lua binding for `pdf2swf` — there is already a `gfx.so` shared library, but it used for the Python binding.
5. verify the performance of AdobeReader and test the limits of pdf2swf.

THE PracTEX Journal
TEX USERS GROUP

# Enhancing Command Completion for TeXShop

Herbert Schulz

Abstract

LaTeX environments and commands are rather wordy markup. This markup makes the author's intentions clear but it is often difficult to remember how to use it. Using Command Completion, authors can write a few letters and trigger an expansion into complete environments and commands, along with guides to the use of their arguments. In this paper I present an enhancement to Command Completion in TeXShop that allows more consistent completions, and inclusion of short comments to help authors remember the order and contents of the arguments to those environments and commands.

I'm a retired Professor of Physics, having spent 32 years teaching at a Community College in the Chicago area. My experience using TeX started in the mid-1980s and I used it for all my paperwork, and especially for making exams. That doesn't mean I really have more than 20 years experience, since there was a period of time, during the mid to late 1990s, when I was attempting to use FrameMaker on the Mac. When FrameMaker was no longer being supported on the Mac, and OS X came along, it was only natural for me to switch back to TeX. I first used Gerben Weirda's re-distribution of teTeX with additions and then, finally, MacTeX. I've used many editors over the years but have settled on TeXShop for most of my work, because it is both easy to use and easily extensible with custom engines and other tools.

You can contact me by sending email to herbs2@mac.com.

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/02/12

# Enhancing Command Completion for TEXShop

Herbert Schulz
herbs2@mac.com

Abstract LATEX environments and commands are rather wordy markup. These make the intentions of the author easy to determine but more difficult to write. Using Command Completion, authors can write a few letters and trigger an expansion into complete environments and commands along with ways of going between arguments of those commands. In this paper I present an enhancement to Command Completion in TEXShop that allows more consistent completions and inclusion of short comments to help authors remember the order and contents of the arguments to those environments and commands.

## 1 Introduction & History

TEXShop is a popular Editor, Viewer and TEX Front End on the Macintosh. As of v1.34 TEXShop has offered a Command Completion facility that is reasonably powerful, if under-utilized. Command Completion in TEXShop allows continuations (Completions) and substitutions (Abbreviations) for a set of characters bounded on the left by a Word Boundary Character[1] and triggered by the Escape (`Esc`) key.

With the help of the good folks on the Mac OS X TeX e-mail list[2], I put together a `CommandCompletion.txt` file along with associated Applescript macros to take advantage of that facility. The completions and abbreviations supplied often contain bullet characters, '•', called Marks[3], as placeholders for command arguments or to easily get to the end of an environment. Skipping forward/backward and

1. The Word Boundary Characters are space, tab, linefeed(newline), period, comma, semicolon, colon, {, }, (, ) or \ (actually the `TeX Command Character` which can vary in different implementations). The { and \ also become part of the expansion.
2. Subscribe by sending an e-mail to <mailto:MacOSX-TeX-on@email.esm.psu.edu>.
3. Previously called Tabs.

```
\rule[#INS#]{•}{•}
```

```
\rule[|]{•}{•}
```

Figure 1: Original `CommandCompletion.txt` contents and result in the document source. Here | is the insertion point.

selecting/deleting these Marks were accomplished using macros[4]. Most of the abbreviations were inspired by those used in the FasTEX[5] set used with TypeIt4Me[6]. The completion/abbreviation list created for TEXShop is the basis for the similar feature used in TEXworks[7].

Early in 2006, Hugh Neary sent me some Objective C code to implement the macros as an integral part of TEXShop. I modified that code and used it for quite a while in a personal build of TEXShop.

In April of 2006, Will Robertson added an extension to the Applescript macros that allowed the addition of an explanatory Comment within the arguments; the macros selected the Mark and Comment so typing replaced both with the entered information. This was reported in the `mactextoolbox-talk` list and some discussion followed it about the best delimiters for Comments but there was no final conclusion; the original suggestion of "•‹" and "›[8]" has been retained. Unfortunately, the original completion code could only position the insertion point (i.e., the cursor) so the initial selection could only have zero length (see Figure (1)); inconsistent with the behaviour with other arguments since you could not move back to the first argument using the macros.

At that point I decided to do a complete re-write of the code to implement a reasonably general version of Will Robertson's ideas and, at the same time, extend

4. The original `CommandCompletion.txt` files, macros and documentation are still available as `CommandCompletion.zip` from <http://homepage.mac.com/herbs2/>.
5. FasTEX was developed by Filip G. Machi, Jerrold E. Marsden and Wendy G. McKay. For more information see the FasTEX web page, <http://www.cds.caltech.edu/~fastex/>.
6. TypeIt4Me, by Riccardo Ettore, version 3 and later is a preference pane that allows abbreviation replacement in most OS X programs. See the TypeIt4Me web page, <http://www.typeit4me.com/>, for more information.
7. TEXworks is a multi-platform Editor, Viewer and TEX Front End using the same design philosophy as TEXShop. It was written by Jonathan Kew. More information is available at <http://www.tug.org/texworks/>.
8. Note that '‹' and '›' are single "guillemot" glyphs, not '<' and '>'.

the Command Completion code so that the implementation was consistent starting with the first argument of the completion. The result is backward compatible with the original behaviour of that code but with additional capabilities.

# 2 Changes to TEXShop.

Four interconnected changes were made in TEXShop: an addition to the way TEXShop handles completions from `CommandCompletion.txt`; a new menu with commands for searching and selecting Marks within completions; the ability to have comments attached to Marks; a new `CommandCompletion.txt` file that takes some advantages of the previous three changes. The rest of this section discusses each of these changes in more detail.

## 2.1 Changes to Completion Handling

Completions (in the `CommandCompletion.txt` file) in previous versions of TEXShop could contain a single `#INS#` command for the positioning of the insertion point within the completion.

This version of TEXShop allows completions to have *two* copies of `#INS#` and the text between them is selected. A single `#INS#` behaves the same as before; there is complete backward compatibility with previous versions of TEXShop.

## 2.2 A New `Source`→`Completion`→`Marks` Menu

The new `Source`→`Completion`→`Marks` menu contains commands to search for, move to and select Marks and Comments. The commands are shown in Table (1) and the default command menu appears in Figure (2). The `(Del)` versions of the search commands only show in the menu when the Option (`Opt`) key is pressed and the `Insert Comment` command only appears when you hold down the Control (`Ctl`) key. The `Insert Mark` command is added since TEXShop's autocompletion (keybinding) facility will insert `\bullet` in the document when the keystroke that normally inserts a '•' (`Opt`-8 with a US keyboard mapping) is pressed.

| Menu Item | Shortcut | Internal Connection |
|---|---|---|
| `Next Mark` | `Ctl-Cmd-F` | Jump to and select the next Mark and/or Comment. |
| `Next Mark (Del)` | `Ctl-Opt-Cmd-F` | Jump to and select the next Mark and/or Comment and delete the Mark. This is most useful when you have nested environments to automatically delete a Mark at the end of an inner environment. |
| `Previous Mark` | `Ctl-Cmd-G` | Like `Next Mark` but search backwards. |
| `Previous Mark (Del)` | `Ctl-Opt-Cmd-G` | Like `Next Mark (Del)` but search backwards. |
| `Insert Mark` | `Cmd-8` | Places a Mark at the insertion point. Handy for creating completions in `CommandCompletion.txt`. |
| `Insert Comment` | `Ctl-Cmd-8` | Places a Comment Skeleton, "•‹›" with the insertion point before the "›", at the insertion point. Handy for creating comments in `CommandCompletion.txt`. |

Table 1: Commands in the `Source`→`Completion`→`Marks` Menu.

## 2.3 Comments

The change mentioned in the previous sub-sections allow completions to contain Comments—short "memory joggers" that have some information about the contents of a given argument. The comments are contained within arguments and are surrounded by "•‹" and "›" within the arguments; if the first argument contains a comment it should be surrounded by two `#INS#` so it is the initial selection.

## 2.4 The New `CommandCompletion.txt` File

The `CommandCompletion.txt` file that comes with this version of TEXShop replaces all single `#INS#` commands by `#INS#•#INS#` so that the initial selection is a se-

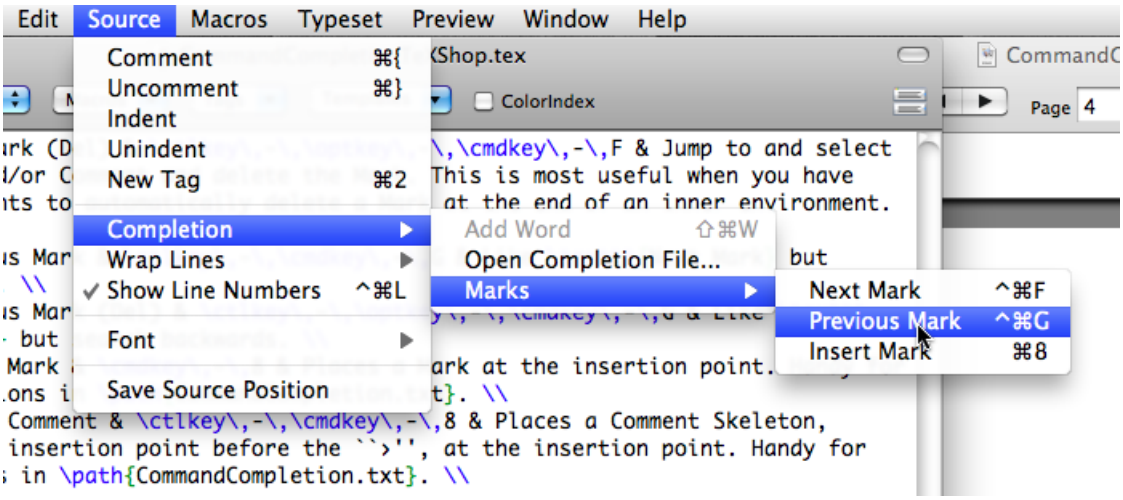



Figure 2: The Default `Source` → `Completion` → `Marks` Menu.

```
\rule[#INS#•#INS#]{•}{•}
```

```
\rule[•]{•}{•}
```

Figure 3: New `CommandCompletion.txt` contents and result in document.

lected Mark, •, for consistency in appearance and behavior when using the commands in the `Source`→`Completion`→`Marks` Menu. Figure (3) shows what this looks like.

The file, in addition, contains a few (too few?) examples of using comments.

# 3 Usage

## 3.1 Command Completion

A Command Completion is typically used to set up environments. To do this type `\b` and `Esc`; this should return `\begin{`. Then start to type the environment name; e.g., `eq` and `Esc` will give

```
\begin{equation}
•
\end{equation}•
```

while the next Esc gives eqnarray followed by it's *-variant. After entering your equation text at the cursor run the Source→Completion→Marks→Next Mark command and the cursor will select (and delete if Next Mark (Del) is used) the next '•' so you can start to type following text.

The macros are also handy for commands with multiple arguments. For example, to create a new command with an optional argument type \new or \newc and then Esc three times to get

```
\newcommand{•}[•][•]{•}
```

with the first mark selected. After entering the new command's name, please use the Next Mark command to jump to the next argument, etc.

## 3.2 Abbreviations

In addition to command completion, there are many abbreviations for commands. The principal difference is that the abbreviations are not just the start of a command name. For example typing benu and then pressing Esc *at the beginning of a line*[9] will produce the complete enumerated list environment:

```
\begin{enumerate}
\item
•
\end{enumerate}•
```

as you might expect. Abbreviations like this exist for many environments as well as sectioning commands. Alternate command versions with one or more options or *-variants have names that end with 'o' (one or more) or 's' respectively: e.g., sec and two presses of Esc or secs and a single Esc at the start of a new line give \section*{•}. By the way, After typing the text for the first item, typing it and Esc on a new line will generate another \item with a selected Mark on the line below it; continued presses of Esc will give \item[•] with a Mark on the

9. Or after any other Word Boundary Character.

following line, \textit{•} and finally \itshape before returning to the original it.

Remember that you must have one of the Word Boundary Characters before use; otherwise the substitution won't operate properly. This a not a problem with environments and sectioning commands, since you usually start them on a new line, but it can be for other abbreviations. Therefore many abbreviation also have a '\' version; e.g., `tt and Esc will not expand properly since the '`' isn't a Word Boundary Character while `\tt and Esc will expand to `\texttt{•} and a second Esc will give the declaration `\ttfamily[10].

Many of the Greek characters and in-line math versions of the Greek characters have abbreviations with the following rules:

1. The abbreviations for Greek characters all start with an 'x' and a notation for the character: e.g., xa or \xa[11] and Esc give \alpha.

2. The var version of several Greek characters start with 'xv' and the notation for the character: e.g., xth gives \theta while xvth and Esc gives \vartheta.

3. To get capitals for some letters use an 'xc': e.g., xg gives \gamma while xcg gives \Gamma.

4. Finally, preceding by a 'd' gives the following Greek character as an in-line math equation: e.g., dxcd gives \(\Delta\).

Abbreviations will be completed and cycle through matches just like the command completions: e.g., both the abbreviation newcoo (note the 'oo' at the end of the abbreviation) and Esc or newc followed by three Esc key presses on a new line give \newcommand{•}[•][•]{•}, the \newcommand with two optional arguments. There are alternate abbreviations for some commands: e.g., ncm gives the same result as newc.

Read the CommandCompletion.txt file to see what abbreviations are available; all lines with ':=' are abbreviations. Naturally, you can change them to suit your needs, adding or deleting others.

10. Similar abbreviations exist for bf, sf, sc, etc. Math versions have a preceding m; e.g., mbf and Esc will give \mathbf{•}.
11. All of the Greek character abbreviations have \ versions.

```
\rule[#INS#•<lift>#INS#]{•<width>}{•<height>}

\rule[•<lift>]{•<width>}{•<height>}
```

Figure 4: New `CommandCompletion.txt` contents with comments and result in document.

### 3.3 Comments

I tend to remember the arguments for commands that I use fairly often but forget those I rarely use; these are the perfect candidates for comments. I can never remember the order of the arguments for the `\rule` command so I type `\rul` and `Esc` twice to get the result show in in Figure (4). Another example is the `wrapfigure` environment, from the `wrapfig` package, which has multiple versions with differing numbers and positions of optional arguments. To see the variations with the comments type `bwr` on an empty line and press `Esc` to get:

```
\begin{wrapfigure}{•<placement: r,R,l,L,i,I,o,O>}{•<width>}
•
\end{wrapfigure}•
```

and versions with optional arguments on succeeding presses of `Esc`.

#### Other Environments

Environments that aren't built into the `CommandCompletion.txt` file can always be added if you use them a lot but there is an alternative for occasional use. Built into the completion algorithm is a way to complete environments. First press `\b` and `Esc` to get `\begin{`, enter the environment name and the closing `}` and then `Esc` again; the closing `\end{...}` with the corresponding environment name will be generated on a separate line.

## 4 Making Additions to `CommandCompletion.txt`

If you are adding items to the `CommandCompletion.txt` there are a few things you should know about its structure:

- Each environment has three entries: a completion that removes the leading `\begin`, i.e., it starts with a leading '`{`' and the environment name; two abbreviations that have an abbreviation name without a backslash (\) and the same abbreviation with the backslash. Commands usually have three or more forms, with and without a leading \, as well as possible abbreviations, also with and without \.

- You should add all the variations with slightly different endings for the abbreviations. I use an '`o`' at the end of an abbreviation if that variation has an optional argument, '`oo`' for two optional arguments, '`s`' for starred forms of commands, etc.

- The order of similar items in the file *does* make a dramatic difference in the order in which items are found; items placed *later* will be found *earlier* (the file is searched backwards). E.g., the order of items obtained when you press `\b` and then `Esc` depends purely on the order of matches in the `CommandCompletion.txt` file.

- For maximum convenience place a Mark[12] within each argument of commands. Surround the very first argument with two `#INS#` commands so it comes out selected. If you want to have a comment in any arguments insert a Comment Skeleton[13] and fill it in.

I'd suggest taking a look in the `CommandCompletion.txt` file for examples.

# 5 What's Missing

I'd love to be able to have the completions preserve indentation but that is not in the books for now.

Any other suggestions are welcome and will be considered for inclusion in later iterations of the Command Completion code.

12. Using `Insert Mark` (`Cmd-8`) from the `Source→Completion→Marks` menu.
13. Using `Insert Comment` (`Ctl-Cmd-8`) from the `Source→Completion→Marks` menu.

# 6 Obtaining the version of TEXShop.

The enhanced version of Command Completion is incorporated into TEXShop 2.30 and later. It is available at the TEXShop web site, <http://www.uoregon.edu/~koch/texshop/texshop.html>. Make sure you read the `Help→About This Release` document to enable the updated `CommandCompletion.txt`, etc.

THE PracTEX Journal

# Tools for creating LaTeX-integrated graphics and animations under GNU/Linux

Francesc Sunol

Abstract

This paper describes how to easily create graphics and animations that can be included in LaTeX documents. This article discusses three kinds of figures: plots, schematics, and pictures. The tools presented here can quickly generate plots, and are based on simple gnuplot and bash scripts that display the final result on the screen. Ipe is an excellent program to deal with complex figures and schematics, and the animate package is used to make a series of figures change over time to simulate a movie. All the programs used in this article are free software.

Francesc Suñol is a physics Ph.D. student at Universitat Politècnica de Catalunya, in Barcelona, Spain. Since discovering LaTeX in 2005, he has been using it for composing articles, presentations, posters, and other documents.

You can contact him by sending an email to `francesc@fa.upc.edu`, or visiting his webpage http://dfa.upc.es/personals/francesc.

- PDF version of paper
- Article source files
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/05/22

# Tools for creating LaTeX-integrated graphics and animations under GNU/Linux

Francesc Suñol

Email francesc@fa.upc.edu
Website http://dfa.upc.es/personals/francesc/
Address Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract This paper describes how to easily create graphics and animations that can be included in LaTeX documents. This article discusses three kinds of figures: plots, schematics, and pictures. The tools presented here can quickly generate plots, and are based on simple gnuplot and bash scripts that display the final result on the screen. Ipe is an excellent program to deal with complex figures and schematics, and the animate package is used to make a series of figures change over time to simulate a movie. All the programs used in this article are free software.

## 1 Introduction

It's quite common to see documents with figures that do not preserve a consistent style. Graphics that contain words or symbols must match the text in the journal or document in order to give a pleasant appearance. TeX was not originally designed for graphical work, but fortunately today we have many tools that can generate high quality figures. These generated figures can be integrated into our documents in a style consistent with the surrounding text. To give a few examples, PSTricks, PGF/Ti*k*Z, and METAPOSTare extraordinary tools that do this job well. Unfortunately, if one needs to plot a large number of points, these tools can run out of memory. In a similar way, if one needs to draw a complex figure with these tools, the task can become difficult. Other approaches are based on the removal of figure labels (nicely described by J. Levine [1]), directly hacking the `ps` or `eps` files, and rewriting the labels with the help of LaTeX packages like `psfrag` or `overpic`. I think these methods may be appropriate in some cases, but often this process can be slow and tedious.

In this article, the method proposed to automate this process is based on the use of simple bash scripts. All the tools described here are free software, and have been tested under Debian GNU/Linux. They should work in other Linux distributions as well. If you are using Windows or Mac you can try similar methods, but I think they may be more difficult to set up.

When writing scientific papers, one has to deal with at least three types of figures: plots, schematics, and pictures. Sometimes it is useful to make these figures appear to change over time (animations), for example in slideshows, or simply for teaching purposes. In the next sections I will present the tools I have developed for creating these *static* and *dynamic* plots, sketches, and pictures.

# 2 Static figures

In this section I explain step-by-step how to generate graphics files that do not change over time. I reserve the name "dynamic graphics" for those that simulate videos, that is, movies or animations, and these will be described in Section 3.

The objective of the current section is to create figures both in `eps` and `pdf` formats (in order to run `latex` or `pdflatex`) that can be included in documents using the `graphicx` package in the usual way: including the line

```
\usepackage{graphicx}
```

in the preamble, and

```
\begin{figure}[!h]
\centering
\includegraphics[options]{your-figure}
\caption{A caption.}
\label{your-label}
\end{figure}
```

in the document.

## 2.1 Plots

The best way I have found to quickly create plots is using gnuplot. Gnuplot is a very powerful program that is versatile and portable, and allows you to visualize

mathematical functions or data. I am not going to give a detailed description of how to use this program; if you are interested in the commands and screen terminals available please see the gnuplot manual by T. Williams et. al. [2].

One problem that often occurs is that after spending some time customizing your plot, when you see it on the screen the conversion to LaTeX (using `latex` or `pslatex` terminals) gives results that can be disappointing: the plot legends do not have the correct spacing, the font sizes are not appropriate, and labels are often put in the wrong place or even outside the plot. How do you deal with these problems?

I think the easiest solution is to have a file with some gnuplot commands in it, and then run a bash script in the command prompt that displays the resulting final plot on the screen. With this method you can modify the gnuplot file, run the script, and see the modifications in the plot instantaneously. If the result is not as desired, just correct the gnuplot file and rerun the script.

At its most basic level, the script should do the following:

1. *gnuplot file.gp* (redirecting the output to `pslatex terminal`).
2. *latex file.tex* (compilation of the source)
3. *dvips file.dvi* (conversion to `eps`)
4. *epstool file.eps* (creation of the bounding box)
5. *epstopdf file.eps* (conversion to `pdf`, so you can run `pdflatex`)
6. *rm auxiliary-files* (to remove auxiliary files except the original gnuplot file and the recently created `file.eps` and `file.pdf`)
7. *xpdf file.pdf* (to display the result in the screen)

I wrote a small bash script[1] (`gp2epspdf.bash`) that performs this task. You can download this script from the site where this article appears, or from my website. The script requires `latex`, `dvips`, `gnuplot`, `epstool`, `epstopdf` and `xpdf`. Assuming that you have a working LaTeX installation, you'll have `latex`, `dvips`, and probably `epstopdf` and `xpdf`. If you don't have these programs, you can install them in Debian (or any other Debian-based distribution) just by typing

```
apt-get install gnuplot epstool epstopdf xpdf
```

1. I called the script `gp2epspdf`, because it converts a gnuplot file (I usually add the extension `gp` to gnuplot files), to both an `eps` file and a `pdf` file.

at the command prompt.

Let's look at an example. Using the text editor of your choice create a file (with name `plot.gp`, for example[2]) containing the following text:

*plot.gp*

```
set size 0.75,0.65
set key 6.3,27 Left reverse samplen 1 spacing 1.2
set border 31 lw 0.5
set xlabel '$t$ (s)'
set ylabel '$y$ (m)'
set xrange [-2:6]
set yrange [-10:30]
plot -x**2+20 w p ps 1.3 pt 1 title '$y(t)=-2t^2+20$'
```

Once the file is created (and it has read permissions), just type

```
./gp2epspdf.bash plot.gp
```

in the bash prompt, and instantaneously a new window will open with Figure 1 in it. Quick and easy!

In order to see all the line styles and point types available, create a file with the word `test` in it (let's call this file `all.gp`),

*all.gp*

```
test
```

and then type

```
./gp2epspdf.bash all.gp
```

A new window will appear with the available options. In addition to these options, in gnuplot version 4.2 or later you can define your own line colors with the linecolor option.

If you find you are using this `gp2epspdf` script often, you can create an alias in your `.bashrc` file. The script is very simple, and you can easily modify it to fit your needs.

2. The name and the extension of the file are not relevant.

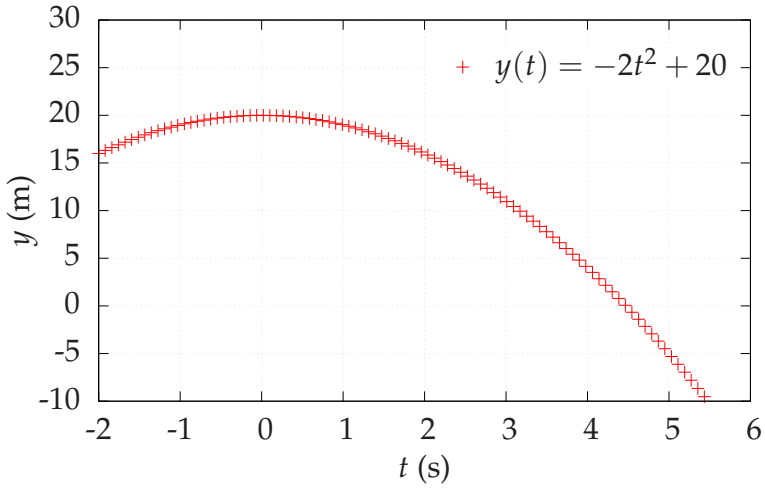



Figure 1: Example of a static plot.

## 2.2 Sketches

The creation of schematics or sketches usually involves two main approaches: on one hand you have the option to write the figure drawing commands using METAPOST, PSTricks, PGF/Ti*k*Z or others. On the other hand, you can have a visual idea of how the sketch should look, and draw it directly to paper. Personally, I am the kind of person who prefers the second option for complex figures, and I prefer to use the program Ipe in this case.

Ipe [3] is an easy-to-use drawing editor with a friendly graphical user interface. It contains a variety of basic geometry primitives like lines, splines, polygons, circles... and has snapping and grouping options, scaling and rotating features, and many more advantages. The figures can be saved in `pdf`, `eps`, or `xml` formats, and text entry is done by using LaTeX source code, which makes it easy to enter mathematical expressions. Another important point is that Ipe can accept so-called *Ipelets*, which are user-created plugins that extend the functionality of the program. O. Cheong [4] and J. Hlaveck [5] wrote intuitive manuals for learning and using Ipe.

An example of a sketch created by Ipe is shown in Figure 2.

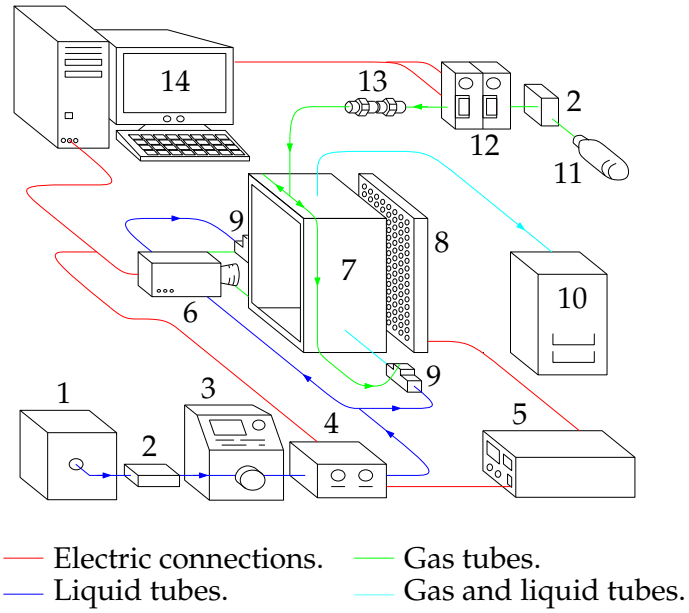



Figure 2: Example of a figure created with Ipe.

## 2.3 Pictures

Pictures are commonly used in documents. The one point I want to make here is that there are several tools in GNU/Linux to edit pictures. The most popular program is The Gimp, which has a wide range of capabilities. Another image editing software that fits my purposes perfectly is Imagemagick [6]. Imagemagick is a collection of programs to edit images, convert between file formats (more than one hundred formats supported), and similar functions. It is typically run from the command line. For example, if you have a `bmp` image and you need a `pdf`, just type

```
convert file.bmp file.pdf
```

To include labels (or whatever you want) in the pictures, you can import the figure to Ipe, place your labels, and save the file. Once again... Quick and easy!

An example of a picture containing labels is shown in Figure 3.

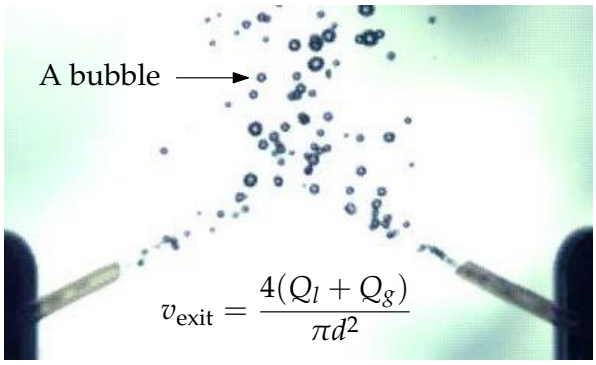



Figure 3: Example of a picture with some labels in it.

# 3 Dynamic figures: animations

Here I describe how to generate animations in LaTeX (plots, sketches, and movies) using the `animate` package [7]. It's important to note that these animations currently only work with Acrobat Reader 6 or later, and javascript should be enabled. All of the figures created in this section can be included in documents using the `animate` package by including the line

```
\usepackage{animate}
```

in the preamble, and

```
\begin{figure}[!h]
\centering
\animategraphics[options]{fps}{basename}{first}{last}
\caption{A caption.}
\label{your-label}
\end{figure}
```

in the document. The final `pdf` can be viewed in Adobe Reader on all supported platforms.

## 3.1 Plots

In order to make a plot animation, I created a bash script (`plot-animation.bash`[3]) that uses a procedure similar to the one shown in Section 2.1, inside a while loop. Mainly, it does the following:

1. *while* $i \leq N$, *plot function* $+\Delta i$ (using the same procedure as the script `gp2epspdf.bash`, but without displaying each individual plot in the screen).
2. *pdftk created-files.pdf cat output animation.pdf* (Merge all the resulting `pdf` files into one single file with `pdftk`).
3. *rm auxiliary-files* (Remove all the created `pdf` files except the one created by `pdftk`).
4. *xpdf animation.pdf* (Display the final result in the screen).

If you don't have `pdftk`, in Debian you can install it by typing

```
apt-get install pdftk
```

This script will create a single `pdf` with as many pages as the number of frames, and doesn't need an input gnuplot file; you have to edit the script in order to define what are you planning to plot. Of course, you can do the same with the script `gp2epspdf.bash`, if you don't want to have a gnuplot input file. This script is very simple, and once more you can easily modify its contents in order to fit your needs, or just to improve it.

To create the animation, type the following line in the command prompt:

```
./plot-animation.bash
```

An example of an animation created with this script is presented in Figure 4.

If you need to decrease the size of the animation, you can separate the moving parts (data points and/or functions) from the non-moving parts (axis, labels, tics, grid, ...) and use the `timeline` option of the `animate` package.

3. The script `plot-animation.bash` is available along with this article or from my website.

Figure 4: Example of a moving plot (only works with Acrobat Reader 6 or later, and javascript should be enabled!).

## 3.2 Sketches

Often it's useful to see a series of sketches that appear to change over time. Once again, Ipe can help us to do this job. The animation shown in Figure 5 has been created using Ipe as follows:

- In the first page, I draw all the lines and labels that do not change their position in time.
- In the next page, I draw the pendulum in its first position.
- In the next page, I draw the pendulum in its second position: rotated a little bit with respect to the first position.
- ...
- In the following pages I draw the ball, the spring and the moving labels.

As you can see, this process is completely manual. It works for simple animations, but not for large ones. Currently I am searching for some kind of automation, but until now I haven't found any better way to do this. Fortunately, Ipe is easy to use and it's not difficult to create large animations.

Figure 5: Example of a time-evolving sketch (only works with Acrobat Reader 6 or later, and javascript should be enabled!).

## 3.3 Movies

Including a movie in a `pdf` file can be done in several ways (e.g. using the `movie15` or the `easymovie` packages), but I prefer to use the `animate` package for two reasons:

1. The movie is embedded in the `pdf` file, so everything is contained in a single file.
2. Adobe Reader plays the animation, without calling any external programs. This is important to make your document truly portable across platforms.

First of all we need to split all the images of the movie file. In the case where we have an `avi` movie, and we want the frames in `jpg` format, this can be done by typing the command

```
mplayer movie.avi -vo jpeg
```

Normal cameras usually have a recording speed of around 30 frames per second, so if you have a movie of one minute duration, this command will create approximately 1800 images. Let's rename the images as `basename1.jpg`, `basename2.jpg`, ...and `basename1800.jpg`. Now, to merge the images in the new animation, include the following lines in your document

```
\begin{figure}[!h]
\centering
\animategraphics[options]{15}{basename}{1}{1800}
\caption{A movie caption.}
\label{movie-label}
\end{figure}
```

and the work is done. In the preceding example I set the fps to 15. The reason for this is that 15 frames per second is the average speed that the human eye can register images, and it's not necessary to force our computer to do more work than needed. The example presented would create a two-minute duration animation, since we are playing at 15 fps. If we want a one-minute animation, we can play at 30 fps. A better solution is to skip all the odd-numbered frames which significantly reduces the size of the resulting `pdf`.

An example of a movie, generated with the steps explained above (using 17 frames inside a loop, playing at 14 fps), is presented in Figure 6.

Figure 6: Example of a movie (only works with Acrobat Reader 6 or later, and javascript should be enabled!).

# 4 Summary

We have learned about some useful tools for generating graphics and animations in GNU/Linux. Three types of graphics have been discussed: plots, sketches, and pictures, both static and changing over time.

The LaTeX user community is continually growing, and along with this growth the facilities and useful packages to make high quality graphics are increasing as well. As a result, you now have to decide which of the many available tools are the best for your purposes — it's your choice.

# References


[1] Jenny Levine. *Label replacement in graphics*. The PracTEX Journal 2005-1.
http://tug.org/pracjoun/2005-1/levine

[2] Thomas Williams et. al. *Gnuplot. An interactive plotting program* (2007).
http://www.gnuplot.info/docs/gnuplot.pdf

[3] Ipe home page.
http://tclab.kaist.ac.kr/ipe/

[4] Otfried Cheong. *The Ipe manual* (2007).
http://www.cosy.sbg.ac.at/~held/teaching/wiss_arbeiten/Ipe/Ipe_manual.pdf

[5] Jan Hlavacek. *Ipe — a graphics editor for LaTeX*. The PracTEX Journal 2006-2.
http://www.tug.org/pracjourn/2006-2/hlavacek/

[6] Imagemagick home page.
http://www.imagemagick.org

[7] Alexander Grahn. *The animate package* (2008).
www.ctan.org/tex-archive/macros/latex/contrib/animate/doc/animate.pdf

THE PracTEX Journal

# An Argument for Learning LaTeX: Benefits Beyond Typesetting

Evan J. Wessler

Abstract

A friend introduced me to LaTeX, and after a short time I became a frequent user and enthusiastic proponent of this typesetting system. While there exists an abundance of pro-LaTeX literature, much of it is focused on comparing LaTeX to WYSIWYG editors and subsequently lambasting the latter. Here, I take a different approach in promoting LaTeX by shifting my focus toward recognizing several merits of learning this typesetting system. By citing personal examples, I argue for the benefits of learning LaTeX that arise apart from having the ability to typeset high quality documents.

Evan Wessler graduated from Bucknell University in 2009 with a Bachelor of Science degree in biology. While he finds biology most fascinating, he has many other interests which include astronomy, geology, physics, mathematics, history, and philosophy. He has used LaTeX for everything, from academic work to personal letters, for several years, and has grown to enjoy exploring LaTeX and promoting its merits as a hobby. In addition, Evan enjoys solving TeXnical problems by helping friends who use LaTeX to typeset their documents.

You can contact him by sending an email to evan.wessler@gmail.com

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

# An Argument for Learning LaTeX: Benefits Beyond Typesetting

Evan J. Wessler

Abstract I discovered LaTeX more or less by accident, and I could not have estimated the positive impacts that would result from learning the typesetting system. Here, I argue for the benefits of LaTeX from perspectives apart from/stemming out of typesetting.

## Introduction

As an undergraduate biology major, I had little reason and no impetus to leave the world of the WYSIWYG word processor for an advanced typesetting system. After all, the documents I was producing featured almost exclusively text, with an occasional chemical formula (e.g. $CaCl_2$) or simple mathematical equation (e.g. for linear regression analyses) used in laboratory reports. It was by chance that I was introduced to LaTeX during my sophomore year of college, when a friend who had used it to typeset a bioengineering paper happened to send me his source and output. I became immediately interested in the typesetting system, because I had recognized the appearance (i.e. the Computer Modern font and the well-formatted mathematics) as similar to that which I had seen on my calculus exams and homework sets. (Admittedly, I was always impressed with the aesthetics of these documents, and had [in retrospect, rather embarrassingly] tried to replicate their style in Microsoft Word, to no avail). I quickly learned the ins and outs of the typesetting system, and since have been a regular user, enthusiast, and unabashed proponent of LaTeX versus conventional word processing and presentation software.

Over the past three years, I have used LaTeX to produce an array of documents: analytical chemistry laboratory reports, formal business letters, physics equation sheets, charts, a symposium presentation... the list goes on. All of them were higher in quality—in terms of aesthetics—than comparable documents produced

by my peers (most of whom used Microsoft Office or Open Office) for the same purposes. However, in looking back on my experience with typesetting, I realized that there are many benefits of a *non-aesthetic* nature to learning and using LaTeX. In the remainder of this article, I will present and assess these advantages, and explain how learning the typesetting system has developed my skills, both in typesetting and beyond.

## Problem Solving

Whether one has used it for ten years or ten minutes, all users of the LaTeX typesetting system are intimately familiar with LaTeX error messages. These notifications appear when errors in the source are encountered during typesetting. An experienced user knows they can be due to a number of things, among which are incorrectly-spelled commands, missing or extraneous brackets, failure to close environments, and other errantly typed and/or conceived text in the source. However, a new user—that is, one who is new to LaTeX and has no experience in dealing with code-based, debuggable source entry (e.g. in computer programming)—will be unfamiliar with the presentation and interpretation of errors, as well as with the proper action(s) that must be taken to correct them. Moreover, this process is often non-trivial, because it is not as straightforward as the new user might assume. For example, a message may indicate that there is something wrong at a certain line, whereas the actual incorrect element may be present several lines before the stated location. In addition, the very syntax of error messages may be puzzling (bewilderment with errors during creation of complex tables comes to mind). It is also the case that some commands and environments cannot be used in tandem (e.g. the `\verb` command cannot be used as-is inside `\section{}` commands). Incompatible environments and non-global commands may be unbeknownst to the author; the consequent errors are often the source of several frustrating problems that require an advanced solution.

Thus, to be able to produce a correctly typeset document using LaTeX, one must become proficient at troubleshooting. The process may be as simple as searching a few lines for errant symbols, or trying different commands. However, a particularly perplexing error message containing ambiguous and unhelpful language may demand more creative solutions. An especially useful approach that I discovered at some point early on is what I will call the "incremental comment-

out" strategy. This involves systematically commenting-out (i.e., marking lines with the "%" symbol, so that they are ignored by the typesetting engine) different short segments of the source in succession, and attempting to typeset each time. For an error message that reveals little to no information about the location of the error, this tactic is invaluable; one will see that the document fails to typeset every time except for when the region in which the error is contained is commented-out. In this way, the error is pinpointed and can be corrected. This method may be dismissed as inefficient by those users who produce very large documents (intermittent typesetting is always advisable) and those who are more experienced and highly-versed in the nuances of LaTeX warnings; however, to the novice, this is a good way to learn about error syntax, typical problems encountered, and the methods behind locating and fixing mistakes. Of course, once an error is found, solving the problem is usually a matter of referencing any decent LaTeX manual; but in order to get to this point, the significant work of finding and understanding the error must be performed by the user. Any exercise of this nature is bound to increase one's capacity and ability to problem solve.

## Taking Command of the Command Line

Before I started using LaTeX, the command-line interface was largely unknown to me; its seemingly obscure commands and cold, intimidating appearance (it is, after all, manipulated using *just* text, which is difficult for someone like myself—who started using computers when graphical user interfaces were the norm—to accept) made it esoteric at best. I had previously dabbled in the Windows "Command Prompt" program, but had no real idea of what I was doing. (Fortunately, this did not lead to any catastrophes). Upon first introduction to LaTeX, I edited and typeset my documents in the way that most new users probably do; I wrote the source in a user-friendly GUI front-end program (in my case, Richard Koch's "TeXShop") and hit the "Typeset" button. It was only when I started reading up on LaTeX that I discovered that editing, typesetting and previewing could be accomplished within a terminal (via the use of editors such as Emacs and Vim, and issuing commands such as `latex` and `xdvi`). This discovery led me to utilize the terminal more often, and in turn to start experimenting with its other uses.

Since then, I have become fairly proficient at the command line. But this is not just for the sake of using it as a neat or exhibitionist alternative (I have

become convinced that anyone who is unfamiliar with the terminal that sees me using it thinks I am either up to no good, or that I am performing operations too sophisticated to be relevant to the "normal" computer user); I regularly use it to perform necessary functions (e.g. secure shell, efficient exploration, creation, copying and moving of directories and their contents, elementary programming, etc.). In this way, LaTeX served as a sort of "gateway" into learning to use my computer to its maximum potential. I would argue that this kind of exploration would serve as a similar boon to other users, and that it is always positive when one learns more about the workings of the technology they depend on and use frequently.

## An Appreciation for Formatting

As a burgeoning scientist, I have read countless papers and borne witness to a host of presentations that suffered from a major deficit: lack of logical formatting. It has become evident to me that this often has a significant, negative impact on the content of a scientific message. Blatantly incorrectly constructed outlines, disordered talking points, and poorly formatted section labels often turn what would be great papers and presentations into travesties of communication. Anyone who has tried to understand science knows that if ideas and data are not presented in an organized, logical fashion, they can be lost in a swirl of seemingly incomprehensible babble. The same can be said for material in other fields; it is a universal fact that ignoring logical structure can be disastrous.

That said, it is quite obvious that formatting is given little credence by most people who produce documents and give presentations. Much of the problem—as cited in so many pro-LaTeX pieces of literature—is that authors often make bad choices when detailing the aesthetic layout of their media. The consensus solution is to remove this task from the author's responsibility; this is successfully achieved by LaTeX.

Of course, an inanimate typesetting system cannot absolve one from focusing on *how* to organizing their content. But at least for me, something interesting happened upon learning LaTeX and using it for a period of time: I began not only to realize the cruciality of logical formatting, but also to *think carefully* about it. In other words, when I use LaTeX, I know I don't have to worry about the boldness or the size of my section headers; in turn, I am empowered to dedicate more

of my focus toward what I want to say, and where I want to say it. The side effects of this have transcended my use of the typesetting system. For example, when I take notes in my laboratory notebook, I notice a greater attention toward organized and systematized record-keeping; when I create a presentation, I find myself conscious of my outline, how each slide fits into it, and the efficiency with which I move between points. Thus, LaTeX has helped me gain an appreciation for logical formatting that has extended into activities which are essential for the purveyance of information.

## Synthesis

The widespread use of LaTeX has obvious explicit utilitarian impact; it has made possible the creation of well-formatted documents, whether they have significant mathematical content or otherwise. Numerous proponents of the typesetting system seem to enjoy focusing their efforts on berating conventional productivity software programs for their inefficiency, and for the aesthetic inferiority of the documents they produce. But it is not often the case that the *learning* of LaTeX is the topic of discussion. (There are a few pieces that deal with the learning curve of LaTeX, but not the process and consequences themselves). Here, I have attempted to make this my focus. I have proposed that there is significant weight to the argument that learning LaTeX not only allows you to produce great-looking documents, but also confers benefits that may not directly relate to typesetting, such as extension of problem solving skills, learning more about the technology in use, and culturing of logical planning skills.

There exists of course the potential for additional benefits to arise. Indeed, "learning" LaTeX is not a one-shot deal; rather, it is a continuous process, in which the user constantly develops his/her skills in typesetting, and as a result finds new and better ways to produce high-quality, well-formatted documents. As a high-level typesetting system, LaTeX demands curiosity, encourages tinkering, and promotes careful thinking, leading to positive developments in typesetting and beyond.

THE PracTEX Journal
TEX USERS GROUP

# Travels in TeX Land: memoir, TtH, and a booklet signature

David Walden

Abstract

In this column in each issue I muse on my wanderings around the TeX world. In this issue I describe three efforts: (1) I describe my first attempt to use the *memoir* class to produce a book; (2) I describe my first time using TtH to convert from LaTeX to HTML; and (3) I describe creating a 16-page booklet signature using a method described by another author in an earlier issue of this journal.

This will be my final TeX Land column in this journal. I am pleased to have provided a column for every previous issue, but it is now time for me to focus on other things. I won't stop using TeX, however, and probably will continue to write about TeX once in a while, but without the concerns of a regular column. I wish the editors of this journal ``all the best'' as they continue publication of *The PracTeX Journal*.

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. More information is available at www.walden-family-com. He may be contacted at dave@walden-family.com. A complete list of his pieces on TeX is at http://www.walden-family.com/public/texland/.

- PDF version of paper
- Using TtH: original LaTeX file
- PDF from LaTeX file
- HTML file from LaTeX via TtH
- Word file from HTML file
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/6/7

# Travels in TEX Land: *memoir*, T$_T$H, and a booklet signature

David Walden

Abstract In this column in each issue I have mused on my wanderings around the TEX world. In this issue I describe three efforts. In section 1, I describe my first attempt to use the *memoir* class to produce a book. In section 2 (page 6), I describe my first time using T$_T$H to convert from LATEX to HTML. In section 3 (page 9), I describe creating a 16-page booklet signature using a method described by another author in an earlier issue of this journal.

This will be my final TEX Land column in this journal. I am pleased to have provided a column for every previous issue, but it is now time for me to focus on other things. I won't stop using TEX, however, and probably will continue to write about TEX once in a while, but without the concerns of a regular column. I wish the editors of this journal "all the best" as they continue publication of *The PracTEX Journal.*

## 1 Using *memoir*

Once before I briefly tried the *memoir* class for something small — I can't remember what. As I began to draft a recent book, I decided it was time to try *memoir* for a substantial project.

### Starting out

First I downloaded a copy of the manual from

www.ctan.org/tex-archive/macros/latex/contrib/memoir/memman.pdf

and saved it in the directory of my book project for easy access, and I put the following command at the beginning of the include file for my book:

```
\documentclass[book,b5paper,showtrims]{memoir}
  %b5 = 176 x 250 mm = 6.8 x 9.8 inches
```

Note that I used *memoir's* `showtrims` option to the `\documentclass` command as I wanted to be able to see how the typeset text fit on the actual book size page. (I will drop this option just before sending the file to the printing company because I have no experience with having trim marks in a file going to a printer and printers have successfully trimmed pages from book files I sent without trim marks.)

I did nothing else except use *memoir* just as if I was using LaTeX as I drafted the books for several months.

Some time later, I added another command not available in LaTeX

```
\tightlists  %close up spacing for itemize, enumerate, etc.
```

and I went several more weeks without thinking about *memoir* again.

Finally, as I neared completion of the first draft of the book, I decided to change the text block size within the page size. I printed out a copy of chapter 6 of the *memoir* manual, glanced over it, and tried different commands that I thought should change the text black size or the margin sizes. Nothing worked. Eventually, I sought help from `comp.text.tex` and found a January 4, 2002, exchange of messages where *memoir* creator Peter Wilson reminded someone about the necessity of issuing the

```
\checkandfixthelayout
```

before the

```
\begin{document}
```

command. I did this, and then the margin and text block size commands had more reasonable effect.

However, I still could not (immediately) get the exact layout I wanted. The *memoir* system was checking for correct arithmetic or correct styling (I don't know which) and giving error messages. Since I needed to finish my first draft, I decided to not worry about the text block size and position any more at that time; I was able to achieve something closer to what I wanted, and there would be plenty of time to struggle with further refinement later.

So far, using *memoir* had not *required* much new study (although undoubtedly I would have benefitted from doing a little more study rather than doing almost none).

## Bibliography, footnotes, and references

A few weeks went by after I drafted the previous section. I had continued drafting my book and struggling with a problem that I have had with previous books and papers — how to handle footnotes, the bibliography, and references in the main text to items in the bibliography.

In my three previously published books, I have in each case bowed to the marketing wisdom or publisher's demand that footnotes not be at the bottom of pages and instead be endnotes. In the first published book, the notes are at the end of each chapter, referenced by superscript numbers in the main text, and bibliographic entries are mingled with the other notes. In my second published book, there are again notes at the end of each chapter referenced by superscript number in the main text; however, all the bibliography entries are at the back of the book in alphabetical order by first author's last name and referenced by sequence numbers in square brackets in the main text. In this case, the bibliographic entries were done with `bibtex`. In my third (self-published) book, all of the notes are at the end of the book in separate sections for each chapter with the notes referenced from the main text by superscript numbers which restart at 1 for each chapter. In this case, I created the bibliography manually (not using `bibtex`) in alphabetical order by first author's last name and used (unconventionally) the first author's last name and publication year in square brackets in the main text, e.g., [Walden98] or [Walden05b]; I like this unconventional convention quite a lot and would have used it again in my current book if I knew how to make `bibtex` do the work for me.

My decision with the current book was to put all of the notes and bibliographic entries in a single References list at the end of the book, referencing the list with superscript numbers in the main text. I looked in the *memoir* manual for how I could do this, but didn't find anything relevant and then went to `comp.text.tex`, where I found the following information from Boris Veytsman in July 1997:[1]

1. Yuri Robbers says, "I think the natbib package (an add-on for bibtex) is able to provide bibliographic references according to your unconventional convention. If not, the makebst package will definitely be able to create a `.bst` file according to that convention, and it only requires you to answer a load of questions in order to do its job."

   Yuri is undoubtedly correct. However, the approach I used didn't take much time to implement, and I suspect I was happier going in this ad hoc way than bothering to learn a new package.

> . . . to make footnotes appear as end-notes intermixed with citations is rather easy. Just create a file footnotes.bib with all your footnotes written as @misc entries like this:
>
> ```
> @Misc{footone,
>    note ={This is a footnote}}
> ```
>
> Then in the body of your document instead of the command
>
> `\footnote{This is a footnote}`
>
> put
>
> `\cite{footone}`
>
> In the list of your bibliography files put `footnotes.bib` like this:
>
> `\bibliography{physics,math,footnotes}`
>
> and you are done.
>
> If you need the labels to be superscripted instead of normal square brackets like [1], install the cite suite from CTAN and put
>
> `\usepackage{cite}`
>
> in the preamble of your document.

So I did that and it all worked except that the numbers on the items in the References list at the end of the book were still in square brackets. I looked in the `cite.sty` file and found that I needed another cite option, i.e.,

`\usepackage[super,biblabel]{cite}`.

That worked very well, until I decided to make the text of the References text be `\small`, i.e.,

```
\backmatter
\renewcommand{\bibname}{References}
\small
\raggedright
\bibliography{biblio}
```

at which point the superscripted numbers on the References list entries became almost too small to read. I read a little more of `cite.sty` and decided I needed

```
\usepackage[super]{cite}
\makeatletter
\renewcommand\@biblabel[1]{#1.}
\makeatother
```

which put unsuperscripted numbers the same size as the rest of the text in front of each entry in the References list.

Another issue had to do with URLs which were alone in the reference list. If in the main text I reference an organization's website, I often wanted to include the URL in the References list, e.g., "...CII's website`\cite{CIIurl}`" Then, in the References list I put an entry such as

```
@misc{CIIurl,
note="\url{www.ciionline.org}"
}
```

However, that had the problem that `bibtex` added a final period after the URL, and could confuse a reader. Once again I queried `comp.text.tex` and Lars Madsen responded with the following trick:

```
\def\myurl#1#2{\url{#1}}
...
@misc{CIIurl,
note="\myurl{www.ciionline.org}"
}
```

where `\myurl` uses TEX's macro calling process to throw away the extra period inserted by `bibtex`. (Lest Lars be faulted, he told me to use `\newcommand`, i.e.,

```
\newcommand\myurl[2]{\url{#1}}
```

rather than `\def`. I don't have any valid excuse for not using `\newcommand`.)

## Finishing

Another couple of weeks later, I once again felt I was nearing completion of drafting the book and it seemed like time to think about part and chapter running

headings, which I have used in my previous books done in LaTeX and not *memoir*. This seemed pretty complicated in LaTeX. From reading the manual, creating running headings seemed much simpler in *memoir*. However, after thinking about it for a little while, I decided to skip having running headings for this book, for a lighter if less informative look.

I guess I am going to seek reviews with the formatting I have now, which I am thinking will be my final formatting. I haven't really used many of *memoir's* capabilities, but the few that I have used have been easily accessible. It has been a great convenience to have the *memoir* manual readily available in the directory of my book files, and I have often gone to it.

In the previous section on "bibliography, footnotes, and references," I described how I handled the bibliography, footnotes, and references in this book. It occurs to me now that I didn't bother to look in the *memoir* manual to see if it already had a mechanism for what I wanted to do. In any case, I'm not going to make any changes in that handling at this late date.

All in all, I made the change from using LaTeX to using *memoir* with minimal effort and with less need to explicitly load additional packages. I assume that if I had used more of *memoir's* capabilities, I would have found it even more useful.

### Acknowledgments for this section

Yuri Robbers reviewed this section, made several suggestions for improvement, and caught a number of typos. Thank you, Yuri.

As always, the people past and present at `comp.text.tex` provide much support. I also enjoy the fact that MiKTeX automatically loads packages when one gives a `\usepackage` command for a package that is not already installed. Finally, I benefit every day from the packages, format, classes, etc., that members of the TeX community write or have written; I don't think often enough to thank them.

## 2 Trying T$_T$H to solve a problem

Publishers all too often want a submission to arrive as an MS Word file which a copy editor will then edit. Next in my experience, the publisher flows the content of the Word file into a typesetting program such as InDesign; or the publisher

may actually format the Word file for publication and generate a PDF file from that.

More than a decade ago I decided to stop using Word for composing documents submitted for publication, and LaTeX (along with MiKTeX) has been my system of choice for composing and formatting documents I am going to later submit for publication. However, once it is time actually to provide the document to the publisher, I am faced with converting it into Word, because that is what the publisher wants.

I have used various ad hoc methods for going from LaTeX to Word. For instance, for an 800-page book I used VisualTeX to generate HTML directly from the LaTeX for each chapter and then loaded the HTML into a Word file for each chapter. For small documents, I often have copied-and-pasted the text of my LaTeX document into Word and then manually in Word removed all the LaTeX commands and executed the Word formatting commands. For a while, I used TeX2Word (http://tug.org/pracjourn/2005-4/walden/), until I changed computers and scrapped the computer on which the program was installed.

A few months ago I again had to convert a LaTeX document into Word. I had sent the final draft to the editor of the journal as a PDF file. However, the journal's publication approach was to use a Word template to format documents for publication on the web (it is an electronic-only journal) and then output the formatted Word document as a PDF with a link from an HTML cover page in the on-line journal. I decided it was time to try one of the TeX-to-HTML conversion programs and to again get to Word's `.doc` format via HTML.

I googled for "`LaTeX to Word conversion`" and found myself at a webpage on the TUG server: http://www.tug.org/utilities/texconv/textopc.html entitled "Converters from LaTeX to PC Textprocessors — Overview." I skimmed to the section called 'HTML as intermediate format," and following the link to the homepage for `TeX4ht`. However, there the first paragraph under "Installation" said, "To be installed, the system needs a port made up of native utilities of TeX4ht and of non-native utilities. The easiest way to establish an up to date port is to download an installed distribution of the system, and upgrade it with the files provided here." That didn't sound as easy as I like installations to be (I hate configuring computer programs).

The next program listed on the TUG server page after `TeX4ht` was T$_T$H. So, I followed the link to its home page (http://hutchinson.belmont.ma.us/tth/)

and continued following the links to "T$_T$H Distribution," "link to download list," and "windows executable." Clicking on the last of these caused a `.zip` file to be saved on my computer. I unzipped it, opened the file `tth_manual.html`, read a few lines under "Usage," and tried it under Cygwin (I run Windows XP):

```
tth.exe <myfile.tex
```

I struggled for a few minutes with the fact that `tth.exe` was not being found by Windows XP (downloading T$_T$H didn't install it and update the execution search path). But eventually I conquered that problem, and the above command tried to compile my LaTeX file into HTML.[2]

However, T$_T$H complained about some of my LaTeX commands. It didn't recognize `\thinspace` and it didn't know about the `paralist` and `geometry` packages. I commented out the commands to use the *geometry* and *paralist* packages and changed the `paralist` commands to standard list commands, e.g., from `\begin{compactitem}` to `\begin{itemize}`. I also deleted my use of `\thinspace`. The publisher would reformat all this stuff anyway.

I can't remember if there were a couple of other LaTeX commands T$_T$H didn't recognize, or not. In any case, after a few such changes, T$_T$H compiled my whole LaTeX file into HTML.

However, instances of `---` failed to produce the correct thing in HTML. Rather than bothering to figure out why, I replaced the three instances of `---` with the code `*MDASH*`, which passed through the HTML phase and into Word where I did a Replace All of `*MDASH*` with an em-dash.

My writings never include significant math, so I don't know how T$_T$H would handle that. But for my non-math document, the HTML looked pretty good. I started Word, opened the HTML file from within Word, and it still looked pretty good — good enough for submission to the publisher's editing and typesetting process which would be reformatting the document according to their style. I did do a little editing of paragraph indents in the the Word file in addition to the `*MDASH*` replacements mentioned in the previous paragraph. I finished the process by saving my document as a `.doc` file from Word.

Using T$_T$H will be at the top of my list of options next time I need to go from LaTeX to HTML or Word, even if a little final editing in HTML or Word is

2. Probably the commercial version of T$_T$H (http://hutchinson.belmont.ma.us/tth/tthgold.html) avoids some or all of the problems mentioned in this and the next paragraph.

required.

For comparison, my LaTeX, PDF-from-LaTeX, and HTML-via-TtH, and Word-by-opening-HTML files are available via links on the HTML page for this note.

## 3 Creating a booklet signature

Recently I wanted to create a 16-page booklet just by folding in half 8-1/2-by-11-inch pages of paper that are normal in the United States. I remembered an article by D.V.L.K.D.P. Venugopal entitled "Creating Pocket-size Books Using LaTeX" that appeared in an issue of *The PracTeX Journal* (tug.org/pracjourn). Using the author index of the journal, I found the article in issue 2006-3 (http://www.tug.org/pracjourn/2006-3/venugopal-pocketbook).

To create my 16-page booklet, I closely followed Venugopal's first couple of steps, except using a different size sheet of paper as the starting point.

I first created a LaTeX file to produce a file called `content.pdf` containing the content of the booklet on sixteen 5-1/2-by-8-1/2-inch pages. The LaTeX file `content.tex` had the following structure:

```
\documentclass{article}
\usepackage{geometry}
\geometry{papersize={5.5in,8.5in},lmargin=.5in,%
        rmargin=.5in,tmargin=.5in,bmargin=.5in}
\begin{document}
... ... ...
\end{document}
```

Then, still following Venugopal's model, I created another LaTeX file called `signatures.tex` as follows:

```
\documentclass{article}
\usepackage[final]{pdfpages}
\begin{document}
\includepdf[pages=-,nup=1x2,landscape,signature=16]{content.pdf}
\end{document}
```

I compiled the file `signatures.tex` and then printed out the file `signatures.pdf` using my laser printer's option for printing on both sides of a sheet of paper. The pages were printed such that when I folded the resultant four sheets of paper in half, I had the desired 16-page booklet with the pages in the proper order and orientation.

Once again, an article in a back issue of *The PracTEX Journal* allowed me to copy what someone had done before rather than learning something on my own.

## Acknowledgment

Barbara Beeton caught almost a dozen typos in the abstract and three sections of a prepublication draft of this column.

## Biographical note

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. He holds an undergraduate math degree and completed a graduate school sequence of courses in computer science. More history is at www.walden-family.com/dave.

THE PracTEX Journal

# Book review: *LaTeX Quick Start* (Portuguese)

Francisco Reinaldo et al

Abstract

English

In this paper we review the book *LaTeX Quick Start: A first guide to document preparation* from a users viewpoint, and give a candid assessment of its contents.

Português

Neste artigo, nós apresentamos uma resenha sobre o livro "LaTeX Quick Start: A first guide to document preparation" sobre a perspectiva de um usuário inexperiente, dando a real idéia de como é seu conteúdo.

Francisco Reinaldo é atualmente professor na Área de Exatas no UnilesteMG, Brasil, em específico, o curso de Computação - Sistemas de Informação. No UnilesteMG, sua área de pesquisa é Inteligência Computacional (Diretor do LIC), mas dedica boa parte de seu tempo com estudos envolvendo LaTeX.

Chelder Lins Simão cursa o 3º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é estagiário do Projeto Informática Solidária - INFOSOL e colaborador do Laboratório de Inteligência Computacional – LIC, ambos os projetos sustentados pelo UnilesteMG.

Adriano Dutra cursa o 7º período do curso de graduação em Computação - Sistemas de Informação no UnilesteMG, Brasil. Atualmente é colaborador no Laboratório de Inteligência Computacional (LIC), trabalhando com Robix.

Os autores podem ser contatados através de reinaldo.opus@gmail.com

- PDF version of paper
- Article source
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/01/01

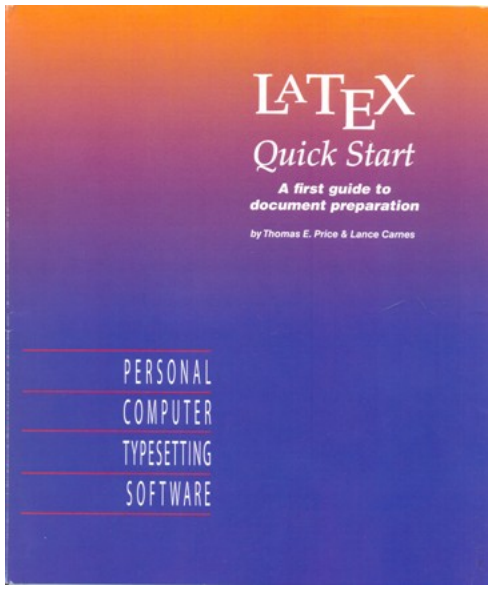
# Resenha sobre o livro LaTeX Quick Start

Francisco Reinaldo[1], Chelder Lins[2], Adriano Dutra[3]

Email [1]reinaldo.opus@gmail.com, [2]chelder.lins@gmail.com, [3]adrianoodutra@gmail.com

Abstract In this paper we review the book *LaTeX Quick Start: A first guide to document preparation* from the viewpoint of a common user, giving you a candid assessment of its contents.

## 1 Resenha de Livro




**Obra**: Thomas E. Price and Lance Carnes. LaTeX Quick Start: A first guide to document preparation. 1. ed. Personal TEX: Estados Unidos, 2009. 130 pg. http://www.pctex.com/

**Comentários**: Francisco Reinaldo, Chelder Lins, Adriano Dutra são todos membros do Laboratório de Inteligência Computacional (LIC) do Centro Universitário do Leste de Minas Gerias (UnilesteMG), Cel. Fabriciano, Minas Gerais, Brasil.

## 1.1 Sobre a obra

O Livro "LaTeX Quick Start" foi escrito para pessoas que desejam aprender a escrever artigos, livros e apresentações profissionais em LaTeX. A obra aborda desde os conceitos mais básicos da linguagem até métodos mais avançados, estando bem definida e direcionada à todos aqueles que desejam aprender LaTeX.

O conteúdo do livro divide-se em 10 capítulos e muitos apêndices com exemplos. Através de exemplos de tarefas acadêmicas para construção ou formatação de textos, os autores apresentam uma nova e fácil maneira de escrever documentos usando LaTeX. A obra é repleta de exemplos diretos e aplicados às dúvidas comuns tais como inserção de imagens, criação de cálculos matemáticos, desenvolvimento de gráficos, desenvolvimento de tabelas e outras tarefas em LaTeX.

O que nos chamou a atenção durante a leitura é a forma como os autores apresentam o conteúdo, ou seja, de forma clara, simples e eficaz, não irritando aqueles que estão com pressa para aprender LaTeX. Através desta obra, os leitores podem (a) desenvolver artigos científicos, (b) abordar partes, capítulos, seções e subseções, equações matemáticas (c) referenciar imagens, tabelas e bibliografia, (d) inserir imagens de vários tipos e até (e) importar bibliotecas de comandos; realmente tem uma infinidade de tarefas já descritas.

A obra também auxilia o usuário nas primeiras configurações do aplicativo PCTEX para utilizar a linguagem, ensinando passo a passo a sua instalação, o salvamento dos arquivos gerados e as configurações gerais necessárias. Apesar de LaTeX não ser direcionada ao aplicativo PCTeX, você também pode usar este livro para aprender LaTeX com outras ferramentas, outro ponto muito importante nesta obra. Nas últimas páginas, os autores apresentam modelos de documentos já formatados na linguagem.

Por ter qualidades tipográficas superiores, tais como letras de bom tamanho, bom espaçamento entre linhas, ilustrações de modelos prontos para auxiliam a identificação de tarefas, exemplos claros e muito bem explicados, a obra é um verdadeiro livro de cabeceira, um manual detalhado com todos os passos para

quem nunca programou em LaTeX. Parabéns aos autores.

Concluímos que o conteúdo do livro está muito bem explicado e de fácil entendimento. Realmente, é uma excelente obra para quem deseja aprender LaTeX de forma rápida e não sabe por onde começar. Os autores fizeram um excelente trabalho, pois é difícil encontrar todo o conteúdo abordado nesta obra de forma tão enxuta. Muitas vezes, o leitor precisa ler três ou quatro apostilas para complementar o conteúdo de um livro, o que não o caso deste. Usamos este livro em nosso laboratório desde seu lançamento. O único desapontamento, é que o livro é apresentado em inglês, dificultando para alguns, mas existe a possibilidade de que a segunda versão do livro seja impressa também em português.

# 2 Agradecimentos

Os autores deste artigo agradecem a Thomas E. Price e a Lance Carnes pela obra duplamente cedida para estudos em nosso laboratório.

THE PracTEX Journal

# Book review: *LaTeX Quick Start* (English)

Francisco Reinaldo et al

Abstract

In this piece we review the book *LaTeX Quick Start: A first guide to document preparation* from a user's viewpoint, and give a candid assessment. (This piece was translated from the Portuguese version.)

Francisco Reinaldo is currently a professor at Exact UnilesteMG, Brazil, in the Computing - Information Systems department. In UnilesteMG, his area of research is Computational Intelligence (Director of LIC), and devotes much of his time to studying the use of LaTeX.

Chelder Lins Simon is in the 3rd level of the undergraduate program in Computer Science - Information Systems at UnilesteMG, Brazil. He is currently a trainee in the Computing Partnership Project - INFOSOL and a collaborator with the Laboratory of Computational Intelligence - LIC, both projects supported by UnilesteMG.

Adriano Dutra is in the 7th level of the undergraduate program in Computer Science - Information Systems in UnilesteMG, Brazil. He is currently a collaborator in the Laboratory of Computational Intelligence (LIC), working with ROBIX.

Contact the authors at reinaldo.opus@gmail.com

- PDF version of paper
- Comment on this paper
- Send submission idea to editor

---

The PracTEX Journal, 2010, No. 1
Article revision 2010/05/22

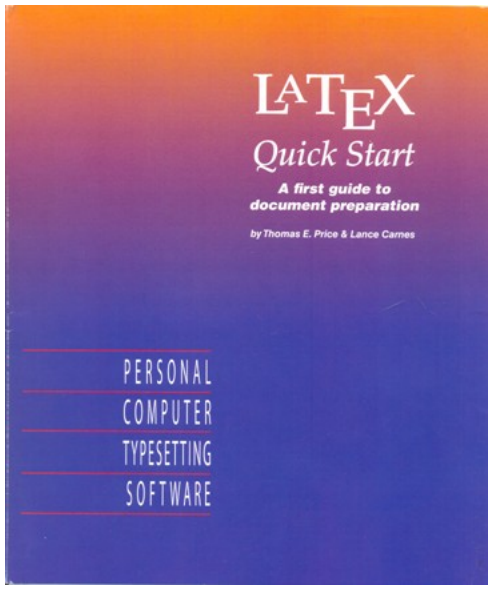
# Book review: *LaTeX Quick Start*

Francisco Reinaldo[1], Chelder Lins[2], Adriano Dutra[3]

Email [1]reinaldo.opus@gmail.com, [2]chelder.lins@gmail.com, [3]adrianoodutra@gmail.com

Abstract In this paper we review the book *LaTeX Quick Start: A first guide to document preparation* from a users viewpoint, and give a candid assessment of its contents.

## 1 Book review




**The book**: Thomas E. Price and Lance Carnes. LaTeX Quick Start: A first guide to document preparation. 1. ed. Personal TEX, Inc.: 2009. 130 pg. http://www.pctex.com/

**Comments by**: Francisco Reinaldo, Chelder Lins, and Adriano Dutra, all members of the Laboratório de Inteligência Computacional (LIC) do Centro Universitário do Leste de Minas Gerias (UnilesteMG), Cel. Fabriciano, Minas Gerais, Brasil.

## 1.1 About the book

*LaTeX Quick Start* was written for those who want to learn to write articles, books, and professional presentations in LaTeX. The book covers everything from the most basic concepts to more advanced methods. The book is well designed for anyone who wants to learn LaTeX.

The book is divided into ten chapters and several appendices, with examples throughout. By using examples of elementary tasks for building or formatting text, the authors present a new and easy way to write documents using LaTeX. The book has many examples to illustrate common needs, such as inserting images, creating mathematical formulas, developing graphics, tables, and many other LaTeX features. At the back of the book, the authors show several sample documents.

What caught our attention while reading is how the authors describe the content in a clear, simple, and effective way for those who just want to learn the basics of LaTeX. Through this book, readers can (a) develop scientific articles, (b) address parts, chapters, sections and subsections, and mathematical equations (c) reference images, tables and bibliography, (d) insert images of various kinds, and even import libraries of commands. All of these topics and more are shown by the use of sample documents.

The guide also assists the user in setting up PCTeX showing step-by-step how to install it, how to work with the generated files, and the general settings required. However, *LaTeX Quick Start* is not just for PCTeX users, and you can use this book with other LaTeX systems.

By explaining good typography, such as appropriately-sized letters, good spacing of words, and other qualities by using clear, hands-on examples, the work is a truly handy guide, and a detailed manual with everything for those who have never used LaTeX.

We found that the book explains the topics well and is easy to understand. Actually, it's an excellent book for those who want to learn LaTeX quickly and

do not know where to start. It is difficult to believe the large amount of content covered in this small book. With other guides the reader often needs to read three or four handouts to supplement the contents, but this was not the case here. We have used this book in our laboratory since it was first published.

The only disappointment is that the book is in English, making it difficult for some, and we hope the second edition will be translated into Portuguese.

THE PracTEX Journal

# Ask Nelly: How can I have the author name for a quotation set on the same line as the quotation or on a new line, according to space requirements?

The Editors

Abstract

**Ask Nelly** is a question and answer column. Nelly is the quiet person who sits at the back corner desk, who knows a lot, and when asked any question is always ready with a patient answer. If Nelly doesn't know the answer, Nelly will know an expert who has the answer. Feel free to Ask Nelly about any aspect of LaTeX, TeX, Context, etc.

- Comment on this paper
- Send submission idea to editor

TPJ

**Q:** Dear Nelly: When I write course materials for my students, I often include quotations at the beginning of a chapter. I've defined my own `Quotation` environment for that. It typesets the quotaton in a `minipage` and sets the name of the author flushed right in italics on a new line . I was wondering, however, is it possible to make LaTeX typeset the name of the author on the same line as the last words, if there is enough space, and automatically start a new line if there isn't?

**A:** There is indeed! You can define an `author` environment like this:

```
\newcommand{\author}[1]{%
    \unskip\hspace*{1em plus 1fill}%
    \nolinebreak[3]\hspace*{\hfill}\mbox{\emph{#1}}%
}
```

be sure to use `\hspace*` rather than `\hspace` because the starred version makes sure that the space you so painstakingly created doesn't immediately disappear again

if your author name is long enough to require a line of its own.

The above question was answered by **Yuri Robbers**, a member of the editorial board of this journal. He can be reached at yuri.robbers@gmail.com

TPJ

---

THE PracTEX Journal
TEX USERS GROUP

# Distractions: Typesetting a fancy curriculum vitae

The Editors

- Comment on this paper
- Send submission idea to editor

For this issue of TPJ, we selected a class for typesetting something we all need very often: a resumé. The class, written by Xavier Danaux, is called *moderncv* and can be found on CTAN. We created an example, based on the template provided by the author. Here is the pdf and the source.

Enjoy!

---

# Paul Blaga

*Resumé*

*1, Kogălniceanu Street*
*400084 Cluj-Napoca, Romania*
☎ *+40-264405300*
✉ *pablaga@cs.ubbcluj.ro*

*All changes, even the most longed for, have their melancholy; for what we leave behind us is a part of ourselves; we must die to one life before we can enter another.*
*Anatole France*

## Education

1985–1989 **BS**, *"Babeș-Bolyai" University*, Cluj-Napoca.
Bachelor Degree in Mathematics

1989–1990 **MS**, *University of Bucharest*, Bucharest.
Master Degree in Mathematics

## Master thesis

year 1990
title *Singularities in General Relativity*
supervisor Stere Ianuș

## PhD thesis

year 1997
title *Geometrical methods in General Relativity*
advisor Mircea Puta

## Experience

### Vocational

1990–1997 **Teaching Assistant**, *"Babeș-Bolyai"University*, Cluj-Napoca.
1997–2005 **Assistant Professor**, *"Babeș-Bolyai"University*, Cluj-Napoca.
2005–today **Associate Professor**, *"Babeș-Bolyai"University*, Cluj-Napoca.

### Miscellaneous

2007–2010 **Production editor**, *PracTEX*, Net.

## Languages

Romanian **Quite good** *My Mother's Tongue*
English **Average** *Studied it by myself*

| | | |
|---|---|---|
| French | **Average** | *Studied it in school* |
| Russian | **Reading** | *Studied it in school* |
| Italian | **Good** | *Studied it by myself* |

## Computer skills

| | |
|---|---|
| Computer algebra | Maple, Mathematica |
| Typesetting | LaTeX, MS-Office, Open Office |
| Programming | C++, PovRay |

## Interests

| | |
|---|---|
| Guitar | Not a very good player |
| Gardening | Without a garden |
| Computer graphics | Why not?! |

## Publications


[1] Paul A. Blaga, Horia F. Pop – *LaTeX 2$_\varepsilon$*, Editura Tehnică, București, 1999 (Romanian)

[2] Paul A. Blaga – *BRST Approaches to Gravity*, Risoprint, Cluj-Napoca, 2005

[3] Paul A. Blaga – *Lectures on Classical Differential Geometry*, Risoprint, Cluj-Napoca, 2005