THE PracTEX Journal

TEX USERS GROUP

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

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
Approx. November 20, 2007

Current Issue **2007, Number 3**
[Published 2007-08-22]

Notices

Articles

Columns

THE PracTEX Journal

From the Editor: In this issue; Next issue: Teaching LaTeX and TeX; Editorial: Tools for LaTeX and TeX Users

Francisco Reinaldo

- Comment on this paper
- Send submission idea to editor

In this issue

The theme of this issue is "Tools for LaTeX and TeX Users". This collection of articles presents several representative papers that provide the reader with an overview of current tools and methods available to easily edit LaTeX documents. As regards the amount of papers accepted, this is the biggest issue ever edited! I am delighted and deeply grateful to the authors who submitted papers answering our call for papers, and readers who sent us feedback on earlier issues. It is well known that the readers' comments are very important to the improvement of the Journal.

In this sense, several internal improvements have been made to enlarge the communication channel among editors, production editors, and authors. The improvements were extended to: papers accepted in different languages; SVN tools being used to interchange files; papers formatted on both source and screen types; and many other areas. Using Assembla http://www.assembla.com, we were able to assist authors in producing, fine-tuning and publishing their pieces of art.

In this issue we have included only those papers which we thought would guide readers to easily understandable and usable ideas. Trying to obtain both simple and practical flows of ideas, we compiled this issue in four main categories, as follows:

First Category - **Users' experiences**

Experienced users have a lot of information that may be of value to beginners in the amazing LaTeX land. In this sense, the paper presented by Theresa Song Loong reveals some of LaTeX's mystery. She shares with readers her experiences using LaTeX during her high school studies. In the same vein, Antero Neves gives us a high-level introduction on how to easily produce a LaTeX paper, using a top-down approach (this paper is in Portuguese only). In addition, Arne Henningsen shows how an educational institution can achieve valuable results when it promotes the use of LaTeX and other mainstream tools to assist in research.

Second Category - **Useful gadgets**

Currently, users have access to numerous useful WYSIWYG tools to produce all of the text and formatting needed for use in non-LaTeX documents. For some background please see this article

by Conrad Taylor. LaTeX users, of course, have a smaller set of tools they can use but these are becoming more plentiful and some are described in this issue. LaTeX packages are tools, and Arthur Buchsbaum and I chose this Journal to launch the turnstile package, a flexible tool which allows logicians to typeset the "turnstile" sign in its many forms. Wherever you go, you need a text or TeX editor, so Charilaos Skiadas and Thomas Kjosmoen present TextMate as the the most useful and intuitive editor for Mac OS X with a host of nifty LaTeX features. Vinicius Provenzano also shows how simple it can be to run Linux tools, such as a LaTeX editor called Kile, on a MAC OS environment. Jérôme Laurens presents the benefits of synchronism between PDF and TeX files when a pdfsync package is loaded. As far as graphics are concerned, Alexander Tsyplakov shows that almost everything is possible with TpX, a free source code LaTeX-oriented graphical editor. In a different vein, Duvvuri Venu Gopal presents an article that deals with the development of bibliographic databases to be used with BibTeX, which are accessible by various editors like Emacs and WinEdt and dedicated software like BibDB and TkBibTeX. Last, but not least, Fernando Sáenz-Pérez has contributed two interesting papers. The first paper shows a free, multiplatform bibliography manager (BibMgr) and citer for MS Word (Word Citer) using the BibTeX format. His second article describes another free and multiplatform tool that provides a configurable integrated development environment (ACIDE) that can be very useful for TeX users who need to use a free LaTeX editor.

Third Category - **Collaborative efforts**

Collaborative efforts using robust tools are becoming more popular but bring with them the problem of multiple versions of the same file. To address this problem, this category of articles shows some ways of solving the problem of tracking document and file versions. We received several excellent articles that shed light on how to best use these tools. Mark Eli Kalderon presents two papers on this topic. The first is a high-level paper on common problems of how to set up and use subversion, svn-multi, and LaTeX. The second one was a gift for PracTeX Journal and is a "helpdesk" paper which guides non-expert users on how to get in touch with production editors and co-authors by using SVN tools and Assembla. Additionally, Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon present a deep study of Textmate and SVN bundle. Similarly, Uwe Ziegenhagen also describes the use of svn-multi with technical details for intermediate users. Martin Scharrer shows the user's viewpoint about the main features of SVN, and stresses the use of the svn-multi package when creating a collaborative paper.

Fourth Category - **Quality**

What could possibly be said here is that TeX is simply the best.

To make it easy for readers, here is a list of tools referenced or discussed in this issue:

- Packages:
  - turnstile: Arthur Buchsbaum, Francisco Reinaldo
  - svn: Antero Neves; Arne Henningsen; Mark Eli Kalderon(Assembla)(Subversion); Martin Scharrer; Charilaos Skiadas and Thomas Kjosmoen; Uwe Ziegenhagen; Antero Neves;
  - svninfo: Arne Henningsen; Mark Eli Kalderon(Subversion); Uwe Ziegenhagen
  - svn-multi: Arne Henningsen; Mark Eli Kalderon(Subversion); Martin Scharrer; Uwe Ziegenhagen
  - pdfsync: Charilaos Skiadas and Thomas Kjosmoen; Jérôme Lauren
  - pstricks: Alexander Tsyplakov; Charilaos Skiadas and Thomas Kjosmoen;
  - pgf and tikz: Alexander Tsyplakov
  - psgreek: Theresa Song Loong
  - booktabs: Theresa Song Loong
  - longtable: Theresa Song Loong
  - array: Theresa Song Loong
  - natbib: Theresa Song Loong

    - citet: Theresa Song Loong
    - citep: Theresa Song Loong
    - graphicx: Theresa Song Loong
    - parskip: Theresa Song Loong
    - psfrag: Alexander Tsyplakov
    - overpic: Alexander Tsyplakov
- Version Control Systems:
    - Subversion: Arne Henningsen; Mark Eli Kalderon(Assembla)(Subversion); Martin Scharrer; Uwe Ziegenhagen;
    - CVS: Arne Henningsen; Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon; Martin Scharrer; Uwe Ziegenhagen
    - Git: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
    - Mercurial: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
    - SVK: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
    - Assembla Server: Charilaos Skiadas, Thomas Kjosmoen, and Mark Eli Kalderon; Mark Eli Kalderon(Assembla)
    - Client tools:
        - TortoiseSVN: Arne Henningsen; Mark Eli Kalderon(Assembla); Martin Scharrer; Uwe Ziegenhagen
        - eSvn: Arne Henningsen
        - SCPlugin: Mark Eli Kalderon(Assembla)
        - KDESVN: Martin Scharrer
        - RapidSVN: Martin Scharrer; Uwe Ziegenhagen
        - SmartSVN: Martin Scharrer
- Diff tools:
    - KDiff3: Arne Henningsen
    - wdiff: Arne Henningsen; Mark Eli Kalderon(Subversion)
    - dwdiff: Arne Henningsen
    - FileMerge: Charilaos Skiadas, Thomas Kjosmoen, and Mark Eli Kalderon; Mark Eli Kalderon(Subversion),
    - NEXTSTEP: Mark Eli Kalderon(Subversion)
    - GNU diff: Mark Eli Kalderon(Subversion)
    - opendiff: Mark Eli Kalderon(Subversion)
    - fmdiff: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon; Mark Eli Kalderon(Subversion)
- Merge tools:
    - Svnmerge: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
- PDF Viewer:
    - xpdf: Jérôme Laurens
- Text Editors:
    - Programmers Notepad 2: Theresa Song Loong
    - WinEdt: D.V.L.K.D.P. Venugopal; Fernando Sáenz-Pérez(ACIDE)
    - Emacs: D.V.L.K.D.P. Venugopal; Mark Eli Kalderon(Assembla), Jérôme Laurens, Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderonl; Vinicius Provenzano
    - Kile: Arne Henningsen; Mark Eli Kalderon(Assembla), Vinicius Provenzano
    - LEd: Mark Eli Kalderon(Assembla)
    - PCTEX: Mark Eli Kalderon(Assembla)
    - TeXShop: Mark Eli Kalderon(Assembla); Jérôme Laurens; Charilaos Skiadas and Thomas Kjosmoen; Vinicius Provenzano
    - TextMate: Mark Eli Kalderon(Assembla); Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon; Charilaos Skiadas and Thomas Kjosmoen
    - Vim: Mark Eli Kalderon(Assembla)
    - LyX: Jérôme Laurens; Fernando Sáenz-Pérez(ACIDE)
    - Scientific Word: Jérôme Laurens
    - AUCTeX: Jérôme Laurens
    - ITeXMac2: Jérôme Laurens
    - ACIDE: Fernando Sáenz-Pérez(ACIDE)

    - JEdit: Fernando Sáenz-Pérez(ACIDE)
    - WinShell: Fernando Sáenz-Pérez(ACIDE)
    - TexnicCenter: Fernando Sáenz-Pérez(ACIDE)
    - LaTeX Editor: Fernando Sáenz-Pérez(ACIDE)
    - TexMaker: Fernando Sáenz-Pérez(ACIDE); Vinicius Provenzano
    - BBEdit: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
    - Aquamacs: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
    - Carbon Emacs: Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon
    - E-Text Editor: Charilaos Skiadas and Thomas Kjosmoen
- Graphic Editors:
    - TpX: Alexander Tsyplakov
    - TeXCAD: Alexander Tsyplakov
    - jPicEdt: Alexander Tsyplakov
    - Ipe: Alexander Tsyplakov
    - LaTeX Draw: Alexander Tsyplakov
    - Inkscape: Alexander Tsyplakov
- Bibliographic Editors:
    - Ebib: D.V.L.K.D.P. Venugopal
    - Pybliographer: D.V.L.K.D.P. Venugopal
    - gBib: D.V.L.K.D.P. Venugopal
    - Barracuda: D.V.L.K.D.P. Venugopal
    - KBibTeX: D.V.L.K.D.P. Venugopal
    - Sixpack: D.V.L.K.D.P. Venugopal; Fernando Sáenz-Pérez(BIB)
    - JBibtexManager: D.V.L.K.D.P. Venugopal
    - Javabib: D.V.L.K.D.P. Venugopal
    - JabRef: Arne Henningsen; D.V.L.K.D.P. Venugopal;
    - BibDB: D.V.L.K.D.P. Venugopal
    - TkBibTeX: D.V.L.K.D.P. Venugopal
    - aux2bib: Arne Henningsen
    - WIKINDX: Arne Henningsen
    - Aigaion: Arne Henningsen
    - refBASE: Arne Henningsen
    - BibMgr: Fernando Sáenz-Pérez(ACIDE)(BIB)
    - Word Citer: Fernando Sáenz-Pérez(BIB)
    - BibShare: Fernando Sáenz-Pérez(BIB)
    - BibWord: Fernando Sáenz-Pérez(BIB)
- Books:
    - LaTeX echt einfach: Theresa Song Loong
    - Guide to LaTeX: Theresa Song Loong
    - Wikibooks. Collaborative writing of LaTeX documents: Arne Henningsen
    - svnbook.red-bean: Mark Eli Kalderon(Subversion)
    - James Edward Gray II. TextMate: Power Editing for the Mac. Pragmatic Bookshelf, 2007. : Charilaos Skiadas and Thomas Kjosmoen
    - Mike Mason. Pragmatic Version Control Using Subversion. Pragmatic Programmers LLC., 2006. : Arne Henningsen; Charilaos Skiadas and Thomas Kjosmoen;
- TeX distributions:
    - MiKTeX: Arne Henningsen; Fernando Sáenz-Pérez(ACIDE)(BIB); Martin Scharrer; Theresa Song Loong
    - codingmonkeys: Mark Eli Kalderon(Subversion)

Thank you and enjoy this issue!

Next issue

Theme: **Teaching LaTeX and TeX**
Editors: Paul Blaga and Lance Carnes

For this issue we would like to see papers from a variety of people and points of view. If you are the LaTeX expert at your university or company, how do you teach LaTeX or TeX to new users? How do you update experienced users on new techniques. How do you teach LaTeX to: students, production staff, people in the math and physics departments, people in other disciplines (social science, literature, etc.). How do you teach basic LaTeX usage, specialized LaTeX usage (e.g. advanced mathematics, critical editions), standard LaTeX usage (is there a correct way to write LaTeX documents?).

If you are a newcomer to LaTeX, how would you like to see it taught? Are the current books and guides available good or bad? How would you improve them?

If you would like to relate your teaching experiences, or voice your opinions as a user trying to learn LaTeX, please contact the editors.

Thanks

Many people have collaborated directly or indirectly to the success of this electronic journal: 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 Journal; and Allan Odgaard who kindly offered TEXTMATE gift licenses for each of us to use this amazing product; and some authors with whom friendships were formed throughout the preparation of this Journal, the most gratifying part of this work. This issue is for you!

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

Editorial: Tools for LaTeX and TeX Users

When I heard about LaTeX for the first time some years ago, I was very excited with the possibility of getting back the control of my texts. I remember that I used to type my texts in simple editors in console mode and there were no helpful tools. Today things have changed considerably and we have front-end tools, packages, internet, and a lot of gizmos to assist us with the hard task of producing good texts (.tex). In fact, it might be more difficult to explain why we do not use LaTeX 24h/day, and why we do not invite others to collaborate with us.

Professionals who write love LaTeX. The main reason is because LaTeX has exceptional facilities to work with text: book-quality typography, the ability to get the final art in PDF, legibility in any text editor. And it is completely stable and free. Therefore, in this Journal there are several papers giving their "testimonials" that LaTeX is for everyone.

When using LaTeX tools, your imagination is your limit! You can format texts, memos, letters, contracts, reports, and all sorts of documents. All you need is to be up-to-date with the right tools and packages. Having these resources at your disposal, you can do much more than the "basic" text WYSIWYG editors are offering today. In addition, graphics tools are also available. Last, but not least, our work is made easier with the use of free bibliography dataset development and management tools.

Today, we can produce fine texts in collaboration with people from different countries. Unfortunately, producing text will not be good enough if you loose control of your versions. Thus, I realised that version control systems are not being used only by programmers. In the programming field, where I include myself, we typically spend our time creating/fine tuning

codes, and then undo these changes the next day. In a broader sense, the version control utility is mostly for us, writers! People usually use computers to organise their textual information so that it can undergo constant changes. This creates a natural need for ``version control''. Subversion, a widely-used version control system, is sufficiently robust to keep a record of all changes made, and can restore any earlier version of a file or document if desired. Subversion is a tool that is being used in producing this Journal because it allows the editors to easily organise, update, and exchange all the documents and files that make up an issue.

This issue presents some compelling experiences showing that producing text with LaTeX really works, for both expert and regular users. We focus on tools that can help us work more intuitively. Thus, the scope of the issue includes tools that assist the author in preparing graphics, indexes, bibliographies, and other parts of documents; previews and PostScript/PDF manipulation tools; free or almost free tools; and cross-platform tools. Therefore, I think we have presented articles on a range of tools that can help make hard work more pleasant.

Francisco Reinaldo
Issue Editor 2007-3

THE PracTEX Journal

News from Around: Upcoming conferences in Pisa and Cluj (Romania); LaTeX workshop in Berkeley

The Editors

- Comment on this paper
- Send submission idea to editor

Upcoming events

Transylvania TeX Conference September 1-2, 2007
LaTeX workshops in Berkeley October 12, 2007
Italian TeX User Meeting October 13, 2007

The Italian TeX Users (GuIT) will be holding their annual **GuIT Meeting** on October 13, 2007 in Pisa. See the conference site for details. For a description of last year's well-attended meeting see the report by Onofrio de Bari.

(Recently, GuIT published the third annual issue of their journal, *Ars TeXnica*. Contact GuIT to order a copy of this nicely typeset journal (in Italian).)

The **Transylvania TeX Conference**, and the founding meeting of the **Romanian TeX Users (GROTeX)**, will be held September 1-2, 2007. The organizers are Paul Blaga, Zoltan Kasa and Horia Pop of the BABEŞ-BOLYAI University in Cluj, Romania. More details available at the conference site. This is a great opportunity to meet with knowledgable TeX and LaTeX users, and to visit this exotic and beautiful part of Europe.



LaTeX workshops in Berkeley A LaTeX workshop will be held October 12, 2007 at the UC Faculty Club on the University of California, Berkeley campus. This event is sponsored by PCTeX; while we prefer attendees use a PCTeX system it will be open to users of all TeX systems. The presenters are: Paulo Ney de Souza, David Auslander, and Richard Cottle. Each is an experienced teacher and LaTeX user. The topics covered will be: Introduction to LaTeX, Document structure, LaTeX graphics and tables, and Typesetting Math. The workshops will be recorded and made available for viewing later. Please see workshop information for more details.

THE PracTEX Journal

Feedback

From the Readers

- Comment on this paper
- Send submission idea to editor

TpJ

For the table with footnotes (2007-2-asknelly), I'd like to draw your attention to my ctable package, which combines the properties of threeparttable, plus those of booktabs and more. It generates better typography, has many options, and is easier to use (at least in my opinion...). Here is the source for the example table (uncomment the width option to obtain better results).

Also see the pdf output for threeparttable and for ctable without and with the width option. The ctable documentation is also included.

Wybo Dekker
Dutch TeX Users Group
http://www.ntg.nl

TpJ

Sirs: I think it might be useful if you include the source code of all your articles. The pdf version, while instructive, would be much improved if the .tex file were included.

thanks
re-v

[We have included links to the source files for most all of the articles in this issue, and we plan to do this for all future articles where the authors give their permission. - The Editors]

TpJ

Hello. I enjoyed reading this essay (Making an Electronic Journal with web tools, Wiki, and version control by Paul Blaga). I would like to share a point of experience with you.

After using a Wiki, just like you, for the management of the project cycle (tasks allocation and status, etc.), I later learned that it is worthwhile to use the Wiki also for the essays themselves, eliminating the use of Subversion. Simply, the TeX documents themselves are loaded on the Wiki as Wiki pages, and the Wiki does the role of Subversion in collaborative editing, locking, differences viewing, versioning, etc.

There are a few advantages to this mode over using Subversion:

1. One less tool to learn and maintain.

2. A web interface which allows people to work on the essays without installing a client, and over port 80 using browsers that are available in every environment.
3. More intuitive and more user-friendly.

Of course, when viewing a TeX file on the Wiki it comes out only as raw text, but this is not an issue. After all, SVN does not produce a WYSIWYG experience either. It's the keeping of versions, locks and diffs that's important.

Regards,
Hagai Bar-El
Rehovot, Israel

[Paul Blaga replies:

First of all, thank you for your interest in our work.

I don't share entirely your opinion regarding the use of Subversion. I do think that Subversion does a better job in managing different versions of the repository than the Wiki, which was designed for a slightly different purpose.

There is another reason why, personally, I wouldn't give up using Subversion. Although our journal is entirely electronic, there are some principles of publishing that have to be observed in this framework. I have in mind, first of all, confidentiality. We receive many articles for publication. Some of them are refused, and others are published in a final form which is significantly different from the initial one. Therefore I think it's natural for the authors to expect that all the editorial process remain private, between them and the editors/reviewers.

In response to the third point, we have found Subversion easy to learn and use, especially when using a GUI client.]

TpJ

Congratulations on bringing up this nice journal. In my opinon, the pdf format should be designed for screen reading using pdfscreen since it is an online journal, rather than A4 paper which is suitable for printout. If you wish you may like to have two pdf files, one for screen and the other for printing.

H.S.Rai

TpJ

Hi PracTeX Journal People,

There are a some errors on your website.

The LaTeX example given contains an error:
```
  \TPJrevision{year-mm-dd}
```
should be:
```
  \TPJrevision{year}{mm}{dd}
```

Also the manual http://tug.org/pracjourn/styles/latex/pracjourn.pdf lists the command with only one argument but in the .cls file the macro is clearly defined with three arguments. Please update your web site and documentation.

Best Regards,
Martin Scharrer

[Martin, Thanks for your excellent article in this issue, which you managed to write despite the errors in our journal style information. We have updated the web site and documentation. -The Editors]

TpJ

THE PracTEX Journal

The beginner's forest of LaTeX

Theresa Song Loong

Abstract

Images kept floating away, and keeping the style within one project both beautiful and consistent took up a lot of time. I have never been really content with a word processor. For a large project containing lots of math formulas, I assumed learning to use LaTeX would take as much time as trying to input the maths in a word processor. The final result was beautiful, but I was very wrong about how much time it would consume: using Greek fonts, making tables, using some packages, and trying to solve issues by reading incomprehensible package documentation (some of which didn't even explain how to use the package, only how it was coded) were very time-consuming indeed.

Theresa is an undergraduate student in Political Science at the Radboud University Nijmegen, the Netherlands. She started using LaTeX in 2006, when she edited a report, using a certain word processor, with someone who saw her going throug a lot of pain and horror and subsequently introduced her to LaTeX. She felt right at home with LaTeX's separation of semantical and presentational markup, but struggled with memorising the many commands and packages available. Contact her at theresasl@gmail.com

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

The Beginner's Forest of LaTeX

Theresa Song Loong

Email theresasl@gmail.com

Abstract Images kept floating away, and keeping the style within one project both beautiful and consistent took up a lot of time. I have never been really content with a word processor. For a large project containing lots of math formulas, I assumed that learning to use LaTeX would take as much time as trying to input the maths in a word processor. The final result was beautiful, but I was very wrong about how much time it would consume: using Greek fonts, making tables, using some packages, and trying to solve issues by reading incomprehensible package documentation (some of which didn't even explain how to use the package, only how it was coded) were very time-consuming indeed.

1 Introduction

The paper that I wrote with LaTeX is about the mathematical, aesthetic, and natural occurrences of the Golden Section, also known as the Greek letter φ (phi). Mathematically written it would be $\frac{1+\sqrt{5}}{2}$. In words it means "the ratio between the sum of those quantities and the larger one is the same as the ratio between the larger one and the smaller"[1]. It is already apparent that the math typesetting engine of TeX would be very beneficial to the project. But nice maths was unfortunately not the only thing I needed from LaTeX.

I started my project with "Programmers Notepad 2"[2] as text editor. This editor has a limited scheme of LaTeX-command highlights, but is very light-weight. Besides that, the friend who had suggested using LaTeX had given me two books: "*LaTeX echt einfach*"[3] (in German, but I managed), and the perhaps better known "*Guide to LaTeX*"[4]. The latter also included a CD with TeX Live 2003, which I then easily installed.

Since that same friend had once before configured the layout for another project I had done, I already knew how to use some basic commands and I used some of his work as a base for my new project:

```
\documentclass[12pt,a4paper,openany,dutch]{amsbook}
    \usepackage[Sonny]{fncychap}
    \usepackage{graphicx} % include graphics
```

```
    \usepackage{remreset}
    \usepackage{babel}
\title%
    {De ontdekking van de Phinguin\\
    {\small De natuurlijke en esthetische verschijnselen
    van de Gulden Snede}}
    \author{Theresa \and Merina}
    \date{\today}
\begin{document}
\maketitle
\tableofcontents
...
\end{document}
```

To make working together with a partner as pleasant as possible, I read some things about document organisation[4, Chapter 3, "Document Layout and Organization" which explained the book structure][5]. With the knowledge gained I organised my document as follows:

```
\frontmatter
    \setcounter{tocdepth}{0}
    \maketitle
    \tableofcontents
    \include{inleiding2}
\mainmatter
    \include{1algemeen}
    \include{2natuur}
    \include{3algoritme}
    \include{4practicum}
\backmatter
    \include{conclusie}
    \include{bronnenlijst}
\appendix
    \include{appendix}   %log of work done
    \include{appendix2}  %log of meetings with teachers
    \include{appendix3}  %source of computer program used
```

Having made a solid basis, it seemed to me my first journey in TEX Land would become a very pleasant one.

2 Using greek fonts

The Golden Section in math formulas is displayed as φ. Since it is a constant number and not a variable, it is incorrect to use the available `$\phi$`, as this gives a slanted phi. Besides, I wanted to be able to write things such as "phyllotaxis is derived from φυλον and ταξις" with a plain `\textgreek{}` command.

Achieving such things in LaTeX should be as easy as

```
\usepackage[greek,dutch]{babel}
\newcommand{\f}{\textgreek{f}}
\newcommand{\mf}{\ensuremath\mbox{\textgreek{f}}}
```

and thereafter the command `\f` or in math mode `\mf` should do the trick.

But I encountered several problems with that, the main one being my system giving me an error about "font <insert strange message here> not found". Then I analysed the error message, which in this case was `Font grmn1200 at 600 not found`.

2.1 Searching the database

If I understand correctly, if the font exists in my system, then it would have to be put into the database so that the system can find the files when called upon by pdflatex. Since the files weren't found, I rebuilt the database with the shortcut available in my Start menu[1]. This didn't work. Then I looked for my database in the `texmf` folder and opened it. I searched for "grmn1200", which I found twice under the headers `./fonts/source/public/cb/drivers:` and `./fonts/tfm/public/cb:`. So perhaps the fonts are actually available in my system and in the database, but still not found. By now, I was rather puzzled what to do.

2.2 Searching the mailing lists

There are several mailing lists for problems about using LaTeX, of which texhax[6] and comp.text.tex[7] are the most important ones. I searched the archives of these lists and found a lot of different problems, but most answers I either didn't understand or they didn't apply to me. For example, I found a problem that was very similar to mine at http://tug.org/mailman/htdig/texhax/2006-October/007160.html. The answers consisted of directions on installing

1. So basically I was using mktexlsr.

fonts or packages, which I didn't think was the problem, and finally there was a very elaborate answer which I didn't understand.

2.3 Reinstalling TEX Live

Since the "missing" files actually existed on my system, I just thought something went wrong somewhere. I reinstalled my distribution then, since I don't understand enough of the system to fix it. The problem still existed after reinstalling. I also searched the mailing lists for problems with fonts in the TEX Live 2003 distribution, but I couldn't find anything.

2.4 Installing new packages

Since the mailing list did suggest installing some other greek fonts and packages, I decided to try that. I started with the `psgreek` package, and followed the installation instructions in the documentation. Then using it as instructed, I still got the "font <strange message> not found" error, but with `greeregu at 720` as <strange message>.

Finally, I installed a package called `Ibycus4` on my system by creating a bunch of folders in my texmf-local folder, placing the correct files in correct folders and rebuilding the database. With

```
\usepackage{psibycus}
\newcommand{\f}{\textgreek{f}}
```

the intended result of `\f` becoming φ succeeded. The fonts that were missing are still missing according to my system, and updating to a newer TEX Live version is a future plan for solving it[2].

3 Tables containing a large amount of text, several columns and several pages long

Since this was a school project, it was required to append a log of what work was done when and another one with summaries of meetings with our supporting teachers. In Word these kind of tables are made easily, and I assumed the same could easily be done in LATEX. I started with a normal table, with `p{width}` as

2. According to the editor of this article, the use of the babel package works correctly in TEX Live 2007.

column formatting so that the text in a column can be several lines long. Also I used widths in `ex` so that I could assess best how much text would fit on one line.

```
\begin{tabular}[t]{|p{10ex}|p{12ex}|p{8ex}||p{25ex}||p{20ex}|}
   \hline
Date & Name & Time used & Work done & Intention\\
   \hline\hline
End of school year & Theresa and Merina & 30 minutes
& consult teacher & think about subject and how to limit the
broad angles we might encounter of the subject\\
   \hline
... & & & & \\
   \hline
\end{tabular}
```

which made this:

Date	Name	Time used	Work done	Intention
End of school year	Theresa and Merina	30 minutes	consult teacher	think about subject and how to limit the broad angles we might encounter of the subject
...				

This worked, but was ugly in several ways: the table extends to the page margins and because of the large number of columns there are large gaps of whitespace between words. Also, it was not yet possible to let the table be more than one page. Because of this, the table would float off the page if it didn't fit, leaving large amounts of vertical whitespace.

There are several packages available for LaTeX, which in small part have been covered in a previous PracTeX Journal article[8]. However, this article, which deals with a large number of packages, is confusing for a beginner, and perhaps more importantly: it did not yet exist when I was struggling with my table.

3.1 Using tabularx

To fix the problem of the table extending to the page margin and make the table, for example, as broad as `\textwidth`, the `tabularx` package provides a fairly

simple solution. I only have to `\usepackage{tabularx}` and change the first line of the table to

```
\begin{tabularx}{\textwidth}{|p{10ex}|p{12ex}|X||X||X|}
```

and the last line of course to `\end{tabularx}`, where X represents a column that may vary in length.

3.2 Using booktabs

While looking for things about tables, I stumbled upon the documentation of the `booktabs` package[9]. This manual is written in a very convincing way. It told me that I was breaking tons of styling rules by making the table as I did[3]. To solve it I should, of course, use the `booktabs` package. The thing that convinced me to do so was the fact that the manual had version number 1.61803.

The section "Use of the new commands" explained which commands you can use with the package and tried to explain how to use them. Luckily, this manual was one of the few that included an example with an example output. From this I could derive where to use which command, of which I used `\toprule`, `\midrule`, `\cmidrule{a-b}` and `bottomrule`.

3.3 Using longtable

As described earlier, the table would be several pages long, and luckily the package `longtable` is described in the *Guide to LaTeX*[4, Subsection 6.2.4, Extension packages for tables]. However, it only gave an example of code without explanation of the command syntax. The longtabs documentation also didn't describe this and I ended up studying the source code of the documentation to find out that the commands should be placed *after* each table part:

```
\begin{longtable}[t]{p{10ex}p{12ex}p{8ex}p{25ex}p{20ex}}
 \toprule
Date & Name & Time used & Work done & Intention\\
\midrule \midrule
\endhead %longtable command for general heading
\endfoot %longtable command with nothing in front,
         %since no general footer is needed
\bottomrule
\endlastfoot %longtable command for the last footer
```

3. "Never, ever use vertical rules" and "Never use double rules" were pretty clear rules.

...

Since I am used to the syntax `\command{Things the command has effect on}` this was a very confusing syntax!

With the use of longtable however I couldn't use both `longtable` and `tabularx` in one table, since there is no way to combine

```
\begin{longtable}[t]{p{10ex}p{12ex}p{8ex}p{25ex}p{20ex}}
```

and

```
\begin{tabularx}{\textwidth}{|p{10ex}|p{12ex}|X||X||X|}
```

Since having long tables took priority over the functionality of `tabularx` I only used `longtable`[4].

3.4 Using array

The last issue I dealt with was that of the large amount of whitespace between words when using `p{width}` as column formatting, since the content is automatically centred then. Instead of putting a `\raggedright` in each table cell I used the `array` package, which has a function to insert a command in an entire column, so I used `>{\raggedright}` in front of `p{width}`.

This worked very nicely, except for the fact that when I included this on the last column I got very scary errors — for those interested:

```
! Misplaced \noalign.
\midrule ->\noalign
                    {\ifnum 0=`}\fi \@aboverulesep =\aboverulesep
                                   \global \@...
l.12 \midrule
              \midrule
```

so then I just didn't include it on the last column.

The final result using the packages `longtable`, `booktabs` and `array` was

4. The editor of this article pointed out that the functionality of both packages are combined in the package `ltxtable`.

Date	Name	Time used	Work done	Intention
End of school year	Theresa and Merina	30 minutes	consult teacher	think about subject and how to limit the broad angles we might encounter of the subject
...				

by using

```
\begin{longtable}[t]{>{\raggedright}p{10ex}>{\raggedright}
p{12ex}>{\raggedright}p{8ex}>{\raggedright}p{25ex}p{20ex}}
 \toprule
Date & Name & Time used & Work done & Intention\\
\midrule \midrule
\endhead
\endfoot
\bottomrule
\endlastfoot
End of school year & Theresa and Merina & 30 minutes
& consult teacher & think about subject and how to limit the
broad angles we might encounter of the subject\\
\cmidrule{1-5}
... & & & & \\
\end{longtable}
```

4 Learning to use the bibliography

It was easy to find some tutorials on the internet explaining how to make a bibliography in LaTeX. In my earlier project, I had simply made an enumerated list, but I like things semantically correct. So at first I made this:

References

[1] GOODWIN, B.C. (1994) How the Leopard Changed Its Spots: the Evolution of Complexity. Londen: Phoenix.

[2] HAMBIDGE, J. (1924) The Parthenon and other Greek temples: their Dynamic Symmetry. New Haven: Yale University Press.

by using

```
\begin{thebibliography}{99}
   \bibitem{goo94} \textsc{Goodwin, B.C.} (1994)
How the Leopard Changed Its Spots: the Evolution of
Complexity. Londen: Phoenix.\\
   \bibitem{ham24} \textsc{Hambidge, J.} (1924)
The Parthenon and other Greek temples: their Dynamic Symmetry.
New Haven: Yale University Press.\\
\end{thebibliography}
```

however, the use of BIBTEX sounded appealing to me. It was easy enough to make a new file named `phinguin.bib` and include the bibliography, since it was explained very well by an article of the PracTEX Journal[10] by using

```
\bibliographystyle{unsrtnat}
\bibliography{phinguin}
```

and including book names with

```
@Book{ham24,
Author = {Jay Hambidge},
Title = {The Parthenon and other Greek temples: their Dynamic Symmetry},
Publisher = {Yale University Press},
Address = {New Haven},
Year = {1924},
}
```

Something that I did have a problem with was including internet links. There is no @Webpage for that sort of thing and my books mentioned nothing on the matter. I came up with the following, with `\-` indicating points where the url can be hyphenated,

```
@Book{smi03,
Author = {Smith College},
```

```
Title = {Phyllotaxis},
Year = {2003},
Publisher = {http://\-maven.\-smith.\-edu/\-~phyl\-lo/\-Con\-tact/
\-in\-dex.\-html},
}
```

but it feels very wrong.

A large benefit of all this trouble was the ability to cross-reference without thinking too much. After loading the package `natbib` the common author-year notation used in natural sciences could be used. The package offered more possibilities than just the `citet` and `citep` that I was looking for! I quickly forgot about all the commands and kept the documentation at hand. The large number of LaTeX packages and their commands can be pretty overwhelming, most of the time.

5 Maths with someone who doesn't know LaTeX

One of the best features in LaTeX is the logical structuring of a document. Since I myself have interest in the evolution of HTML[5] I am very much aware of the benefits of such structuring. However, the person that I worked with had only learnt to work with a word processor (which is very common in high school here) and actually liked working with it. I decided to relieve her from installing a LaTeX-distribution and to teach her the few basic commands we needed.

To let her check her math-equations thoroughly, I told her to use the LaTeX-previewer available on the internet[11]. I also taught her the basic commands of `\frac{a}{b}` and `\sqrt{x}`, and I tried to explain the differences between using `$ <math formula>$` and `\[ <math formula>\]`, but this was very illogical to her. In addition to that I showed that when you want to have a space after the phi, she needed to add an extra `\␣` so that it became `\f\␣`. This she did, but perhaps too frequently, because the "terminating a command by having a non-letter appear at the end" is very strange for someone who has no experience with coding.

Teaching her to write mathematics was fairly easy, since it *is* fairly easy with TeX, but teaching her how commands work and what their *semantic meaning* is, is something else. For example, she tended to use `\\␣\\␣` to make a new

5. HyperText Markup Language, which grew from a purely semantical markup language to a mixture of semantical and presentational markup, and now again to a more strict separation of presentation and meaning.

paragraph, since leaving a line open between paragraphs gave only an indent in the standard output. I told her this was *very* wrong, but I knew that this is just standard behaviour for someone who works with a word processor:

```
From $\mf\^{2}=\mf\+1$ follows that $\mf\=1+\frac{1}{\mf\}$.
If we write this out, we get:\\ \\

$\mf\ = 1 + \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}%
{1+\frac{1}{1+\frac{1}{\dots}}}}}}$\\
```

Also, the automatic numbering of equations by using the appropriate environments gave an equation number to the left[6]. She didn't like how that looked, so naturally she switched to the use of: `\frac{ES}{BS}␣=␣\frac{AE}{BD}␣\␣\␣\␣ (1)`. This of course gave inconsistent results.

The use of `\mf` for φ does not work in the previewer, so we had some problems with producing the maths. She could use `$\phi$` in the previewer, but it was at that time that she told me she had installed something called MikTEX. "Oh! But that's a LATEX-distribution!" I told her and helped her compile a document. This didn't work on her distribution, since it lacked many packages that I used in the project. I made a different LATEX-file for her which she then could compile and I could later easily incorporate in the entire project by commenting out the preamble (and "`\end{document}`") I made for her:

```
\documentclass[12pt,final,a4paper,reqn,openany,dutch]{amsbook}
\newcommand{\f}{f}
\newcommand{\F}{f}
\newcommand{\mf}{mf}
\newcommand{\mF}{mF}
\newcommand{\textgreek}[1]{#1}
\newcommand{\comment}[1]{#1}
\newcommand{\citet}[1]{#1}
\newcommand{\citep}[1]{#1}
\newcommand{\epigraph}[2]{#1,#2}
\newcommand{\epigraphwidth}{}
\newcommand{\citeauthor}[1]{#1}
\newcommand{\degree}{\ensuremath{^\circ}}
```

6. The standard output of most classes is an equation number to the right, but not with the `amsbook` class. Now I have learnt I could have simply put `reqno` as class option, but I couldn't find this option in my books or on the internet, since only `leqno` was mentioned. Then there came the strange ways of the mind into play, and I did not logically derive that I should use the option `reqno`. Notice that I did, however, try out the option `reqn` which doesn't exist, in the class options in one of the examples.

```
\usepackage{graphicx}
\begin{document}
```

Working together like this went pretty well! I did clean up her code in the end though and tried to load the `parskip` package to control the spacing of paragraphs. This gave very creepy errors[7] again, and since the *Guide to LaTeX* mentioned how to get the same effect, I copied their

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex}
```

into my preamble.

6 Conclusion

After having written this article, I can only conclude that I have learnt a lot. The many examples that I have given are pretty basic usages of LaTeX, but I have put a lot of effort in learning it. I suppose I had wished there already was a LaTeXpedia[12] with a lot of practical information.

The books *Guide to LaTeX* and *LaTeX echt einfach* proved to be my friends in need and the PracTeX Journal provided huge amounts of other information, too. The only problem with these friends is that their knowledge ends somewhere, and going beyond where these resources leave off, with the large amount of documentation to be waded through, is either too overwhelming or incomprehensible. A lot of packages document the code of the package, but not how to use it.

References

[1] Golden section. http://en.wikipedia.org/wiki/Golden_section.

[2] Programmers notepad 2. http://www.pnotepad.org/.

[3] Roland Willms. *LaTeX echt einfach*. Franzis Verlag, Poing, 2005.

[4] Helmut Kopka and Patrick W. Daly. *Guide to LaTeX*. Addison-Wesley, Boston, fourth edition, 2004.

7. I just tried to reproduce the errors, but the project now worked perfectly fine with the package, so I don't know what the problem was.

[5] Dave Walden. Travels in TEXland: LATEX for productivity in book writing. *The PracTEX Journal*, 2, 2006. http://www.tug.org/pracjourn/2006-2/walden/walden.pdf.

[6] Texhax. http://tug.org/mailman/listinfo/texhax.

[7] Comp.text.tex. http://groups.google.com/group/comp.text.tex/topics.

[8] Lapo Filippo Mori. Tables in LATEX2ε: Packages and methods. *The PracTEX Journal*, 1, 2007. http://www.tug.org/pracjourn/2007-1/mori/mori.pdf.

[9] Booktabs documentation. http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/booktabs.pdf.

[10] Jürgen Fenn. Managing citations and your bibliography with BIBTEX. *The PracTEX Journal*, 4, 2006. http://www.tug.org/pracjourn/2006-4/fenn/fenn.pdf.

[11] LATEX previewer. http://www.tlhiv.org/cgi-bin/LaTeXpreviewer/index.cgi.

[12] Lapo Filippo Mori. LATEXpedia: the future of LATEX documentation. *The PracTEX Journal*, 1, 2007. http://www.tug.org/pracjourn/2007-1/mori2/mori2.pdf.

THE PracTEX Journal

A minha experiência em LaTeX

Antero Neves

Abstract

Este artigo tem dois objectivos claros. O primeiro é partilhar a minha experiência com o LaTeX no que diz respeito a resultados e conhecimentos, e o segundo, consequência do primeiro, é facilitar o começo daqueles que agora se iniciam neste mundo.

Embora seja impossível descrever aqui todos as potencialidades presentes no LaTeX apresentam-se aquelas que julgo serem as básicas para qualquer pessoa poder começar e procuro também responder às dúvidas, que em minha opinião são mais recorrentes nos principiantes.

Para uma melhor compreensão do texto, estão presentes alguns exemplos de aplicação específica assim como de uma estrutura completa de um documento LaTeX.

Espero que com este artigo consiga dar o meu contributo para o crescimento de uma comunidade que sempre me apoiou, ajudando a eliminar o espectro da dificuldade que teima em reinar sobre o mundo do LaTeX.

O meu nome é Antero Neves, tenho 27 anos. Sou licenciado em Ensino de Matemática, uso o LaTeX desde o meu último ano de licenciatura e ainda hoje descubro as suas maravilhas nas tarefas que se me apresentam.

Considero-me perfeccionista e o LaTeX ajuda-me bastante a manter essa minha característica. Uso-o para tudo: fazer testes, fazer exposições escritas, na correspondência, em apresentações, no meu curriculum vitae. Qualquer tipo de trabalho escrito! Você pode contactar Antero no anteroneves.reg@gmail.com e visitar meu site em http://aprendolatex.wordpress.com

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/18

A minha experiência em LaTeX

Antero Neves

Eletrônico anteroneves.reg@gmail.com
Liame http://aprendolatex.wordpress.com/

Resumo Este artigo tem dois objectivos claros. O primeiro é partilhar a minha experiência com o LaTeX no que diz respeito a resultados e conhecimentos, e o segundo, consequência do primeiro, é facilitar o começo daqueles que agora se iniciam neste mundo.

Embora seja impossível descrever aqui todos as potencialidades presentes no LaTeX apresentam-se aquelas que julgo serem as básicas para qualquer pessoa poder começar e procuro também responder às dúvidas, que em minha opinião são mais recorrentes nos principiantes.

Para uma melhor compreensão do texto, estão presentes alguns exemplos de aplicação específica assim como de uma estrutura completa de um documento LaTeX.

Espero que com este artigo consiga dar o meu contributo para o crescimento de uma comunidade que sempre me apoiou, ajudando a eliminar o espectro da dificuldade que teima em reinar sobre o mundo do LaTeX.

1 Introdução

A mudança sempre nos faz pensar e quando se faz para melhor, aí é que devemos pegar com as duas mãos! É isso que eu digo a todos que querem se aventurar neste mundo que é o LaTeX. A verdade é que a dificuldade não está em saber LaTeX, mas sim em ultrapassar a diferença que existe entre LaTeX e outros.

Não podemos encarar a edição de um texto em LaTeX da mesma maneira como encaramos no MS Word pura e simplesmente porque diferem na forma de como são feitas, pois têm abordagens diferentes, têm procedimentos diferentes e claro, têm resultados diferentes - ainda bem! :-)

Com este artigo, eu venho mostrar a todos que o LaTeX é muito simples e é muito mais fácil do que aparenta à primeira vista. Primeiro, porque os procedimentos não são assim tantos como dizem as más línguas, nem tão difíceis

de entender como dizem os humanistas. Segundo porque, para além de um conjunto de livros que podemos adquirir ou encontrar em bibliotecas, existe um mar de documentação livre de custos na internet e também muitas pessoas que estão dispostas a nos ajudar. E em terceiro lugar, temos que contar também com a nossa capacidade natural de experimentar alterações de forma a corrigir erros que possam aparecer (adaptação).

Por outro lado, o uso desta tecnologia reflecte-se também na nossa conta bancária, uma vez que podemos adquirir o necessário para trabalhar a custo 0 e não precisamos mudar de computador, ou fazer upgrades de hardware dispendiosos para acompanhar a evolução do software.

2 Meu primeiro contacto com LaTeX

Comecei a escrever em LaTeX em 2005 e devo confessar que o fiz com um sentimento misto de obrigatoriedade e curiosidade. Obrigatoriedade, porque tinha que entregar a minha monografia de final de curso – Matemática – escrita em LaTeX. Curiosidade, porque embora nunca tivesse feito nenhum documento em LaTeX, já havia tido algumas conversas com colegas do meu curso e também de outros cursos ligados à informática. Todos me diziam maravilhas do "lah-teque-se" – era assim que lhe chamávamos na altura, só mais tarde descobri que se deveria dizer "Lay-Tech".

Um outro factor que fazia com que o LaTeX estivesse nas minhas boas graças era o facto de eu conseguir tirar melhores notas em todas as disciplinas que estudava com o apoio de documentos feitos em LaTeX. Isto é um facto!

Como diz o ditado: *os olhos também comem!* Ninguém gosta de estudar por documentos sem apresentação e isto nota-se muito na área da matemática onde é preciso apresentar imensos símbolos que, em documentos escritos por outros métodos, ficam com uma aparência . . . horrível!

Mas este artigo destina-se a futuros escritores, e quando concebemos um documento, umas das questões que nos assalta é: será que os meus leitores vão achar isto agradável à vista? Julgo ser consensual que um documento produzido em LaTeX tenha uma aparência muito mais profissional, o que pode ajudar qualquer um a tirar uma nota superior àquela que tiraria se apresentasse o mesmo trabalho feito de outra forma, e disso sou testemunha!

Por isso pensem bem, pode ser um pouco difícil à partida, porque é algo novo

e diferente, mas poderão perceber rapidamente esse efeito de apreciação positiva em termos estéticos. Esta é a vantagem mais imediata que encontro no LaTeX– a estética.

Depois de acabado o curso, ingressei num mestrado, também na área da Matemática, onde encontrei gente de todas as tribos, quase todos mais velhos que eu e a maior parte nunca tinham sequer visto um editor de LaTeX à frente, pois continuavam a usar as ferramentas comuns para fazer os seus trabalhos e mantinham aqueles problemas típicos, tais como problemas de formatação de fontes, de margens, de tamanho do papel, colocação de imagens, tamanho enorme dos ficheiros, compatibilidade entre versões e por aí adiante. Sempre que mudavam de computador tinham que ter tudo isso em atenção e alterar o que fosse necessário – se possível.

É verdade que hoje existem impressoras virtuais que produzem ficheiros *.pdf*, mas é um facto que o tamanho não se torna assim tão reduzido e continuamos com um problema grave de possibilidade de edição. Por exemplo, quando queremos enviar um texto a alguém para ser corrigido, aumentado ou editado a qualquer nível. É realmente terrível!

E sim, é também verdade que o documento final do LaTeX pode ser também um ficheiro *.pdf*, mas temos duas situações distintas: a primeira é o trabalho final, que podemos apresentar no referido *.pdf*, a segunda é que querermos que alguém edite o documento e para isso temos o ficheiro *.tex* gerado pelo editor de LaTeX.

Por isso, agora que estou numa fase adiantada do mestrado, uma das minhas tarefas tem sido introduzir os meus colegas ao LaTeXe também como eu, eles estão um pouco contrariados, mas é início e já serão rendidos às suas vantagens.

3 Por onde começar

3.1 Pré-requisitos

Como tudo na vida, o mais importante é compreender como as formas se encaixam e como as coisas funcionam.

Primeiro é preciso saber que no LaTeX estão envolvidos dois processos, o da edição do texto e o da compilação, isso implica a existência de um editor e um compilador.

O que se faz no editor?

Um editor de texto – compreenda-se de LaTeX– serve para colocar todo o texto que queremos que apareça no trabalho e, para além disso, colocar todas as informações de formatação necessárias. As formatações e outras especificações são fornecidas por comandos.

Hoje há vários editores de LaTeX e muitos são freeware, ou seja, podemos usa-los livremente sem pagar, o que representa uma enorme vantagem!

Programa	**Sistema Operativo**	**Tipo**	**Site na internet**
TeXnicCenter	Windows	Freeware	http://www.toolscenter.org/
TeXmaker	Vários	Freeware	http://www.xm1math.net/texmaker/
LEd	Windows	Freeware	http://www.latexeditor.org/
Kile	Linux	Freeware	http://kile.sourceforge.net/
Lyx	Linux	Freeware	http://www.lyx.org/
Vim	Vários	Freeware	http://www.vim.org/
WinEdt	Windows	Shareware	http://www.winedt.com/

Table 1: Alguns editores de LaTeX

O que faz um compilador?

O compilador "agarra" em tudo, texto e instruções dadas, e transforma-os num belo documento *.pdf*. Lindo não é?

Compilador	**Sistema**	**Site**
MikTeX	Windows	http://miktex.org/
teTeX	Linux	http://www.tug.org/tetex/
OzTeX	Mac	http://www.trevorrow.com/oztex/

Table 2: Alguns compiladores de LaTeX

3.2 Antes de começar o texto

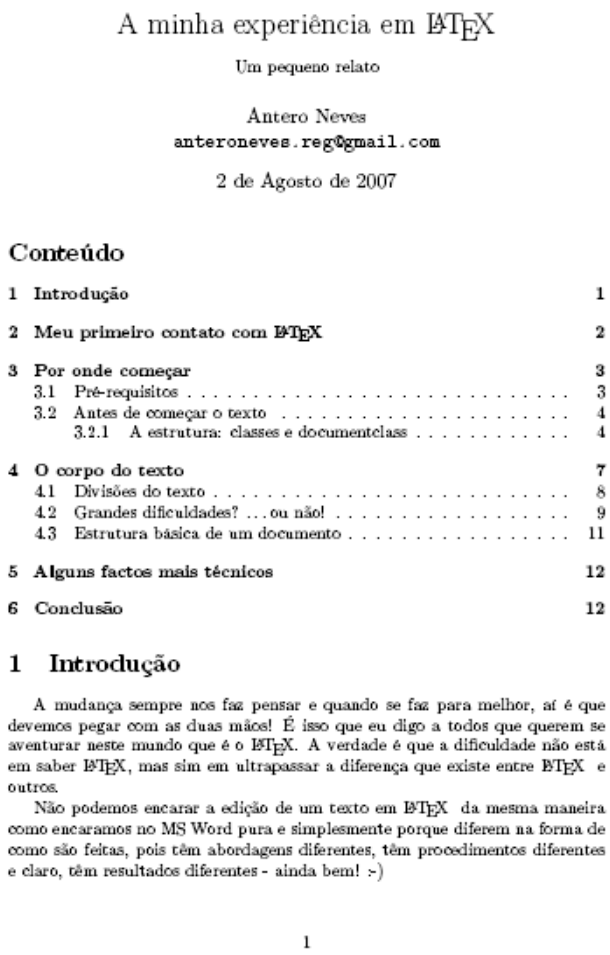
Até agora, nas apresentações que fiz sobre LaTeX, há uma coisa que convence qualquer um da superioridade deste sistema: a facilidade de formatação e como podemos configurá-la completamente!

3.2.1 A estrutura: classes e documentclass

O comando pelo qual todos os documentos LaTeX começam é:
`\documentclass{classe}`[1]
É este comando que irá definir a estrutura do documento.

As classes

Se no local onde está `classe` colocarmos `article` obtemos uma primeira página com o aspecto seguinte:



A minha experiência em LaTeX

Um pequeno relato

Antero Neves
anteroneves.reg@gmail.com

2 de Agosto de 2007

Conteúdo

1 Introdução

A mudança sempre nos faz pensar e quando se faz para melhor, aí é que devemos pegar com as duas mãos! É isso que eu digo a todos que querem se aventurar neste mundo que é o LaTeX. A verdade é que a dificuldade não está em saber LaTeX, mas sim em ultrapassar a diferença que existe entre LaTeX e outros.

Não podemos encarar a edição de um texto em LaTeX da mesma maneira como encaramos no MS Word pura e simplesmente porque diferem na forma de como são feitas, pois têm abordagens diferentes, têm procedimentos diferentes e claro, têm resultados diferentes - ainda bem! :-)

1

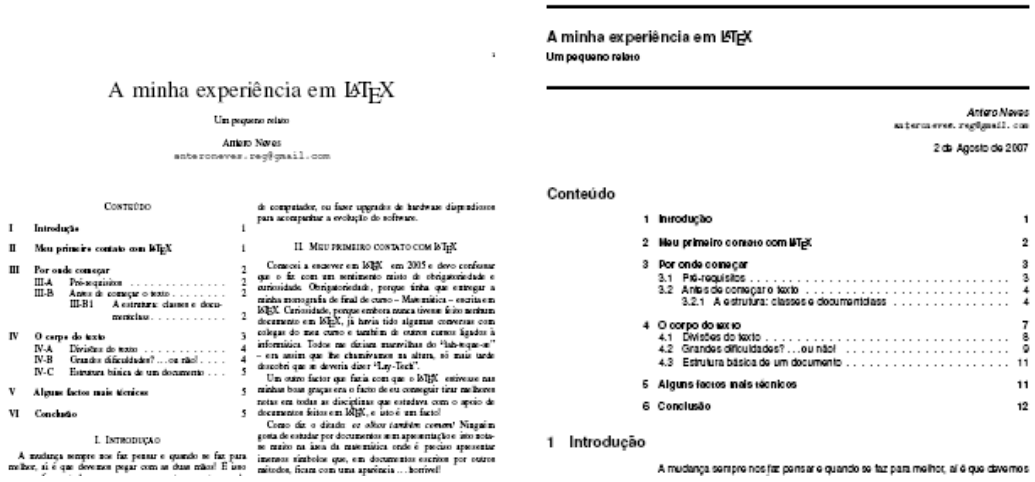
Figure 1: Primeira página de um artigo

1. Todos os comandos em LaTeXsão precedidos de \

O comando será:

`\documentclass{article}`

Mas, com a simples alteração dessa palavra podemos obter uma aparência completamente diferente, e coerente ao longo do texto.



(a) Classe `IEEEtran` (b) Classe `hitec`

Figure 2: Mais dois layouts diferentes

Deixo aqui uma pequena lista de classes para documentos:

article	report	book
paper	amsart	amsbook
amsproc	proc	coursepaper
dtk	extarticle	hitec
IEEEtran	IEEEconf	scrartcl

Para além destas, que servem para os trabalhos "convencionais", temos classes para apresentações: slides ou beamer(a minha favorita), classe para escrever cartas: letter, classes para escrever currículos vitæ: resume ou europecv, e muitas outras que desconheço!

Cada classe apresenta formatações diferentes e requisitos diferentes, por exemplo, para se escrever uma carta temos que dar ao programa informação sobre

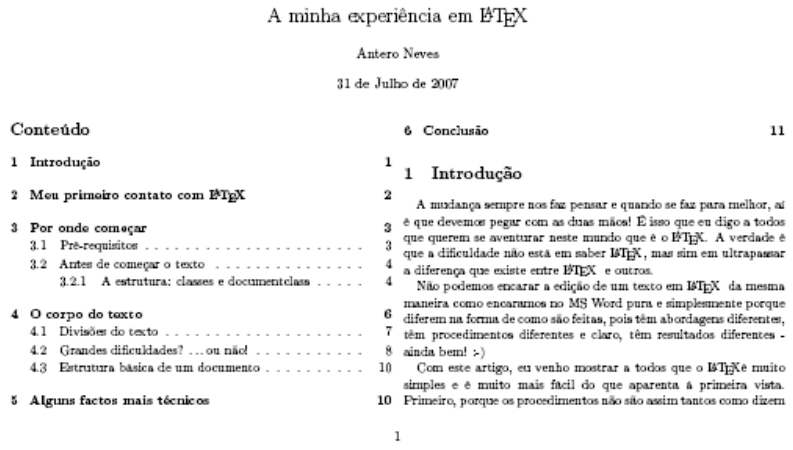
o remetente e o destinatário, só assim ele conseguirá formatar o documento, por outro lado não introduzimos numa carta capítulos ou secções.

O meu grande conselho no que toca a classes é experimentar e ver se ela se adapta às vossas necessidades e então usar. Nunca usar uma classe apenas pela beleza estética.

Os ficheiros que correspondem a classes, tem a extensão `cls`. No caso do exemplo apresentado: `article.cls`. As classes são, normalmente, instaladas com o MiKTeX, mas se assim não for, o ficheiro `.cls` deve estar no local onde se grava o ficheiro `.tex`. Pode também acontecer, aquando da compilação do programa, o compilador pedir para fazer download do ficheiro, e aí ele fica instalado no local adequado.

As opções

Podemos fazer alguns ajustes à classe para que ela se adapte às nossas necessidades.



A minha experiência em LaTeX

Antero Neves

31 de Julho de 2007

Conteúdo

1 Introdução

A mudança sempre nos faz pensar e quando se faz para melhor, aí é que devemos pegar com as duas mãos! É isso que eu digo a todos que querem se aventurar neste mundo que é o LaTeX. A verdade é que a dificuldade não está em saber LaTeX, mas sim em ultrapassar a diferença que existe entre LaTeX e outros.

Não podemos encarar a edição de um texto em LaTeX da mesma maneira como encaramos no MS Word pura e simplesmente porque diferem na forma de como são feitas, pois têm abordagens diferentes, têm procedimentos diferentes e claro, têm resultados diferentes - ainda bem! :-)

Com este artigo, eu venho mostrar a todos que o LaTeX é muito simples e é muito mais fácil do que aparenta à primeira vista. Primeiro, porque os procedimentos não são assim tantos como dizem

1

Figure 3: Artigo formatado para duas colunas e página com orientação horizontal

No exemplo acima, a classe do documento é `article`, mas são visíveis algumas alterações relativamente ao apresentado na figura 1. Para produzir os ajustamentos escrevi o seguinte:

```
\documentclass[11pt,b4paper,twocolumn,landscape]{article}
```

Outras opções que se podem colocar entre [ ] são:

Tamanho da letra 10pt, 11pt ou 12pt.

Tipo de papel `a4paper`, `letterpaper`, `a5paper`, `b5paper`, `legalpaper`, `executivepaper`.

Orientação de página `portrait` para retrato ou `landscape` para paisagem.

Capa `titlepage` para criar página de título e `notitlepage` para não criar.

Duas colunas `twocolumn`.

Equações `fleqn` – numeração de equações à esquerda; `leqno` – numeração à direita.

Impressão em dois lados `twoside` – impressão dos dois lados; `oneside` – impressão de um lado.

Primeira página dos capítulos Para obrigar a começar uma página no início de um capítulo pomos `openright`, se não o quisermos escrevemos `openany`.

A ordem pela qual se colocam as opções não interessa, o compilador interpreta-as de uma forma inteligente. Quando são omitidas as opções, a classe faz as escolhas necessárias.

3.3 Os *packages* – mais funcionalidades

Entre a instrução que define a estrutura – `\documentclass` – e o início do documento, há um rol de coisas que podemos/devemos introduzir.

Um primeiro grupo refere-se aos pacotes específicos de comandos – os *packages*. Para adicionar um determinado *package* temos que o chamar usando o comando: `\usepackage{pacote}` onde no lugar de *pacote* colocamos o nome específico do *package* a incluir no processo de compilação.

Um *package* essencial para quem escreve em português é o `inputenc.sty` que com a opção `latin1` permite a inserção de texto acentuado e ç. Chamamos o *package* digitando:

```
\usepackage[latin1]{inputenc}
```

Isso facilita muito a vida dos iniciantes.

À medida que vamos precisando de mais *packages*, vamos adicionando-os no início do documento. O seu número é infindável, sempre em crescimento e existem para quase tudo, sendo o seu grande objectivo facilitar a edição de texto e a realização de tarefas mais complicadas. Os que eu uso mais regularmente, para além do já referido `inputenc` são:

babel.sty Para colocar em português todo o texto pré-definido, como "conteúdos", "referências", "parte", etc. Uso o comando da seguinte forma:

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

indentfirst.sty Para que o primeiro parágrafo de cada divisão seja indentado.

```
\usepackage{indentfirst}
```

hyperref.sty Para fazer hiperligações no documento.

```
\usepackage{hyperref}
```

graphicx.sty Para poder incluir imagens no meu texto LaTeX.

```
\usepackage{graphicx}
```

3.4 Apresentação do trabalho e autor

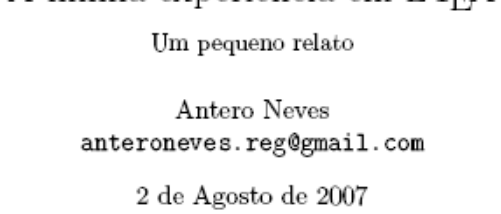


Figure 4: Apresentação do trabalho e autor

É também nesta parte que fazemos a apresentação do documento. Aí, com comandos específicos, indicamos informação sobre o autor do documento e sobre o texto em si.

Para o exemplo que se apresenta foi introduzido o seguinte código:

```
\author{Antero Neves\\
        \texttt{anteroneves.reg@gmail.com}}
\title{ A minha experiência em \LaTeX\\
        {\normalsize Um pequeno relato}}
\date{\today}
```

Os comandos essenciais são: `\author`, `\title` e `\date`.

Para que a informação apareça na primeira página do documento é necessário colocar o comando

```
\maketitle
```

no início do corpo do texto. Esta informação também é colocada automaticamente, em rodapés ou cabeçalhos, dependendo da classe.

No código encontramos várias vezes \\; usamos isto quando queremos obrigar a uma mudança de linha.

No comando `\date`, `\today` faz com que a data que aparece no documento seja a do dia da compilação do documento.

3.4.1 Tipo e tamanho de letra

O comando `\texttt` presente no código acima faz a transformação do tipo de letra para letra de máquina de escrever.

Exemplo:

```
\textbf{Texto a negrito}!
```

Texto a negrito!

Há outros tipos de letra que importa referir nesta fase, nomeadamente *itálico* e **negrito** que se obtêm da mesma forma mas usando `\emph{}` e `\textbf{}` respectivamente.

O tamanho do texto *Um pequeno relato* foi alterado usando o tamanho `\normalsize`. Os tamanhos de letra que se podem usar são: `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\huge`, `\Huge`.

Exemplo:

```
{\footnotesize Tamanho footnotesize}
```

Tamanho footnotesize

---

4 O corpo do texto

Introduzidas todas as informações podemos começar a escrever o corpo do texto.

A grande filosofia presente nesta fase da edição é: *tudo o que começa tem um fim* e é necessário dizer ao LaTeXonde as coisas começam e onde acabam. Assim, todo o texto terá que ser escrito entre o comandos `\begin{document}` e `\end{document}`.

Exemplo:

```
\begin{document}
  Corpo do texto aqui...
\end{document}
```

É portanto fácil de entender que os comandos `\begin` e `\end` são importantíssimos na edição. São eles que definem o início e o final dos chamados *ambientes*. O ambiente de tabelas, de equação, de teorema, de inserção de imagens, de lista, etc. são definidos assim.

Outra forma de delimitar acções é usando { e }, ver o exemplo de `\textbf` acima.

4.1 Índice e divisões do texto

Para que o texto apresente um índice dos conteúdos, tal como aparece na figura 1, é necessário colocar no corpo do texto o comando: `\tableofcontents`[2] É normal e aconselhável que o corpo do texto apareça dividido, para melhor se compreender a sua estrutura. Em LaTeX fazemos essas divisões através de comandos específicos muito intuitivos. São eles:

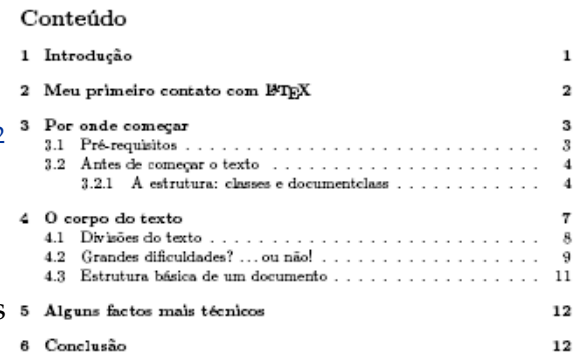


Figure 5: Tabela de conteúdos

```
\part          \subsubsection
\chapter       \paragraph
\section       \subparagraph
\subsection
```

À excepção de `\part`, todos os restantes formam uma estrutura hierárquica de seccionamento.

2. Índice de imagens: `listoffigures`
Índice de tabelas `listoftables`

A grande vantagem em usar estes comando é ficarmos totalmente livres das preocupações com formatações(tipo de letra, tamanho, negrito, itálico...), o LaTeX faz tudo automaticamente.

Existem, contudo, algumas restrições ao uso de alguns tipos de secções dependendo do tipo de documento(classe do documento), assim, se o documento for do tipo `book` ou `report`, a primeira secção permitida é `\chapter`. Já na classe `article` a hierarquia tem o seu topo em `\section`.

Como já disse o uso destes comandos é muito intuitivo e faz-se da seguinte maneira:

```
\tipo_de_secção{Título da secção}
```

Se usarmos

```
\tipo_de_secção*{Título da secção}
```

a secção não aparece numerada. Exemplo: `chapter*{A entropia}`.

4.2 Tabelas

Nome	**Publicações**
José	17
Antónia	13
Sérgio	7

```
\begin{tabular}{|l|c|}
  \hline
  \textbf{Nome}& \textbf{Publicações} \\
  \hline \hline
  José & 17 \\
  \hline
  Antónia & 13 \\
  \hline
  Sérgio & 7 \\
  \hline
\end{tabular}
```

A inserção de tabelas pode ser feita com o ambiente `tabular.sty`.

Logo na primeira linha define-se o alinhamento das colunas, usando-se `l` – alinhamento à esquerda, `c` – alinhamento ao centro e `r` – alinhamento à direita. O símbolo `|` serve para fazer os limites verticais das células da tabela e também são inseridos nessa primeira linha.

Os limites horizontais são definidos pelo comando `\hline`

Mais uma vez o final das linhas é determinado por `\\` e a mudança de colunas por `&`.

4.3 Listas – Items e enumerações e descrições

- Primeiro item
- Segundo item
- Terceiro item

```
\begin{itemize}
  \item Primeiro item
  \item Segundo item
  \item Terceiro item
\end{itemize}
```

1. Primeiro item
2. Segundo item
3. Terceiro item

```
\begin{enumerate}
  \item Primeiro item
  \item Segundo item
  \item Terceiro item
\end{enumerate}
```

Látex do Lat. latex, água nascente, líquido

```
\begin{description}
  \item[Látex] do Lat. latex, água nascente,
                 líquido
\end{description}
```

Para fazer listas temos as três hipóteses mencionadas acima. O seu funcionamento é simples, basta criar o ambiente que queremos com o `begin` e `end` e dentro dele identificar o começo de cada item com o comando `\item`.

4.4 Escrever matemática

Exemplo:

```
1 Se tivermos \(\sin(\alpha)=\frac{\sqrt{3}}{2}\)
2 sabemos que \[\alpha=\frac{\pi}{3}\] e podemos escrever então que:
3 \begin{equation}
4   \sin(\frac{\pi}{3})=\frac{\sqrt{3}}{2}
5 \end{equation}
6
```

Se tivermos $\sin(\alpha) = \frac{\sqrt{3}}{2}$ sabemos que

$$\alpha = \frac{\pi}{3}$$

e podemos escrever então que:

$$\sin(\frac{\pi}{3}) = \frac{\sqrt{3}}{2} \tag{1}$$

A inserção de caracteres matemáticos pode ser feita de várias formas, a primeira será usando `$`. Quando a inserção é feita numa linha usamos `$` para assinalar o começo e outro `$` que assinala o fim, se quisermos que a inserção seja num parágrafo à parte então usamos `$$` para o início e `$$` para finalizar. Hoje em dia, em vez dos `$` é usual `\(` no início e `\)` para finalizar, e para substituir `$$` usamos `\[` e `\]`.

Há muitos outros ambientes que permitem escrever matemática, nomeadamente ambientes multi-linhas, como matrizes.

4.4.1 Ambientes multi-linhas

Para colocar expressões que ocupam mais do que uma linha, como por exemplo definir uma função por troços, podemos recorrer ao ambiente `array.sty`.

Exemplo:

```
$\left[
\begin{array}{lcr}
a & b & c\\
d & e & f\\
g & h & i
\end{array}
\right]$
```

$$\left[\begin{array}{lcr}
a & b & c\\
d & e & f\\
g & h & i
\end{array}\right]$$

Como qualquer ambiente começamos com `\begin{array}` e terminamos com `\end{array}`. Temos também as opções para alinhar o texto: `l` – esquerda, `c` – centro, `r` – direita, e definimos uma letra para cada coluna, no exemplo temos 3 colunas então escrevemos 3 letras - lcr - isso faz com que a primeira coluna

fique alinhada à esquerda, a segunda ao centro e a terceira à direita. Se fossem 4 colunas e as quiséssemos todas alinhadas ao centro poríamos `{cccc}`.

O funcionamento deste ambiente é muito parecido ao ambiente de tabelas.

Um aspecto importante dos ambiente matemáticos reside no tamanho dos parêntesis (curvos ou rectos) e chavetas. Para que eles tenham o tamanho correcto utiliza-se `\left` e `\right` junto aos símbolos delimitadores, estando ele à esquerda ou a direita respectivamente.

Exemplo:

```
\[(\frac{\sin \pi}{3})\]
\[\left(\frac{\sin \pi}{3}\right)\]
```

$$(\frac{\sin \pi}{3})$$

$$\left(\frac{\sin \pi}{3}\right)$$

Temos também o ambiente `align`, mas este necessita do pacote `amsmath.sty`.

Exemplo:

```
\begin{align}
x &= y \\
a &= r \\
\nonumber 3 &= 5f \\
3 &= s
\end{align}
```

$$x = y \tag{2}$$
$$a = r \tag{3}$$
$$3 = 5f$$
$$3 = s \tag{4}$$

Este ambiente não necessita de ser introduzido num ambiente de matemático definido por `$` ou outro. Tem também a particularidade de estar definido com duas colunas e de o seu alinhamento estar pré-definido.

Todas as equações aparecem numeradas, se não o desejarmos podemos usar o comando `\nonumber` antes de cada equação, tal como se pode ver no exemplo.

4.5 Inserir imagens

No campo de tratamento de imagem, o LaTeX é realmente muito diferente. Principalmente porque as imagens não são inseridas no ficheiro em que escrevemos, elas apenas são referidas e depois o compilador vai buscá-las. Por isso recomendo a criação de um directório no local onde criamos o ficheiro *.tex* para assim ser mais fácil ir buscar a imagem.

As imagens podem ter diversos formatos: *.jpg*, *.png*, *.eps*, *.pdf*, devem ser os mais usados.

Como já referi neste artigo, para inserir uma imagem temos que chamar o pacote `graphicx.sty`.

Exemplo:

```
\begin{figure}[h!tbp]
  \center
  \includegraphics[width=8cm]{imagens/info.pdf}
  \caption{Imagem exemplo}
  \label{Img}
\end{figure}
```

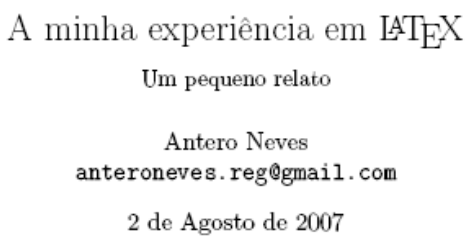


Figure 6: Imagem exemplo

A imagem do exemplo é a figura 4. Como podem verificar, temos aqui outro ambiente: `figure.sty` que facilita a colocação da imagem no local pretendido, pelo uso das opções h – colocar exactamente onde é mencionada(o sinal `!` é um reforço desta instrução), t – colocar no topo, b – colocar no fundo e p – numa página só com imagens e tabelas.

O comando `\center` centra a imagem. Se quisermos alinhar à esquerda usamos `\flushleft` e para a direita `\flushright`.

A imagem em si é colocada no texto com a linha:

`\includegraphics[width=8cm]{imagens/info.pdf}`

Há aqui alguns pontos importantes a referir. No interior dos `[]` colocamos as medidas que a figura deve ter, podemos referir a altura – height ou a largura – width, se não colocarmos nada o tamanho será o original. Caso sejam colocados os dois, convém ter em conta a opção `keepaspectratio=true/false` que forçará, se assim for necessário a que a figura não se deforme. A linha referida ficaria assim:

```
\includegraphics[height=10cm, width=8cm, keepaspectration=true]
{imagens/info.pdf}!.
```

As linhas seguintes referem-se à informação da imagem.

`\caption` define a legenda da figura.

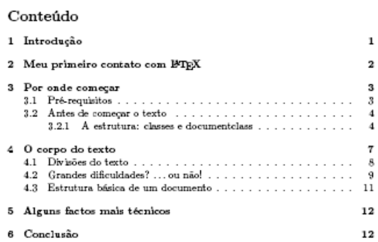
`\label` dá o nome à imagem para que ela possa ser referenciada ao longo do texto.

Para fazer uma referência usa-se o comando `\ref{}`.

4.5.1 Imagens inseridas em parágrafos

Exemplo:

```
\piccaption[\label{TOC}]{Tabela}
\pichskip{2cm}
\parpic[l]{\includegraphics[width=5cm]{imagens/tocs.pdf}}
Texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto
texto texto texto texto texto texto texto texto texto texto.
```



Conteúdo

1 Introdução 1

2 Meu primeiro contato com LaTeX 2

3 Por onde começar 3
3.1 Pré-requisitos 3
3.2 Antes de começar o texto 4
3.2.1 A estrutura: classes e documentclass 4

4 O corpo do texto 7
4.1 Divisões do texto 8
4.2 Grandes dificuldades? . . . ou não! 9
4.3 Estrutura básica de um documento 11

5 Alguns factos mais técnicos 12

6 Conclusão 12

Figure 7: Tabela

Texto texto.

Existe mais do que uma forma de inserir imagens em parágrafos, mas a que vou explanar aqui, e que proporciona o resultado acima, é para mim a mais completa em termos de opções, e a que produz melhores resultados.

Assim, devemos começar por chamar o *package* `picins.sty` e introduzir no preâmbulo, a linha:

`\usepackage{picins}`

Tal como antes, o comando `\includegraphics` coloca a imagem no documento, e as suas opções são exactamente as mesmas. Mas agora essa instrução está inserida numa outra: `\parpic`.

No exemplo dado temos a linha:

`\parpic[l]{\includegraphics[width=3.5cm]{imagens/tocs.pdf}}`

podemos ver que o comando `\parpic` vem acompanhado com uma opção `[l]`, esta opção faz com que a imagem fique na parte esquerda do parágrafo, mas há muitas outras opções que acompanham este comando.

A instrução completa é:

`\parpic(lar., alt.)(x,y)[opções][posição]{imagem}`

E as opções são:

- `lar.` e `alt.` definem a largura e altura da caixa que rodeia a imagem.
- `x` e `y` marcam o deslocamento da figura relativamente à moldura.
- Na parte seguinte podemos colocar duas opções, uma de cada grupo que se segue:
  1ºgrupo

- `l` coloca a imagem na parte esquerda do parágrafo.
- `r` coloca a imagem na parte direita do parágrafo.

2ºGrupo

- `f` - para inserir uma moldura
- `d` - inserir uma moldura a tracejado
- `o` - moldura oval
- `s` - moldura com sombra
- `x` - moldura a 3D

– Na parte da posição tratamos do alinhamento da figura na moldura com as tradicionais opções:

- `l` - esquerda
- `t` - topo
- `r` - direita
- `b` - fundo

Antes da linha `\parpic` podemos inserir vários comandos que influenciam a aparência do documento:

– `\picskip{n}`, onde `n` define o número de linhas indentadas. Por defeito elas são indentadas até ao final da imagem.

– `\pichskip{distância}`, marca a distância horizontal do texto à imagem.

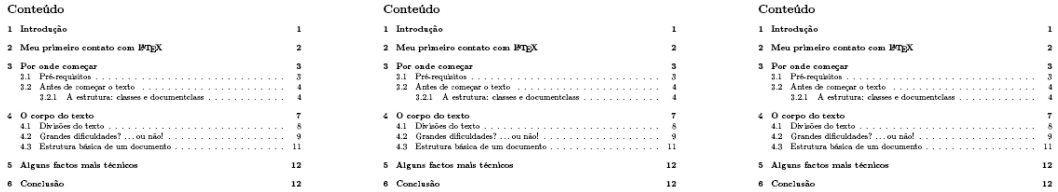
– `\piccaption[]{legenda}`, entre `[]` podemos inserir algumas opções como o nome que o programa dá à imagem usando `\label{nome}`, em `legenda` colocamos a legenda a colocar.

4.5.2 Múltiplas imagens

Há também a possibilidade de colocar imagens lado a lado. Para isso chamamos mais um pacote: `subfigure.sty`.

Exemplo:

```
\begin{figure}[h!]
  \center
  \subfigure[a][Subcaption1]{\includegraphics[width=4cm]
  {imagens/tocs.pdf}}
  \qquad
  \subfigure[b][Subcaption2]{\includegraphics[width=4cm]
  {imagens/tocs.pdf}}
  \qquad
  \subfigure[c][Subcaption1]{\includegraphics[width=4cm]
  {imagens/tocs.pdf}}
  \caption{Imagens lado a lado}
\end{figure}
```



(a) Subcaption1 (b) Subcaption2 (c) Subcaption1

Figure 8: Imagens lado a lado

Então, para começar, colocamos no preâmbulo a linha:

`\usepackage{subfigure}`

Usamos depois o comando `\subfigure` com as opções devidas, mas dentro do ambiente `\figure`.

A instrução completa é:

`\subfigure[ref1][Legenda1]{\includegraphics[width=5cm]{caminhoimagem1}}`

A explicação deste comando é simples.

- `ref1` - nome que a subfigura terá no documento.
- `Legenda1` - legenda que a subfigura terá no documento.

– por fim usamos o comando `\includegraphics` como foi explicado anteriormente.

As opções `ref1` e `Legenda1` são facultativas.
Por último coloca-se a legenda geral usando:

```
\caption{Imagens lado a lado}
```

também esta linha é facultativa.

O comando `\qquad`, usado no exemplo, controla o espaçamento entre as imagens. Podemos recorrer a outros comandos para fazer outro tipo de espaços, como `\quad` ou `\mbox{}`.

5 Alguns factos mais técnicos

Muitos de nós têm ferramentas de edição de texto que já vêm instaladas quando compramos um computador, o tempo avança e muitas vezes é necessário actualizar o software, a verdade é que essas actualizações ficam caras, e muitas vezes não é só o preço do programa em si, mas também do hardware porque os programas exigem mais espaço, mais capacidade de processamento e mais RAM. A RAM necessária para algumas ferramentas, ou pelo menos para activar todos os seus recursos, atinge hoje 1GB!!! E o processador só é mais usado pelo LaTeX aquando da compilação, enquanto que as outras ferramentas necessitam dele continuamente.

Há uns anos atrás fazíamos os mesmos trabalhos em computadores com um terço das capacidades dos computadores de hoje. E eu deixo-vos apenas uma questão: afinal em que é que os trabalhos actuais são melhores do que os que se faziam há uns anos atrás?

6 Conclusão

Com a crescente exigência por parte de professores, colegas de trabalho e da sociedade geral, torna-se essencial uma produção de textos científicos de qualidade. O LaTeX é um enorme contributo para a melhoria dessa qualidade relativamente a ferramentas de edição de texto "tradicionais" em especial na área da Matemática.

Parece-me que a maioria das pessoas que vê um documento concebido em LaTeX pensa imediatamente que ele está bem feito e que fazer uma coisa igual deve ser muito difícil. Nada mais errado! Trabalhar com LaTeX não é difícil, é sim diferente daquilo a que fomos habituados, e essa diferença faz com que seja fundamental a compreensão do mecanismo e da filosofia reinantes. Neste artigo procurou-se explicar de forma simples estes processos.

Tentou-se também facilitar o começo de alguns curiosos e iniciantes do mundo LaTeX pela apresentação de alguns procedimentos básicos, algumas opções nem sempre óbvias e pela apresentação da estrutura de um documento LaTeX.

Espero ter atingido esses objectivos e que mais gente se renda definitivamente ao LaTeX depois de ler estas linhas.

THE PracTEX Journal

Tools for Collaborative Writing of Scientific LaTeX Documents

Arne Henningsen

Abstract

Collaborative writing of documents requires a strong synchronisation among authors. This paper describes a possible way to organise the collaborative preparation of scientific LaTeX documents. The presented solution is primarily based on the version control system "Subversion". The paper describes how "Subversion" can be used together with several other software tools and LaTeX packages to organise the collaborative preparation of LaTeX documents.

Arne Henningsen received his PhD in Agricultural Economics in 2006. Currently, he is a post-doc researcher and lecturer at the Department of Agricultural Economics, University of Kiel, Germany. His main interests are microeconomic modelling and econometrics. He started using LaTeX in 2001 and gradually turned into an enthusiastic LaTeX user. He has written a few LaTeX classes and BibTeX styles for scientific journals in economics that are available in his "economic" bundle at CTAN.

You can reach Arne at ahenningsen@agric-econ.uni-kiel.de

http://www.uni-kiel.de/agrarpol/ahenningsen/index-e.html

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/28

Tools for Collaborative Writing of Scientific LaTeX Documents

Arne Henningsen

Abstract Collaborative writing of documents requires a strong synchronisation among authors. This paper describes a possible way to organise the collaborative preparation of scientific LaTeX documents. The presented solution is primarily based on the version control system **Subversion**. The paper describes how **Subversion** can be used together with several other software tools and LaTeX packages to organise the collaborative preparation of LaTeX documents.

1 Introduction

Many scientific articles, reports, and books are written by more than one author. The collaborative preparation of documents requires a considerable amount of coordination among the authors. This coordination can be organised in many different ways, where the best way depends on the specific circumstances.

In this paper, I describe how the collaborative writing of LaTeX documents is organised at our department[1]. I present our software tools, and describe how we use them. Thus, this paper provides some ideas and hints that will be useful for other LaTeX users who prepare documents together with their co-authors.

2 Interchanging Documents

There are many ways to interchange documents among authors. One possibility is to compose documents by interchanging e-mail messages. This method has the advantage that common users generally do not have to install and learn the usage of any extra software, because virtually all authors have an e-mail account.

1. Division of Agricultural Policy, Department of Agricultural Economics, University of Kiel, Germany.

Furthermore, the author who has modified the document can easily attach the document and explain the changes by e-mail as well. Unfortunately, there is a problem when two or more authors are working, at the same time, on the same document. So, how can authors synchronise these files?

A second possibility is to provide the document on a common file server, which is available in most departments. The risk of overwriting each others' modifications can be eliminated by locking files that are currently edited. However, generally the file server can be only accessed from within a department. Hence, authors, who are out of the building, cannot use this method to update/commit their changes. In this case, they will have to use another way to contour this problem. So, how can authors access these files?

A third possibility is to use a version control system. A comprehensive list of version control systems can be found at [32]. Version control systems keep track of all changes in files in a project. If many authors modify a document at the same time, the version control system tries to merge all modifications automatically. Only if two (or more) authors have modified the same line, the modifications cannot be merged automatically, but the user has to resolve this "conflict" by deciding manually, which of the two changes should be kept. Authors can also comment their modifications so that the co-authors can easily understand the workflow of this file. As version control systems generally communicate over the internet (e.g. through TCP/IP connections), they can be used from different computers with internet connection.[2] The internet is only used for synchronising the files. Hence, a permanent internet connection is not required. The only drawback of a version control system could be that a it has to be installed and configured.[3]

2. A restrictive firewall policy might prevent the version control system from connecting to the internet. In this case, the network administrator has to be asked to open the appropriate port.

3. A version control system is useful even if a single user is working on a project. First, the user can track (and possibly revoke) all previous modifications. Second, this is a convenient way to have a backup of the files on other computers (e.g. on the version control server). Third, this allows the user to easily switch between different computers (e.g. office, laptop, home).

3 The Version Control System Subversion

At our department, we decided to use the open source version control system **Subversion** [26]. This software is considered as an improvement of the popular version control system **CVS**. The **Subversion** (**SVN**) version control system is based on a central **Subversion** server that hosts the "repositories".[4] Each user has a local "working copy" of (a part of) a remote "repository". For instance, users can "update" changes from the repository to their working copy, "commit" changes from their own working copy to the repository, or (re)view the differences between working copy and repository.

To set up a **Subversion** version control system, the **Subversion** *server* software has to be installed on a (single) computer with permanent internet access.[5] It can run on many Unix, modern MS Windows, and Mac OS X platforms.

Users do not have to install the **Subversion** *server* software, but a **Subversion** *client* software. This is the unique way to access the "repositories" on the server. Besides the basic **Subversion** command-line client, there are several Graphical User Interface Tools (GUIs) and plug-ins for accessing the **Subversion** server (see [27]). Additionally, there are very good manuals about **Subversion** freely available on the internet (e.g. [3]).

At our department, we run the **Subversion** server on a **GNU-Linux** system, because most **Linux** distributions include it. In this sense, installing, configuring, and maintaining **Subversion** is a very simple task.

Most MS Windows users access the **Subversion** server by the **TortoiseSVN** client [29], because it provides the most usual interface for common users. Linux users usually use the **Subversion** command-line client or **eSvn** GUI [15] with **KDiff3** [7] for showing complex differences.

4. A Repository can be thought of as a library, where authors keep successive revisions of one or more documents. The version control systems acts as the librarian between the author and the repository. For instance, the authors can ask the librarian to get the latest version of their projects or to commit a new version to the librarian. [5, modified]

5. If this computer has no static IP address, one can use a service like DynDNS [6] to be able to access the server with a static hostname.

4 Hosting LaTeX files in Subversion

On our **Subversion** server, we have one repository for a common `texmf` tree. Its structure complies with the "TeX Directory Structure" guidelines (TDS, [21], see figure 1). This repository provides LaTeX classes, LaTeX styles, and BibTeX styles

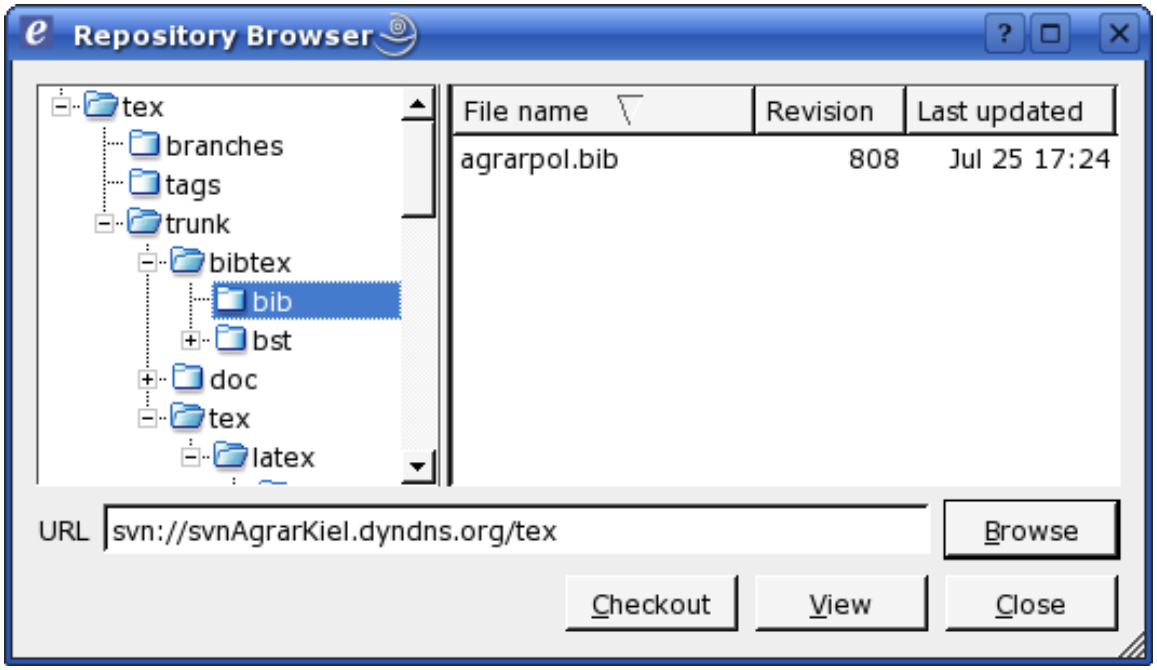


Figure 1: Common `texmf` tree shown in **eSvn**'s Repository Browser

that are not available in the LaTeX distributions of the users, e.g. because they were bought or developed for the internal use at our department. All users have a working copy of this repository and have configured LaTeX to use this as their personal `texmf` tree.[6] If a new class or style file has been added (but not if these files have been modified), the users have to update their "file name data base" (FNDB)

6. For instance, teTeX [8] users can edit their TeX configuration file (e.g. `/etc/texmf/web2c/texmf.cnf`) and set the variable `TEXMFHOME` to the path of the working copy of the common `texmf` tree (e.g. by `TEXMFHOME = $HOME/texmf`); MiKTeX [20] users can add the path of the working copy of the common `texmf` tree in the "Roots" tab of the MiKTeX Options.

before they can use these classes and styles.[7] Furthermore, the repository contains manuals explaining the specific LaTeX software solution at our department (e.g. this document).

The **Subversion** server hosts a separate repository for each project of our department. Although branching, merging, and tagging is less important for writing text documents than for writing source code for software, our repository layouts follow the recommendations of [3]. In this sense, each repository has the three directories `/trunk`, `/branches`, and `/tags`.

The most important directory is `/trunk`. If a single text document belongs to the project, all files and subdirectories of this text document are in `/trunk`. If the project yields two or more different text documents, `/trunk` contains a subdirectory for each text document. A slightly different version (a *branch*) of a text document (e.g. for presentation at a conference) can be prepared either in an additional subdirectory of `/trunk` or in a new subdirectory of `/branches`. When a text document is submitted to a journal or a conference, we create a *tag* in the directory `/tags` so that it is easy to identify the submitted version of the document at a later date. This feature has been proven very useful. When creating branches and tags, it is important always to use the **Subversion** client (and not the tools of the local file system) for these actions, because this saves disk space on the server and it preserves information about the same history of these documents.

Often the question arises, which files should be put under version control. Generally, all files that are directly modified by the user and that are necessary for compiling the document should be included in the version control system. Typically, these are the LaTeX source code (`*.tex`) files (the main document and possibly some subdocuments) and all pictures that are inserted in the document (`*.eps`, `*.jpg`, `*.png`, and `*.pdf` files). All LaTeX classes (`*.cls`), LaTeX styles (`*.sty`), BibTeX data bases (`*.bib`), and BibTeX styles (`*.bst`) generally should be hosted in the repository of the common `texmf` tree, but they could be included in the respective repository, if some (external) co-authors do not have access to the common `texmf` tree. On the other hand, all files that are automatically created or modified during the compilation process (e.g. `*.aut`, `*.aux`, `*.bbl`, `*.bix`, `*.blg`, `*.dvi`, `*.glo`, `*.gls`, `*.idx`, `*.ilg`, `*.ind`, `*.ist`, `*.lof`, `*.log`, `*.lot`, `*.nav`, `*.out`, `*.pdf`, `*.ps`, `*.snm`, and `*.toc` files) or by the (LaTeX or BibTeX) editor

7. For instance, teTeX [8] users have to execute `texhash`; MiKTeX [20] users have to click on the button "Refresh FNDB" in the "General" tab of the MiKTeX Options.

(e.g. `*.bak`, `*.bib~`, `*.kilepr`, `*.prj`, `*.sav`, `*.tcp`, `*.tmp`, `*.tps`, and `*.tex~` files) generally should be *not* under version control, because these files are not necessary for compilation and generally do not include additional information. Furthermore, these files are regularly modified so that conflicts are very likely.

5 **Subversion** really makes the **diff**erence

A great feature of a version control system is that all authors can easily trace the workflow of a project by viewing the differences between arbitrary versions of the files. Authors are primarily interested in "effective" modifications of the source code that change the compiled document, but not in "ineffective" modifications that have no impact on the compiled document (e.g. the position of line breaks). Software tools for comparing text documents ("diff tools") generally cannot differentiate between "effective" and "ineffective" modifications; they highlight both types of modifications. This considerably increases the effort to find and review the "effective" modifications. Therefore, "ineffective" modifications should be avoided.

In this sense, it is very important not to change the positions of line breaks without cause. Hence, automatic line wrapping of the users' LaTeX editors should be turned off and line breaks should be added manually. Otherwise, if a single word in the beginning of a paragraph is added or removed, all line breaks of this paragraph might change so that most diff tools indicate the entire paragraph as modified, because they compare the files line by line. The diff tools **wdiff** [10] and **dwdiff** [11] are not effected by the positions of line breaks, because they compare documents word by word.[8] However, their output is less clear so that modifications are more difficult to track.

A reasonable convention is to add a line break after each sentence and start each new sentence in a new line.[9] Furthermore, we split long sentences into

8. These tools cannot be used directly with the **Subversion** command-line switch `--diff-cmd`, but a small wrapper script has to be used. [1]

9. This also has an advantage beyond version control: if you want to find a sentence in your LaTeX code that you have seen in a compiled (DVI, PS, or PDF) file or on a printout, you can easily identify the first few words of this sentence and screen for these words on the left border of your editor window.

several lines so that each line has at most 80 characters,[10] because it is rather inconvenient to search for (small) differences in long lines. We find it very useful to introduce the additional line breaks at logical breaks of the sentence, e.g. before a relative clause or a new part of the sentence starts. An example LaTeX code that is formatted according to these guidelines is the source code of this document, which is available on PracTeX's website.

There is also another important reason for reducing the number of "ineffective" modifications: if several authors work on the same file, the probability that the same line is modified by two or more authors at the same time increases with the number of modified lines. Hence, "ineffective" modifications unnecessarily increase the risk of conflicts (see section 2).

Furthermore, version control systems allow a very effective quality assurance measure: all authors should critically review their own modifications before they commit them to the repository (see figure 2). The differences between the user's working copy and the repository can be easily inspected with a single **Subversion** command or with one or two clicks in a graphical **Subversion** client. Furthermore, authors should verify that their code can be compiled flawlessly before they commit their modifications to the repository. Otherwise, the co-authors have to pay for these mistakes when they want to compile the document. However, this directive is not only reasonable for version control systems but also for all other ways to interchange documents among authors.

Subversion has a feature called "Keyword Substitution" that includes dynamic version information about a file (e.g. the revision number or the last author) into the contents of the file itself [3, chapter 3]. Sometimes, it is useful to include these information not only as a comment in the LaTeX source code, but also in the (compiled) DVI, PS, or PDF document. This can be achieved with the LaTeX packages **svn** [16], **svninfo** [2], or **svn-multi** [19] (preferably).

The most important directives for collaborative writing of LaTeX documents with version control systems are summarised in box 1.

10. For instance, the LaTeX editor **Kile** [14] can assist the user in this task when it is configured to add a vertical line that marks the 80$^{\text{th}}$ column.

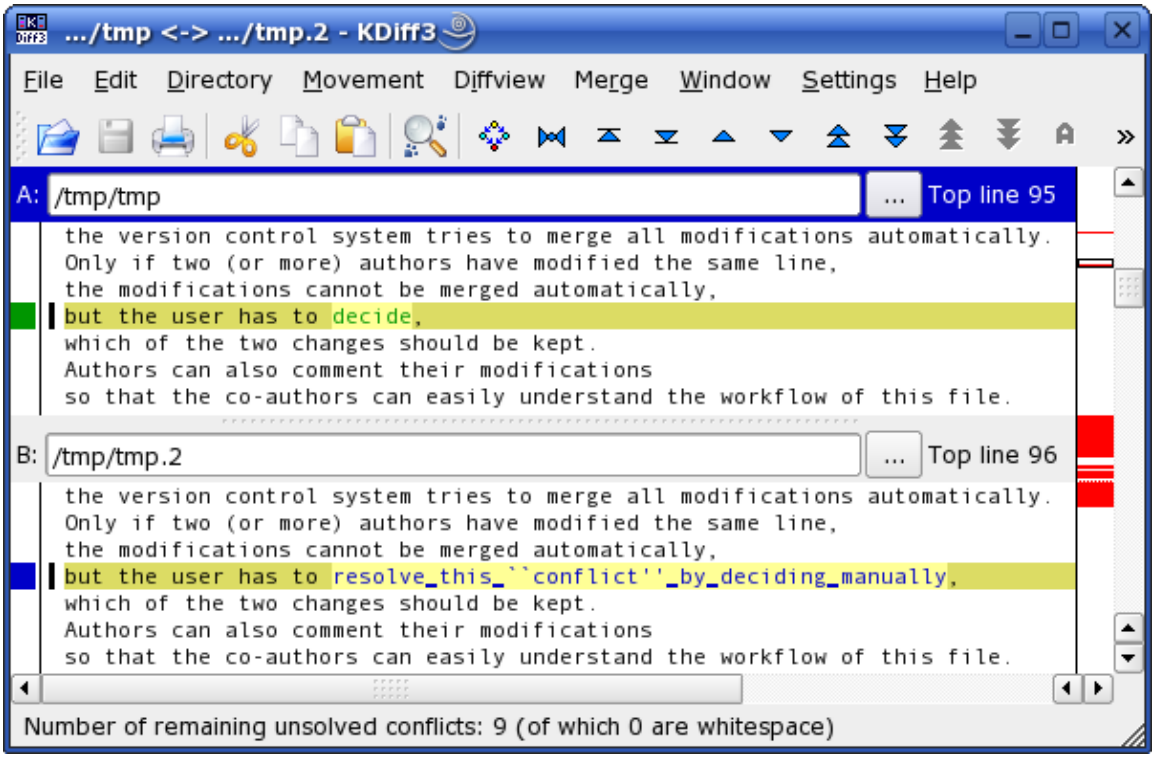


Figure 2: Reviewing modifications in **KDiff3**

6 Bibliography

Writing of scientific articles, reports, and books requires the citation of all relevant sources. BibTeX is an excellent tool for citing references and creating bibliographies [17, 9]. Many different BibTeX styles can be found on CTAN [23] and on the LaTeX Bibliography Styles Database [12]. If no suitable BibTeX style can be found, most desired styles can be conveniently assembled with **custombib**/**makebst** [4]. Furthermore, BibTeX style files can be created or modified manually; however this action requires knowledge of the (unnamed) postfix stack language that is used in BibTeX style files [18].

At our department, we have a common bibliographic data base in the BibTeX format (.bib file). It resides in our common `texmf` tree (see section 4) in the subdirectory `/bibtex/bib/` (see figure 1). Hence, all users can specify this bibliography

1. Avoid “ineffective” modifications.
2. Do not change line breaks without good reason.
3. Turn off automatic line wrapping of your LaTeX editor.
4. Start each new sentence in a new line.
5. Split long sentences into several lines so that each line has at most 80 characters.
6. Put only those files under version control that are directly modified by the user.
7. Verify that your code can be compiled flawlessly before committing your modifications to the repository.
8. Use **Subversion**’s diff feature to critically review your modifications before committing them to the repository.
9. Add a meaningful and descriptive comment when committing your modifications to the repository.
10. Use the **Subversion** client for copying, moving, or renaming files and folders that are under revision control.

Box 1: Directives for using LaTeX with version control systems

by only using the file name (without the full path) — no matter where the user’s working copy of the common `texmf` tree is located.

All users edit our bibliographic data base with the graphical BibTeX editor **JabRef** [24]. As **JabRef** is written in **Java**, it runs on all major operating systems. As different versions of **JabRef** generally save files in a slightly different way (e.g. by introducing line breaks at different positions), all users should use the same (e.g. last stable) version of **JabRef**.[11]

11. Otherwise, there would be many differences between different versions of `.bib` files that solely originate from using different version of **JabRef**. Hence, it would be hard to find the real differences between the compared documents. Furthermore, the probability of conflicts would be much higher (see section 5).

JabRef is highly flexible and can be configured in many details. We make the following changes to the default configuration of **JabRef** to simplify our work. First, we specify the default pattern for BibTeX keys so that **JabRef** can automatically generate keys in our desired format. This can be done by selecting `Options` → `Preferences` → `Key pattern` and modifying the desired pattern in the field `Default pattern`. For instance, we use `[auth:lower][shortyear]` to get the last name of the first author in lower case and the last two digits of the year of the publication (see figure 3).

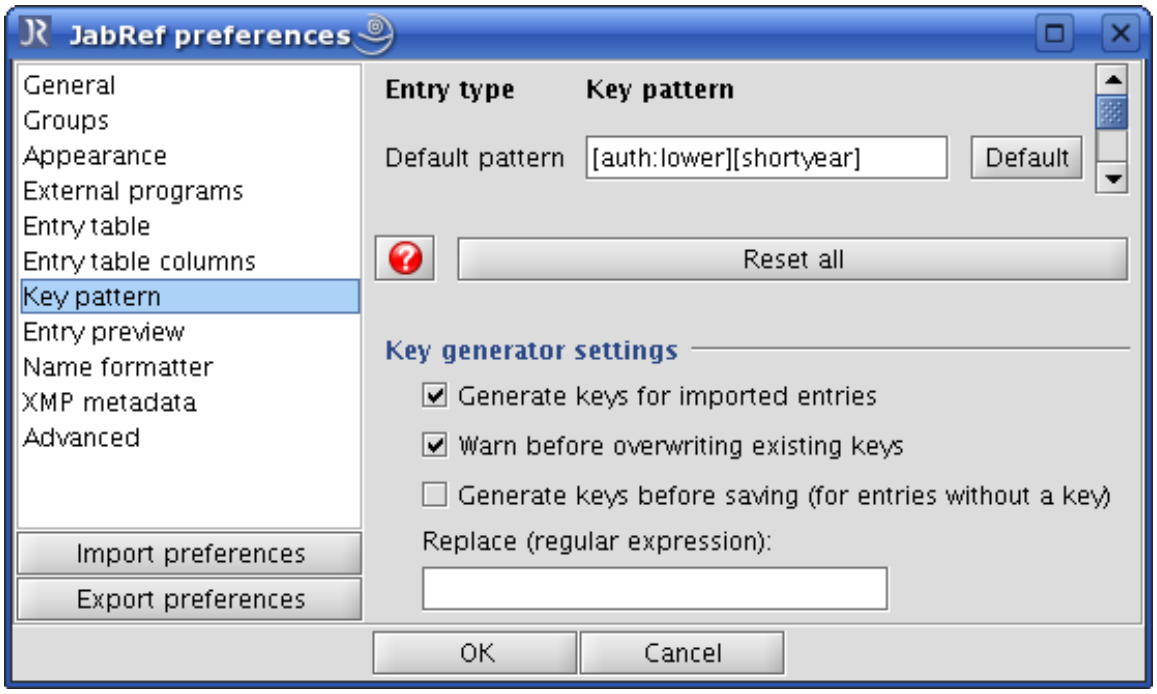


Figure 3: Specify default key pattern in **JabRef**

Second, we add the BibTeX field `location` for information about the location, where the publication is available as hard copy (e.g. a book or a copy of an article). This field can contain the name of the user who has the hard copy and where he has it or the name of a library and the shelf-mark. This field can be added in **JabRef** by selecting `Options` → `Set up general fields` and adding the word `location` (using the semicolon (`;`) as delimiter) somewhere in the line that starts

with `General:` (see figure 4).

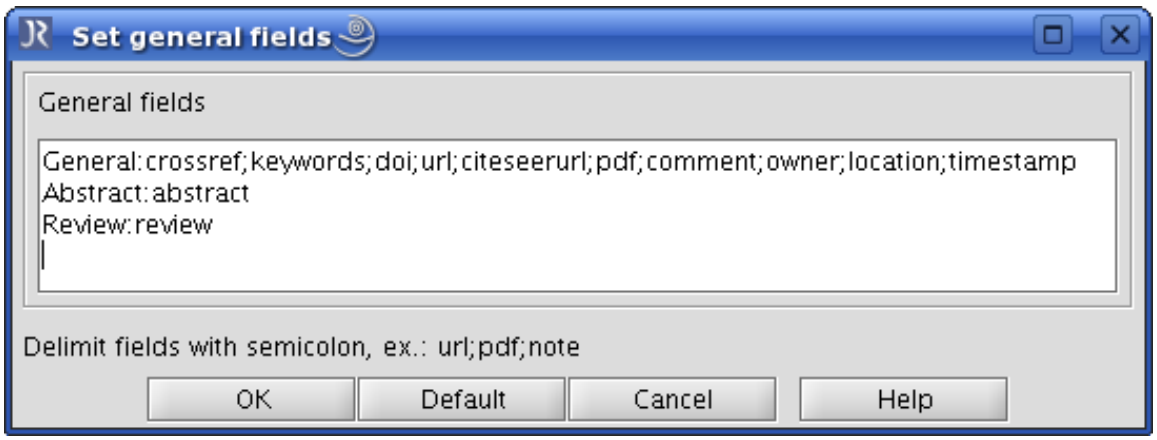


Figure 4: Set up general fields in **JabRef**

Third, we put all PDF files of publications in a specific subdirectory in our file server, where we use the BibTeX key as file name. We inform **JabRef** about this subdirectory by selecting `Options` → `Preferences` → `External programs` and adding the path of the this subdirectory in the field `Main PDF directory` (see figure 5). If a PDF file of a publication is available, the user can push the `Auto` button left of **JabRef**'s `Pdf` field to automatically add the file name of the PDF file. Now, all users who have access to the file server can open the PDF file of a publication by simply clicking on **JabRef**'s PDF icon.

If we send the LaTeX source code of a project to a journal, publisher, or somebody else who has no access to our common `texmf` tree, we do not include our entire bibliographic data base, but extract the relevant entries with the Perl script **aux2bib** [13].

7 Conclusion

This paper describes a possible way to efficiently organise the collaborative preparation of scientific LaTeX documents. The presented solution is based on the **Subversion** version control system and several other software tools and LaTeX packages. However, there are still a few issues that can be improved.

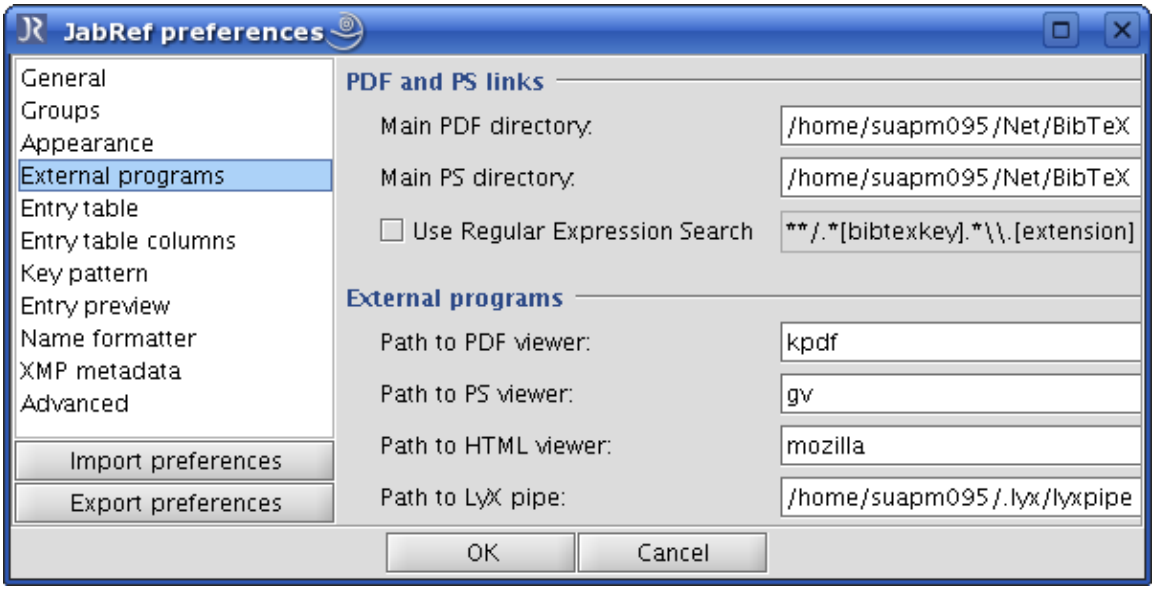


Figure 5: Specify "Main PDF directory" in **JabRef**

First, we plan that all users install the same LaTeX distribution. As the "TeX Live" distribution [28] is available both for Unix *and* MS Windows operating systems, we might recommend our users to switch to this LaTeX distribution in the future.[12]

Second, we consider to simplify the solution for a common bibliographic data base. Currently it is based on the version control system **Subversion**, the graphical BibTeX editor **JabRef**, and a file server for the PDF files of publications in the data base. The usage of three different tools for one task is rather challenging for infrequent users and users that are not familiar with these tools. Furthermore, the file server can be only accessed by local users. Therefore, we consider to implement an integrated server solution like **WIKINDX** [30], **Aigaion** [22], or **refBASE** [25]. Using this solution only requires a computer with internet access and a web browser, which makes the usage of our data base considerably easier for infrequent users. Moreover, the stored PDF files are available not only

12. Currently, our users have different LaTeX distributions that provide a different selection of LaTeX packages and different versions of some packages. We solve this problem by providing some packages on our common `texmf` tree.

from within the department, but throughout the world.[13] Even Non-LaTeX users of our department might benefit from a server-based solution, because it should be easier to use this bibliographic data base in (other) word processing software packages, because these servers provide the data not only in BibTeX format, but also in other formats.

Based on this paper, I have created a "Wikibook" on this subject [31]. All readers are encouraged to contribute to this book by adding further hints or ideas or by providing further solutions to the problem of collaborative writing of LaTeX documents.

Acknowledgements

I thank Francisco Reinaldo and Géraldine Henningsen for comments and suggestions that helped me to improve and clarify this paper, Karsten Heymann for many hints and advices regarding LaTeX and **Subversion**, and Christian Henning as well as my colleagues for supporting my intention to establish LaTeX and **Subversion** at our department.

References

[1] Mark James Adams. wdiff wrapper for svn. http://textsnippets.com/posts/show/1033.

[2] Achim D. Brucker. LaTeX package 'svninfo'. http://www.ctan.org/tex-archive/macros/latex/contrib/svninfo/.

[3] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly Media, 2007. http://svnbook.red-bean.com/.

[4] Patrick W. Daly. LaTeX package 'custom-bib'. http://www.ctan.org/tex-archive/macros/latex/contrib/custom-bib/.

13. Depending on the copy rights of the stored PDF files, the access to the server — or least the access to the PDF files — has to be restricted to members of the department.

[5] Aidan Delaney. Writing academic papers using Latex and Subversion (part 1). http://blogs.linux.ie/balor/2007/05/23/ [accessed 18-July-2007], May 2007.

[6] Dynamic Network Services, Inc. DynDNS – dynamic DNS service. http://www.dyndns.com/.

[7] Joachim Eibl. KDiff3. http://kdiff3.sourceforge.net/.

[8] Thomas Esser. teTeX. http://www.tug.org/tetex/.

[9] Jürgen Fenn. Managing citations and your bibliography with BibTeX. *The PracTEX Journal*, 4, 2006. http://www.tug.org/pracjourn/2006-4/fenn/.

[10] Free Software Foundation. wdiff. http://www.gnu.org/software/wdiff/.

[11] Gertjan P. Halkes. dwdiff. http://os.ghalkes.nl/dwdiff.html.

[12] Jean-Olivier Irisson. LaTeX bibliography styles database. http://jo.irisson.free.fr/bstdatabase/.

[13] Vivek Khera. BibTeX tool 'aux2bib'. http://www.ctan.org/tex-archive/biblio/bibtex/utils/bibtools/aux2bib, 1992.

[14] Kile Team. Kile – an integrated LaTeX environment. http://kile.sourceforge.net/.

[15] Igor V. Kovalenko and Julien Dumont. eSvn – a cross-platform GUI frontend for the Subversion revision system. http://zoneit.free.fr/esvn/.

[16] Richard Lewis. LaTeX package 'svn'. http://www.ctan.org/tex-archive/macros/latex/contrib/svn/.

[17] Nicolas Markey. Tame the BeaST. the B to X of BibTeX. http://www.ctan.org/tex-archive/info/bibtex/tamethebeast/ttb_en.pdf, 2005. Version 1.3.

[18] Oren Patashnik. Designing BibTeX styles. http://www.ctan.org/tex-archive/info/biblio/bibtex/contrib/doc/btxhak.pdf [accessed 18-July-2007], February 1988.

[19] Martin Scharrer. LaTeX package svn-multi. http://www.ctan.org/tex-archive/macros/latex/contrib/svn-multi.

[20] Christian Schenk. MiKTeX. http://www.miktex.org.

[21] TeX Users Group. A directory structure for TeX files. http://www.tug.org/tds/tds.html [accessed 18-July-2007], 2004. version 1.1.

[22] The Aigaion Developers. Aigaion – a web based bibliography management system. http://www.aigaion.nl/.

[23] The CTAN team. CTAN – The Comprehensive TeX Archive Network. http://www.ctan.org.

[24] The JabRef Developers. JabRef – an open source bibliography reference manager. http://jabref.sourceforge.net/.

[25] The refBASE Developers. refBASE – web reference database. http://refbase.sourceforge.net/.

[26] The Subversion developers. Subversion. http://subversion.tigris.org/.

[27] The Subversion developers. Subversion: Clients and plugins. http://subversion.tigris.org/links.html.

[28] The TeX Live Developers. TeX Live. http://www.tug.org/texlive/.

[29] The TortoiseSVN developers. TortoiseSVN – a subversion client implemented as a windows shell extension. http://tortoisesvn.tigris.org/.

[30] The WIKINDX Developers. WIKINDX. http://wikindx.sourceforge.net/.

[31] Wikibooks. Collaborative writing of LaTeX documents. http://en.wikibooks.org/wiki/LaTeX/Collaborative_Writing_of_LaTeX_Documents.

[32] Wikipedia. List of revision control software. http://en.wikipedia.org/w/index.php?title=List_of_revision_control_software&oldid=145420234 [accessed 19-July-2007], 2007.

THE PracTEX Journal

A Tool for Logicians

Arthur Buchsbaum and Francisco Reinaldo

Abstract

Among other uses, the turnstile sign is used by logicians for denoting a consequence relation, related to a given logic, between collections of formulas and formulas. Many logicians have complaint the lack of a LaTeX routine for issuing turnstile signs in a way better than it is provided by standard LaTeX, in any of the forms it could arise. The turnstile package accomplishes that and learning how to use it is very easy.

Arthur Buchsbaum is an Associate Professor of Logic and Discrete Mathematics of Department of Informatics and Statistics of Federal University of Santa Catarina, Brazil. He started using LaTeX in 1992, and gradually realized how important is this language for composing well structured papers and books. He loves Logic not only as an academic activity, but mainly as a powerful tool for learning how to think about everything in a precise way, which helps to get free from preconceived ideas. Visit Arthur's professional site in http://www.inf.ufsc.br/~arthur

Reinaldo's degree, latu sensu degree, master degree and PhD degree(in course) are in Computer Science. He is currently professor of electrical engineering and of computer science at the UnilesteMG/Brazil. At UnilesteMG he co-founded the Computational Intelligence Laboratory, and he is member of several projects involving Artificial Intelligence in Brazil and Portugal. Basically, his research has led to both theoretical and practical advances in artificial intelligence, cognitive science, neural networks, and behaviour-based approach. He has made major contributions in the domains of symbolic and connectionist learning. He has also been involved with many studies of advanced technologies for Multi Strategy Leaning Systems.

- PDF version of paper
- Article source
- Full turnstile package information (English)
- Full turnstile package information (Portuguese)
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/18

A Tool for Logicians

Arthur Buchsbaum and Francisco Reinaldo

Email arthur@inf.ufsc.br, reinaldo.opus@gmail.com

Abstract turnstile is a LATEX package that allows typesetting of the mathematical logic symbol, "turnstile", in all of the various ways it is used. This package was developed because there was no easy way in LATEX to typeset this symbol in its various forms, and place expressions above and below the crossbar.

1 Introduction

Logic is a science whose initial motivation was the analysis of correct reasoning. In recent years it has advanced beyond of the study of reasoning, and has many intersections with research areas such as Mathematics, Philosophy, Computer Science, Linguistics, Physics and Artificial Intelligence. One of the main signs used in Logic is the turnstile sign, from which there are versions such as "$\vdash$" and "$\models$", issued respectively by the LATEX commands `\vdash` and `\models`

2 The turnstile Project

The turnstile[1] is a sign often used by logicians for denoting a consequence relation, related to a given logic, between a collection of formulas and a formula. Many logicians have complained that there is no easy method in LATEX to typeset turnstile signs. They occur in many forms, and must be able to have expressions placed correctly above and below them. LATEX commands such as `\vdash` and `\models` typeset the turnstile sign, but they are not capable of placing data below or above them in an acceptable way. For example, sometimes it is necessary to place the name of a considered logical system below the turnstile sign, and sometimes it is necessary to put additional information above it.

1. http://tug.ctan.org/tex-archive/macros/latex/contrib/turnstile

If we want to say that a formula P is a logical consequence of a collection Γ of formulas in a logic L, we could try to typeset it by

```
\Gamma \vdash_\mathrm{L} P,
```

giving $\Gamma \vdash_{\mathrm{L}} P$.

Note that "L" was not placed correctly with respect to the turnstile sign; it should be placed and centred exactly below the sign. With `turnstile.sty` we can typeset it by

```
\Gamma \sststile{\mathrm{L}}{} P,
```

giving $\Gamma \sststile{\mathrm{L}}{} P$.

On the other hand, if we want to say that a formula P is a logical semantical consequence of a collection Γ of formulas in a logic L, through varying of x and y[2], we could try to typeset it by

```
\Gamma \models_\mathrm{L}^{x,y} P,
```

giving $\Gamma \models_{\mathrm{L}}^{x,y} P$.

Note that both "L" and "x, y" were not placed correctly with respect to turnstile sign; they should be placed and centered exactly below and above the sign. With `turnstile.sty` we can typeset it by

```
\Gamma \sdtstile{\mathrm{L}}{x,y} P,
```

giving $\Gamma \sdtstile{\mathrm{L}}{x,y} P$.

The commands provided by this style are all formed by the string "tstile" preceded by a string with two or three letters. These letters indicate the kind of lines to be drawn successively; they can be "n", "s", "d", or "t". The letter "n" says that the line is empty, the letter "s" that the line is single, the letter "d" that the line is double, and finally "t" indicates that the line is triple. The corresponding commands for the two-lettered strings provide the most common turnstile signs, in which there is not a second vertical line following the horizontal line. The first letter of these strings indicates the kind of the vertical line and the second the

2. In [1] and [2] varying objects are presented.

kind of the horizontal line to be drawn after the vertical line. The three-lettered strings can contain any of the letters “n”, “s”, “d”, and “t”, with the restriction that the last letter must not be “n”, because the case in which the third line is empty is already dealt with by the commands with two-lettered strings preceding “tstile”. The first letter specifies the kind of the first vertical line, the second the kind of the horizontal line, and the third letter the kind of the second vertical line.

All these commands have three arguments, and the first one is optional.

The first argument, which is optional, gives the size by which the internal expressions must be displayed: “d” for displayed formulas, “t” for text formulas, “s” for first subscript or superscript formulas, and “ss” for later subscript or superscript formulas. The default value is “s”. The result of applying “t” or “d” is the same, except if there is a mathematical sign in the second or third argument issued in distinct ways, depending on whether it is used in text math mode or displayed math mode.

The second and third arguments provide the expressions to be placed below and above the turnstile sign respectively, where both these expressions are converted to the size specified by the first argument. On the other hand, if the second or the third argument is empty, then nothing is put below or above the turnstile sign.

3 Examples

Some examples are shown below. For the sake of illustration, Γ is a given collection of formulas and P is a logical formula. Of course, the signs “Γ” and “P” illustrate only one possible context in which the turnstile sign could appear.

```
\Gamma \sststile{}{} P
```

$$\Gamma \sststile{}{} P \tag{1}$$

```
\Gamma \sststile{\mathrm{LPD}}{} P
```

$$\Gamma \sststile{\mathrm{LPD}}{} P \tag{2}$$

```
\Gamma \sststile{}{x,y} P
```

$$\Gamma \sststile{}{x,y} P \tag{3}$$

If the optional argument is not used, then the result is the same as if "s" was the optional argument:

```
\Gamma \sststile{\mathrm{LPD}}{x,y} P
```

$$\Gamma \sststile{\mathrm{LPD}}{x,y} P \tag{4}$$

```
\Gamma \sststile[d]{\mathrm{LPD}}{x,y} P
```

$$\Gamma \sststile[d]{\mathrm{LPD}}{x,y} P \tag{5}$$

```
\Gamma \sststile[t]{\mathrm{LPD}}{x,y} P
```

$$\Gamma \sststile[t]{\mathrm{LPD}}{x,y} P \tag{6}$$

```
\Gamma \sststile[s]{\mathrm{LPD}}{x,y} P
```

$$\Gamma \sststile[s]{\mathrm{LPD}}{x,y} P \tag{7}$$

```
\Gamma \sststile[ss]{\mathrm{LPD}}{x,y} P
```

$$\Gamma \sststile[ss]{\mathrm{LPD}}{x,y} P \tag{8}$$

```
\Gamma \sststile{\mathrm{LPDEFGH}}{x,y} P
```

$$\Gamma \sststile{\mathrm{LPDEFGH}}{x,y} P \tag{9}$$

```
\Gamma \sststile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \sststile{\mathrm{LC}}{x,y,z,w} P \tag{10}$$

```
\Gamma \sdtstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \sdtstile{\mathrm{LC}}{x,y,z,w} P \tag{11}$$

```
\Gamma \dststile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \dststile{\mathrm{LC}}{x,y,z,w} P \tag{12}$$

```
\Gamma \ddtstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \ddtstile{\mathrm{LC}}{x,y,z,w} P \tag{13}$$

```
\Gamma \dtttstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \Vert\!\!\Vert\!\!\Vert\!\!\Vert \overset{x,y,z,w}{\underset{\text{LC}}{}} P \tag{14}$$

```
\Gamma \nssststile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \nVdash \overset{x,y,z,w}{\underset{\text{LC}}{}} P \tag{15}$$

```
\Gamma \ndststile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \nvDash \overset{x,y,z,w}{\underset{\text{LC}}{}} P \tag{16}$$

```
\Gamma \nsdtstile{\mathrm{LC}}{x,y,z,w} P
```

$$\overset{x,y,z,w}{\underset{\text{LC}}{}} \nVdash P \tag{17}$$

```
\Gamma \nddtstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \nvDash \overset{x,y,z,w}{\underset{\text{LC}}{}} \nVdash P \tag{18}$$

```
\Gamma \ndtttstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \Vert\!\!\Vert\!\!\Vert\!\!\Vert \overset{x,y,z,w}{\underset{\text{LC}}{}} \nVdash P \tag{19}$$

```
\Gamma \sssttstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \left|\overset{x,y,z,w}{\underset{\text{LC}}{\equiv}}\right| P \tag{20}$$

```
\Gamma \stststile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \left|\overset{x,y,z,w}{\underset{\text{LC}}{\equiv}}\right| P \tag{21}$$

```
\Gamma \ttttstile{\mathrm{LC}}{x,y,z,w} P
```

$$\Gamma \left|\overset{x,y,z,w}{\underset{\text{LC}}{\equiv}}\right|\!\left| P \right. \tag{22}$$

Below are some examples of mathematical expressions below and above the turnstile sign which show how it changes depending on the optional argument. If no optional argument is given, then it is considered to be “s”.

The reader should also note that the vertical lines don’t stretch according to the heights of the expressions located below and above the turnstile sign. Because logicians use this sign mainly in text mode, we feel that it should have a standard height.

```
\Gamma \sststile{\sum_0^\infty 1/2^n}{\int_a^b f} P
```

$$\Gamma \left|\frac{\int_a^b f}{\sum_0^\infty 1/2^n}\right. P \tag{23}$$

```
\Gamma \sststile[d]{\sum_0^\infty 1/2^n}{\int_a^b f} P
```

$$\Gamma \left|\frac{\displaystyle\int_a^b f}{\displaystyle\sum_0^\infty 1/2^n}\right. P \tag{24}$$

```
\Gamma \sststile[t]{\sum_0^\infty 1/2^n}{\int_a^b f} P
```

$$\Gamma \left|\frac{\int_a^b f}{\sum_0^\infty 1/2^n}\right. P \tag{25}$$

```
\Gamma \sststile[s]{\sum_0^\infty 1/2^n}{\int_a^b f} P
```

$$\Gamma \left|\frac{\int_a^b f}{\sum_0^\infty 1/2^n}\right. P \tag{26}$$

```
\Gamma \sststile[ss]{\sum_0^\infty 1/2^n}{\int_a^b f} P
```

$$\Gamma \left|\frac{\int_a^b f}{\sum_0^\infty 1/2^n}\right. P \tag{27}$$

4 Conclusions

The package `turnstile.sty` seems to be adequate for typesetting the turnstile sign in its many forms. It correctly places additional expressions below and above it, if necessary, and stretches the the crossbar width as much as needed to contain the expressions.

For a future version of this package, we want to look at changing the height of the turnstile sign. This will take into account the heights of the expressions above and below, similar to the way we currently allow for the widths of the expressions.

References

[1] Arthur Buchsbaum and Jean-Yves Béziau. Introduction of implication and generalization in axiomatic calculi. In Jean-Yves Béziau, Alexandre Costa Leite, and Alberto Facchini, editors, *Aspects of Universal Logic*, number 17 in Travaux de Logique, page 231. Centre de Recherches Sémiologiques, Université de Neuchâtel, December 2004.

[2] Arthur Buchsbaum and Tarcisio Pequeno. A general treatment for the deduction theorem in open calculi. *Logique et Analyse*, 157:9–29, January–March 1997.

[3] Helmut Kopka and Patrick W. Daly. *A Guide to LaTeX*. Addison-Wesley, 1999.

[4] Leslie Lamport. *LaTeX – A Document Preparation System – User's Guide and Reference Manual*. Addison-Wesley, 1994.

[5] Anil Nerode and Richard A. Shore. *Logic for Applications*. Springer, 1997.

[6] John Nolt. *Logics*. Wadsworth, 1996.

THE PracTEX Journal

LaTeXing with TextMate

Charilaos Skiadas and Thomas Kjosmoen

Abstract

This article discusses the TextMate text editor and its many capabilities that make working with LaTeX documents a lot easier. Some of its features include syntax highlighting, various methods for automatic insertion of text (such as the begin-end blocks in environments and automatic labels for section commands), lookup of labels and cite keys based on partial matches, as well as tools for dealing with large projects.

TextMate is designed with the user in mind, so it is easy to customize it to your needs. During its short lifetime (about two and a half years) it has gained many supporters and has become a very popular text editor for the Mac OS X platform, and especially among LaTeX users, as can be seen from the exponential growth in the number of users, and the large number of LaTeX related questions on the TextMate mailing list.

In addition to this article, the first author's weblog can be used as a starting point for learning more about using TextMate for LaTeX: http://skiadas.dcostanet.net/afterthought

Charilaos (Haris) Skiadas is an Assistant Professor of Mathematics at Hanover College, and has been using LaTeX for all his writing ever since he found out about it ten years ago. For the last two years he has been very actively involved in the development of the LaTeX support for the TextMate text editor, one of the popular text editors for the Mac OS X operating system. His blog (http://skiadas.dcostanet.net/afterthought) contains a lot of information and screencasts related to LaTeX in TextMate.

Thomas Kjosmoen is a PhD candidate in the Department of Computer Science and Electrical Engineering at the University of Stavanger, Norway, where he works on bioinformatics and genomic signal processing. Ever since he discovered LaTeX about four years ago, it has filled most of his document needs. He particularly appreciates the open nature of the LaTeX documents and the fact that it makes them easy to keep under version control." Homepage: http://kjosmoen.org

You can reach Haris at cskiadas@gmail.com and Thomas at thomas.kjosmoen@uis.no

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

LaTeXing with TextMate

Charilaos Skiadas and Thomas Kjosmoen

Abstract This article discusses the TextMate text editor and its many capabilities that make working with LaTeX documents a lot easier. Some of its features include syntax highlighting, various methods for automatic insertion of text (such as the begin-end blocks in environments and automatic labels for section commands), lookup of labels and cite keys based on partial matches, as well as tools for dealing with large projects.

TextMate is designed with the user in mind, so it is easy to customize it to your needs. During its short lifetime (about two and a half years) it has gained many supporters and has become a very popular text editor for the Mac OS X platform, and especially among LaTeX users, as can be seen from the exponential growth in the number of users, and the large number of LaTeX related questions on the TextMate mailing list.

In addition to this article, the first author's weblog can be used as a starting point for learning more about using TextMate for LaTeX: http://skiadas.dcostanet.net/afterthought

1 Introduction

We spend a large part of our lives working on LaTeX documents. It could be our new hot paper to be published in a top scientific journal, our *magnum opus* that we have worked on for the last 10 years and is already more than 1000 pages long, or just a shopping list for tomorrow's trip to the grocery store.

But whatever the content of that LaTeX document is, we are all faced with a common set of tasks that we need to perform:

1. Common text-editing tasks, such as adding new text, deleting existing text, or simply moving text around
2. Inserting various LaTeX commands and environments, and references
3. Compiling and viewing our output, locating the parts of the code corresponding to a particular part of the output, and locating and fixing errors

Text Editors are particularly suited for editing text. After all that's what they were designed for. Many editors go further and offer a lot of extra functionality particularly targeted toward working with LaTeX documents, thus helping the user with the last two sets of tasks described above. This article is about one such editor, *TextMate*.

TextMate[1] first appeared in 2004, and it has in many ways changed the way we think about text editing. It has a clean, simple interface and it is very easy to customize, even for people with little or no programming skills. It draws most of its power from its modular approach, where a lot of the functionality is placed in various collections of preferences and commands called *bundles*. These bundles can be easily edited and configured to fit each user's particular needs. Most of TextMate's LaTeX-related abilities are drawn from the corresponding LaTeX bundle, which has been refined over the years by the contributions of many individuals, in order to make LaTeXing in TextMate as painless as possible.

One thing that should be mentioned here is that TextMate is available only on the Mac OS X operating system, as it uses a lot of features available only on that platform. However, recently a number of text editors for the Windows platform have been developed to draw from the power of TextMate's bundles mechanism.[2]

We will continue this article in Section 2 by looking closer at the core TextMate features that are useful in editing LaTeX documents. We will then in Section 3 move on to discuss some of the commands and tools that are included in the LaTeX bundle. As hopefully will become apparent, TextMate makes writing LaTeX almost a pleasure.

This article is not an attempt to offer a complete account of all that TextMate has to offer to LaTeX-editing. If you want to find out more, you can see a number of TextMate's features in action via the numerous screencasts available[3], including two specifically for the LaTeX bundle[4]. You can also learn a lot more from the recent book on TextMate ([2]), as well as the first author's weblog[5], which contains a lot of articles related to TextMate in general, and LaTeX in TextMate in

1. http://www.macromates.com
2. http://www.e-texteditor.com and http://intype.info/home/index.php
3. http://macromates.com/screencasts
4. http://macromates.com/screencast/latex_part_1.mov and http://macromates.com/screencast/latex_part_2.mov
5. http://skiadas.dcostanet.net/afterthought.

particular[6].

2 TextMate

Some of TextMate's features will be familiar to people used to working with an advanced text editor, for instance the ability to select whole words with a simple keystroke, to navigate quickly through the text, and to cut and paste whole chunks of text. TextMate also has a powerful "Search and Replace" mechanism that allows the use of regular expressions, which for instance makes it possible to easily search for text of the form: `\verb!anything here!`, and to automatically convert them to:

```
\begin{verbatim}
  anything here
\end{verbatim}
```

Often, we must work on several related files. To handle this, TextMate has a *Project* mode[7], which makes managing multiple files very easy. Specifically, for each project, we can set up a master LaTeX document that pulls in the other documents using `\include` or `\input` commands. TextMate is then able to view the project as a whole and handle the compilation correctly, and pull in cross references from any of the LaTeX documents. You can navigate to the various files via the *Project Drawer*, or you can move around much more quickly via the very elegant "Go to File" menu[8]. All that is required is, that you know a couple of letters from the name of the file you want to open.

2.1 Syntax Highlighting and Scopes

TextMate truly excels when it comes to highlighting a document. This is accomplished via certain parts of the bundles called *Language Grammars*. A language grammar tells TextMate to locate particular patterns in a text, and assign to each

6. See: http://skiadas.dcostanet.net/afterthought/list-of-my-textmate-pages.
7. http://www.macromates.com/textmate/manual/working_with_multiple_files
8. http://www.macromates.com/textmate/manual/working_with_multiple_files#moving_between_files_with_grace

of them a *scope*. Then, the chosen *Color Theme* assigns formatting to each scope, such as foreground and background colors, typeface, and underlining. Figure 1 shows how an early version of this particular article looks in TextMate. The formatting can be customized by the user very easily, and there are available many very nice themes for all tastes.

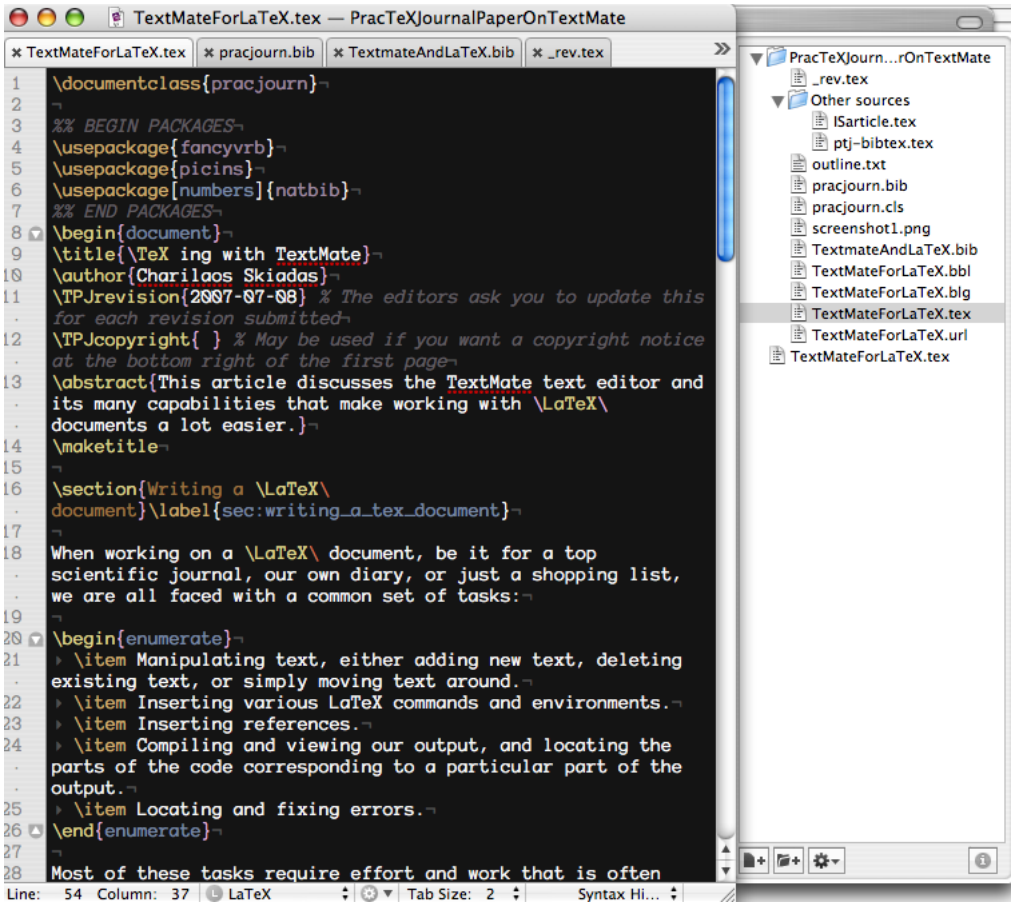


Figure 1: TextMate's Project Drawer, and Syntax Highlighting

In addition to providing visual formatting, scopes enable other important and powerful features; we can make TextMate behave differently depending on context. For example, by default, pressing the escape key offers auto-completion based on all words in the current document. If, on the other hand, the caret[9]

9. The location where the text will appear when you type, as opposed to the *cursor*, which is the

is inside a `\ref` command, then the auto-completion takes into account only the words that appear in a `\label` command. This way, you can keep your labels very long and descriptive, relying on the fact that knowing the first couple of letters is enough to insert the whole label quickly. Likewise, if the caret is inside a `\cite` command, TextMate looks for corresponding cite keys in the associated BibTeX files or in `\bibitem` entries in the text.

In other words, each component of your text is not just a set of characters, but has a particular meaning to TextMate, corresponding to its particular meaning to LaTeX. For instance a LaTeX label is really perceived by TextMate as a label, a section or subsection is understood as such, each environment is a unit of its own. Every part of the document comes alive and interacts with other parts in an intelligent way. A user can address these parts and call them by their "name", and they will reply!

Another advantage of the scope mechanism is that it allows for the insertion of one grammar in another. Thus, if you have an `lstlisting` environment containing Python code like the following:

```
\begin{lstlisting}[label=lst:vertex,float=htbp] % Python
class Vertex:
    def __init__(self,num):
        self.id = num
        self.adj = []
        self.color = 'white'
        self.dist = sys.maxint

    def addNeighbor(self,nbr,cost=0):
        self.adj.append(nbr)
        self.cost[nbr] = cost
\end{lstlisting}
```

then the Python code part will be colored according to the rules for the Python grammar, *i.e.* it will be colored as what it really is. This, as far as we know, is a unique feature of TextMate.

---

mouse pointer on the screen.

2.2 Snippets

One of the features of TextMate that was instantly loved by its users was *snippets*[10]. In its simplest form, a snippet is a piece of text that can be inserted into the document. Often when writing code or documents, we rewrite a lot of text. With TextMate, we can save those chunks of text as snippets, and have them inserted by using a shortcut. Think of them, if you like, as your e-mail signature. However, this would be like saying that LaTeX is just a software for creating shopping lists.

In fact, snippets are so much more than just boilerplate; they can be *dynamic*. In other words, they can include code that gets executed when the snippet is inserted. This way, for instance, you could have a snippet that inserts the current date and time, the amount of free space available in your computer, a full listing of all files in hard drive, a stock price downloaded from the internet that very moment, or anything else you can possibly imagine. More importantly, you can insert *placeholders* and *tab stops*, that you would use to type in text in various parts of the snippet. For example, this allows you to create a little snippet that, at the press of a key, will insert the text:

```
\begin{foo}
  bar
\end{foo}
```

and highlight the word `foo` in the `\begin` block. Whatever you type in there will be automatically *mirrored* on the `foo` in the `\end` block. When you are done with that, pressing tab would automatically select the word `bar` for you. This saves us a lot of keystrokes and makes sure we never forget to close an environment; perhaps the most common mistake we make, not to mention difficult to track down.[11] Taking this concept one step further, we can for instance create a snippet that automatically creates a label based on the section name.[12] Another option, a powerful and versatile one, is to use commands to insert snippets. We will discuss commands in some detail in the next section.

10. http://www.macromates.com/textmate/manual/snippets
11. If you worry about how you would create this particular snippet, don't: It's already built into the LaTeX bundle. But you can take our word for it, that it is indeed very easy to create.
12. Before your rush to your computers and try to create this snippet, keep in mind that this is already built into the LaTeX bundle as well.

The icing on the proverbial cake is that the snippets are easy for non-technical users to create. One quick read through the snippet section in the manual is all it takes to get you started on creating your very own snippets!

2.3 Commands

Snippets are easy to create, but they are somewhat limited. A more powerful feature, one that has almost unlimited capabilities, are *commands*[13]. They do, however, require a bit more work and some programming skills. Simply put, a command is a script which can process either the whole document, or part of it, and make changes to it, create a new document, or even generate an HTML window. This HTML window can even contain links back to specific parts of the document. You can write commands in any scripting language, like Bash, Ruby, Perl, Python, etc. In fact, all of the functionality discussed in the next section is provided by TextMate commands, so there will be a lot of examples of commands there. Here we will briefly discuss the various ways in which commands can be customized.

First of all, we have a number of options about the input text that the command will accept. The input could be the entire document, the selected text, or simply nothing. Secondly, we have many options for the output; it could replace the selected text, it could be used to create a new document, used to create an HTML window, or simply discarded.

Finally, and this is something that can be done for snippets as well, you can specify a scope for the command. Thus, the command can only be executed when the caret is in that particular scope, *e.g.* we can have commands that only work when inside a math environment or if the document is a Beamer document.

Let us consider a specific example of what commands can do. You could create a search command that will take into account the current selection, or the current word if you haven't selected anything, and will find all appearances of the word in the text. The command could then open up an HTML window listing all the locations of the search hits, allowing you to click in the HTML window and be taken to those locations in the text.

A specific type of command, a *Drag Command*, allows you to specify to TextMate what you want it to do when files of particular type are dropped into the

13. http://www.macromates.com/textmate/manual/commands

text. For instance, a command in the LaTeX bundle tells TextMate to automatically create a figure environment when an image file is dropped into a LaTeX document.

We have now covered the basics of what commands can do. The details are beyond the scope of this article, and we encourage you to take a look at the TextMate manual and explore the existing bundles to find out more.[14]

3 TextMate's LaTeX Bundle

Let us now move on to explore the LaTeX bundle in some detail. The LaTeX bundle is a set of *TextMate commands*[15], snippets and *macros*, that collectively enhance TextMate's LaTeX editing abilities. Roughly speaking, they fall under the following broad categories:

- Project manipulation; compilation, navigation, outline
- Automatic text insertion; environments, wrapping, labels, and references

3.1 Project Manipulation

Compilation The most essential command in the bundle is the *Typeset & View* command, which typesets the document and opens the resulting output file (usually PDF) in the appropriate viewer program. The command is smart in the sense that if you are working with a project involving a master document, then the master document will be compiled, regardless of what document the command was issued from. It is also smart about looking at the packages used, and deciding based on those what compilation steps to take.[16]

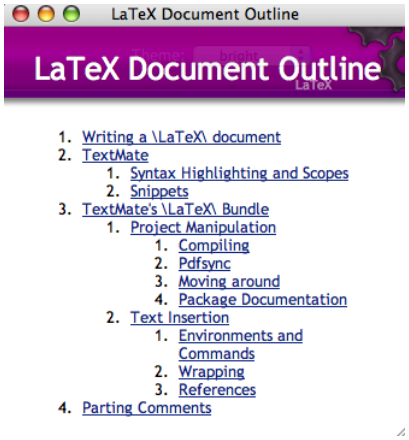
When the command is executed, the compilation log is presented in a separate window with all warnings and errors hyperlinked to their corresponding positions in the document. This is, of course, a feature that most LaTeX-editors have.

14. When TextMate runs a command, it makes many useful environment variables available to the command. These environment variables provide information about the current document. For more details, see `http://www.macromates.com/textmate/manual/environment_variables`.

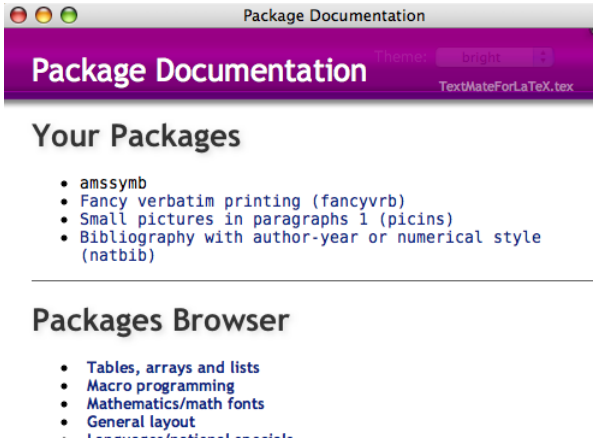
15. `http://www.macromates.com/textmate/manual/commands`

16. For instance if it detects the `pstricks` package, it will use latex+dvi2ps+ps2pdf instead of pdflatex.

Pdfsync If you are using a previewer that utilizes the `pdfsync` package[17], then you can easily go back and forth between the TEX source and the previewed PDF file. Namely, a viewer command can take you from a location in the previewed PDF file to the corresponding location in the LATEX document. Conversely there is a command in TextMate that will take you to the previewer at the area corresponding to the current caret location in the document.[18]



(a) The "Show Outline" command provides an outline of the entire project for easy navigation.



(b) The "Package Browser" window allows navigating through the documentation for the various LATEX packages.

Figure 2: Document and Package Navigation is made easier by two TextMate commands.

Navigation TextMate has a built-in command for navigating among the various 'symbols' of the document, which in the case of LATEX would be the various sections, subsections etc. This command shares the elegant interface of the "Go to File" command[19]. There is also the "Show Outline" command, which produces an outline of the entire project (Figure 2a). Clicking on any item in the outline

17. 'Skim' (`http://skim-app.sourceforge.net`) is a very nice such viewer.
18. The precision of this back and forth is of course limited by the information stored by `pdfsync`. Another popular LATEX editor, TeXShop, utilizes an additional Mac OS X technology, Spotlight, to offer more precise syncing by using text comparisons.
19. `http://www.macromates.com/textmate/manual/navigation_overview#function_pop-up`

takes you to the corresponding position in the project, regardless of which file that position is in. Finally, we have a command that take us from an `\include` line to the actual file being included.

Package Documentation For those of you constantly referring to package documentation, the "Documentation for Package" command offers easy access to the documentation of all packages via a "Package Browser" window. You can then navigate through the various packages, and have the corresponding documentation files open up in your favorite PDF viewer.

3.2 Automatic Text Insertion

Environments and Commands Adding text is probably what we do most of the time when working with a LaTeX document. TextMate has a number of commands to help us do this. Two of the most often used ones are the commands for creating environments and LaTeX commands based on an abbreviation. For instance, typing `it` and pressing the key combination that triggers the "Insert Environment" command produces:

```
\begin{itemize}
  \item
\end{itemize}
```

with the caret right at the end of the `\item` line. Pressing the Enter key at that point will add a new line and automatically insert the `\item` part. The "Insert Environment" command responds to many other abbreviations, and you may add your own abbreviations as well.

Wrapping Two other commands allow you to wrap the currently selected text in a LaTeX environment or command. Suppose that you want to make `two words` bold. This is simply done by selecting the `two words`, and using the appropriate shortcut.[20] The command will then wrap the selection like `\emph{two words}` and automatically select `emph`. You can now enter the wanted command, in this case

20. It should be noted that using these shortcuts without first selecting any text will simply wrap the word which is in contact with the caret.

`textbf`[21], and finish by pressing tab to place the caret after the newly wrapped words. Similarly, we have a shortcut for wrapping a selection in an environment, again giving you the environment argument selected for entry. For example, if you want to center the selection, enter `center` in the argument placeholder and press tab to finish:

```
\begin{center}
  two words
\end{center}
```

There is even a command that will allow you to change the name of an existing environment, and another one that will toggle between a starred and a non-starred environment. Both can be executed if the caret is inside the environment (*i.e.* without having to select the entire environment).

References Perhaps the most tedious task in LaTeX is inserting references, both in the case of the `\cite` and the `\ref` commands. Suppose you want to insert a citation from the PracTeX journal bibliography[22], namely an article on BibTeX ([1]). You are guessing it has "BibTeX" in the title, but can't remember anything else about it, and you certainly can't remember the exact cite key for it. In TextMate, all you have to do is type: `\cite{Bib}` and press the key for completion, and it will become `\cite{Fenn:PJ:2006-4}`. (In fact this is exactly how we created this entry just now.) If, on the other hand, you type `\cite{TeX}` and then press the completion key, then you get a menu to choose from, with all the items that match the word TeX (Figure 3).

A similar principle extends to label referencing. Let us assume you used the common convention that the labels for all equations start with `eqn:`, those for sections with `sec:`, and other labels correspondingly, and that you wanted to add a reference to a particular equation number. Then, you could simply type: `\eqref{eqn}` and press the completion key (the same key as discussed above, but since the scope is different now, TextMate will change behaviour accordingly). TextMate will reply with offering you a list of all the labels in your project, *i.e.* from all the files included in the master document, that start with the word 'eqn'.

21. Actually for this particular case there is another command that will just wrap the text in a textbf block, without you having to type anything else.
22. http://www.tug.org/pracjourn/pracjourn.bib

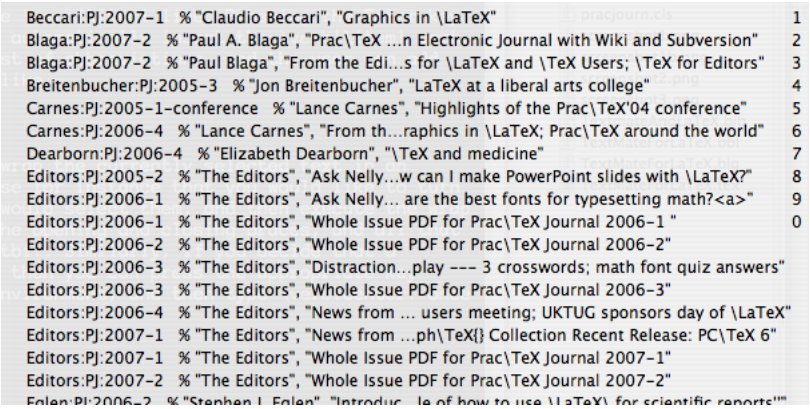


Figure 3: The menu provided by the Bibliography Completion command offers summary information for each matching bibliographic entry.

You can then pick the preferred one. Similarly, in the current document there is a label `sec:text_insertion`. If you simply type: `\ref{ins}` and press the completion key, it will be converted to `\ref{sec:text_insertion}`.

4 Parting Comments

This article has just scratched the surface of what TextMate can do to help you with your LaTeXing. There are commands for creating new tables and for converting text selection to a LaTeX table, for converting a text selection to an itemized list, and much more. Not to mention the support TextMate offers for working with Sweave documents.[23] The screencasts referred to in the introduction can illustrate what TextMate can do much better than any words could, and the LaTeX bundle help file has more up-to-date information, along with the corresponding shortcuts for all the commands.

Though TextMate is not free[24], it comes with a 30 day all-features-included trial version that is more than enough to get you hooked on it. The authors are two of those happy TextMate-addicts, with the first author having used it for more than two years, and the second author having used it for less than a year.

23. http://www.stat.umn.edu/~charlie/Sweave
24. The core TextMate application is not free, but the bundles are an open-source project and more than 100 people contribute to them.

References

[1] J. Fenn. Managing citations and your bibliography with bibtex. *The PracTEX Journal*, (4), 2006.

[2] James Edward Gray II. *TextMate: Power Editing for the Mac*. Pragmatic Bookshelf, 2007.

THE PracTEX Journal

LaTeXe "Linux-like" environment on MacOSX

Vinicius Provenzano

Abstract

Free and commercial LaTeX implementations for the Mac OS X are available on the internet. If you have always used a Mac, the best starting point is to download and install one of these systems. However if you have always used Linux and now find yourself in front of a brand new Mac OS X machine and have no time to learn new tools from scratch, your best option would be to use your familiar Linux applications on Mac OS X. This paper aims to show you how to install and configure a Linux-like LaTeX environment in the Mac OS X, using Fink, teTeX and Kile.

Vinicius Provenzano was born in Rio de Janeiro, Brazil in 1975. He graduated in Economics in 2001 and obtained his Masters Degree in Business and Information Technology in 2003. He used LaTeX to write both his monograph and dissertation, using ABNTEX, a LaTeX and BibTex macro set developed to produce documents in conformity with Brazilian standards (NBR14724:2001, NBR6028:1990, NBR6027:1989, NBR6024:1989, NBR6023:2002, NBR10520:2002) on Linux OS. After working in IT security for a few years, he now works in North Africa as ITC coordinator for an Italian company. While he is a Windows user and administrator at work, a former OS/2 fan, and an early adopter of Linux, he is now fascinated by Mac OS X. You may contact Vinicius at viniciusxp@gmail.com or at http://www.camelomanco.com.

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/21

A LaTeX2ε "Linux-like" environment on Mac OS X

Vinicius Provenzano

Abstract Free and commercial LaTeX2ε implementations for the Mac OS X are available on the internet. If you have always used a Mac, the best starting point is to download and install one of these systems. However if you have always used Linux and now find yourself in front of a brand new Mac OS X machine and have no time to learn new tools from scratch, your best option would be to use your familiar Linux applications on Mac OS X. This paper aims to show you how to install and configure a Linux-like LaTeX2ε environment in the Mac OS X, using Fink, teTeX and Kile.

1 Introduction

The Apple Mac OS X has a great collection of tools for LaTeX2ε users. If you have always used Mac OS X this article is not for you. But, if you need a familiar Linux LaTeX2ε environment and tools, then we can talk.

Most Linux distributions have teTeX as the default LaTeX environment. teTeX is no longer being updated, but is still widely used[1] on Linux systems. Since we want to reproduce a Linux environment on a Mac, we will work with teTeX. I assume you have Apple's original DVD Media and a stable Internet connection — these are the main requirements to achieve our goal.

2 Preparing Mac OS X for the traditional Linux tools

The Linux operating system uses the X Windows System[2] as the server to run graphic applications, and Mac OS X uses Aqua. It is possible to make Mac OS X

1. The migration to TeXLive in the Linux world is still in its early stages.
2. also known as X11, XFree86 or simply X

use an X server to handle graphical applications, such as Kile, originally written for Linux OS. The Apple Website[3] gives more details about X11 and Mac:

> (...) X11 for Mac OS X offers a complete X Window System implementation for running X11-based applications on Mac OS X. Based on the de facto-standard for X11, the open source XFree86 project, X11 for Mac OS X is compatible, fast and fully integrated with Mac OS X. It includes the full XFree86 4.4 distribution including a window server, libraries and basic utilities such as xterm. Native Aqua and X11 applications run side by side on the Mac OS X desktop. You can cut and paste between X11 and Aqua windows. You can minimize X11 windows to the Dock - even with the "Genie Effect." You use the Aqua window controls to close, minimize and zoom X11 windows. And of course, each X11 window comes with its own carefully rendered drop shadow. Experts may choose to replace the native Aqua window manager with their own familiar, standard X Window Manager.

In this sense, Apple provides the X Windows system for Mac OS X. X11 will need to be installed in order to get Kile up and running[4].

Xcode Tools are necessary and can be found on the same DVD set. Xcode Tools contains a set of developer tools provided by Apple, and some of its bundled tools are required in order to have the environment set.

3 Mac meets Linux! Mac meets LaTeX2ε! Thanks to Fink!

The next step is to install Fink. Fink is an open-source project that ports common Linux software to Mac OS X.

The Fink website[5] offers some background on the project:

> The Fink project wants to bring the full world of Unix Open Source software to Darwin and Mac OS X. We modify Unix software so that

3. Please visit http://www.apple.com/macosx/features/x11/
4. Please visit http://developer.apple.com/opensource/tools/runningx11.html.
5. Please visit http://finkproject.org/

> it compiles and runs on Mac OS X ("port" it) and make it available for download as a coherent distribution. Fink uses Debian tools like dpkg and apt-get to provide powerful binary package management. You can choose whether you want to download pre-compiled binary packages or build everything from source.

Installing Fink is easy but its usage can seem a bit complicated at first. If you are a Debian-based distro user you will find yourself at home. Fink uses apt-like commands to deal with its repository, so managing software dependencies is done automatically.

You can download Fink from the link:

http://www.finkproject.org/download/index.php?phpLang=en

Please note that you need to choose the right Fink .dmg package for your Mac OS X version. I have already successfully installed Fink on both PPC and Intel Macs under Tiger, but builds for earlier versions of Mac OS X are also available. Double click the installer and follow the instructions. You can choose the default options without fear. If you have any problems please refer to the Fink page referenced above.

After completing the Fink installation process, check under your Applications Folder to see if FinkCommander's icon was placed there. If not, just drag it from the Finder Window to the Application folder Finkcommander will be installed. Finkcommander is a very powerful tool similar to Ubuntu Linux's synaptic package tool. It can update, delete, compile, or just install software from binary files.

Like Debian distros, Fink categorises software in stable or unstable trees. The stable tree gathers software systems that were tested and approved as not prone to bugs. On the other hand, the Unstable tree has software that still needs testing to be considered stable. You can browse Fink's stable and unstable package lists on Fink's website, and learn more about their packaging system. You can also help developers and maintainers by testing software, like Kile, to get it listed as stable[6].

When you run Finkcommander, it lists all the available software from the stable tree.

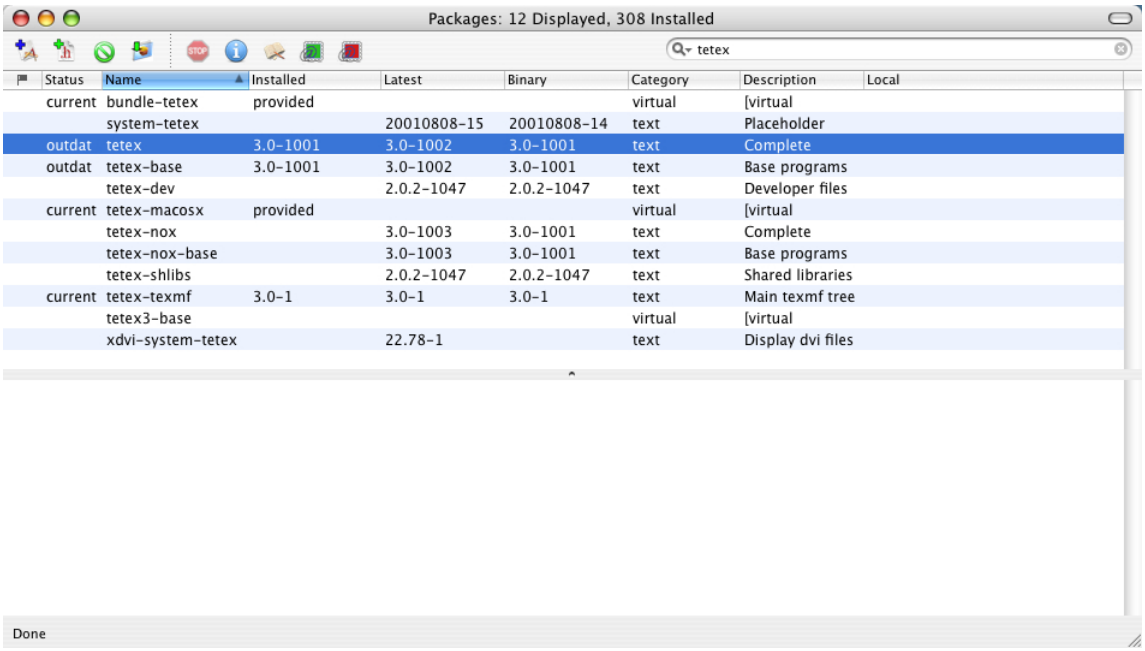
6. http://pdb.finkproject.org/pdb/index.php?phpLang=en

4 Installing teTEX

The complete teTEX packages are not available in Fink's stable software column, and thus we need to tell Fink it should also consider unstable packages. Fink developers encourage you to change the default setting to only install the applications you need from the unstable tree. Once the software you need is installed you should reset Fink to work with stable packages only.

Go to Fink Commander → Preferences → Fink and click on use "unstable packages". It will add the unstable packages repository to the list of repos to check. Save the preferences. Close Fink Commander and open it again so it will update the package list.

On Fink's Commander main window you have a Spotlight like search box. Type teTEX on it to see all teTEX packages available.



Choose teTEX from the list, go to the binary menu and choose install. If you want to do it the Linux way, just type on a terminal:

```
sudo apt-get install tetex
```

The teTEX environment is large and takes some time to download.

After the download finishes, your LATEX2ε environment will be installed and configured. LATEX2ε will be up and running[7]!

5 Kile? Finally!

While this article was being written, the latest version of Kile source code was available in Fink's unstable repository. But since there are no binary packages for it, nor for KDE[8] (which Kile relies on to run), Kile will need to be compiled from source. The compilation can be performed by FinkCommander. To compile Kile, FinkCommander will download all dependencies not already installed and compile all programs from scratch. It takes a long time to download, compile, and install all Kile dependencies. Thus it is better to install available binary packages needed by Kile, and then build and run it.

The available binary packages that can be installed (on my system) are listed below[9]:

> aspell aspell-dev aspell-shlibs audiofile audiofile-bin audiofile-shlibs autoconf cyrus-sasl2-dev cyrus-sasl2-shlibs daemonic db44-aes db44-aes-shlibs default-icon-theme docbook-dsssl-nwalsh docbook-dtd docbook-utils docbook-xsl esound esound-bin esound-common esound-shlibs expat expat-shlibs flex-devel freetype219 freetype219-shlibs gd2 gd2-shlibs gettext gettext-dev gettext-tools glib glib-shlibs glib2 glib2-dev glib2-shlibs gtk-doc help2man html-tagset-pm intltool jadetex lesstif lesstif-shlibs libart2 libart2-shlibs libgettextpo2-shlibs libidn libidn-shlibs libjpeg libjpeg-bin libjpeg-shlibs libkpathsea4 libkpathsea4-shlibs libmad libmad-shlibs libogg libogg-shlibs libpng3 libpng3-shlibs libtiff libtiff-bin libtiff-shlibs libusb libusb-shlibs libvorbis0 libvorbis0-shlibs libwww libwww-bin libwww-pm586 libwww-shlibs libxml2 libxml2-bin libxml2-shlibs libxslt libxslt-bin libxslt-shlibs m4 openjade openmotif3 openmotif3-shlibs openslp-ssl-dev openslp-ssl-shlibs opensp4

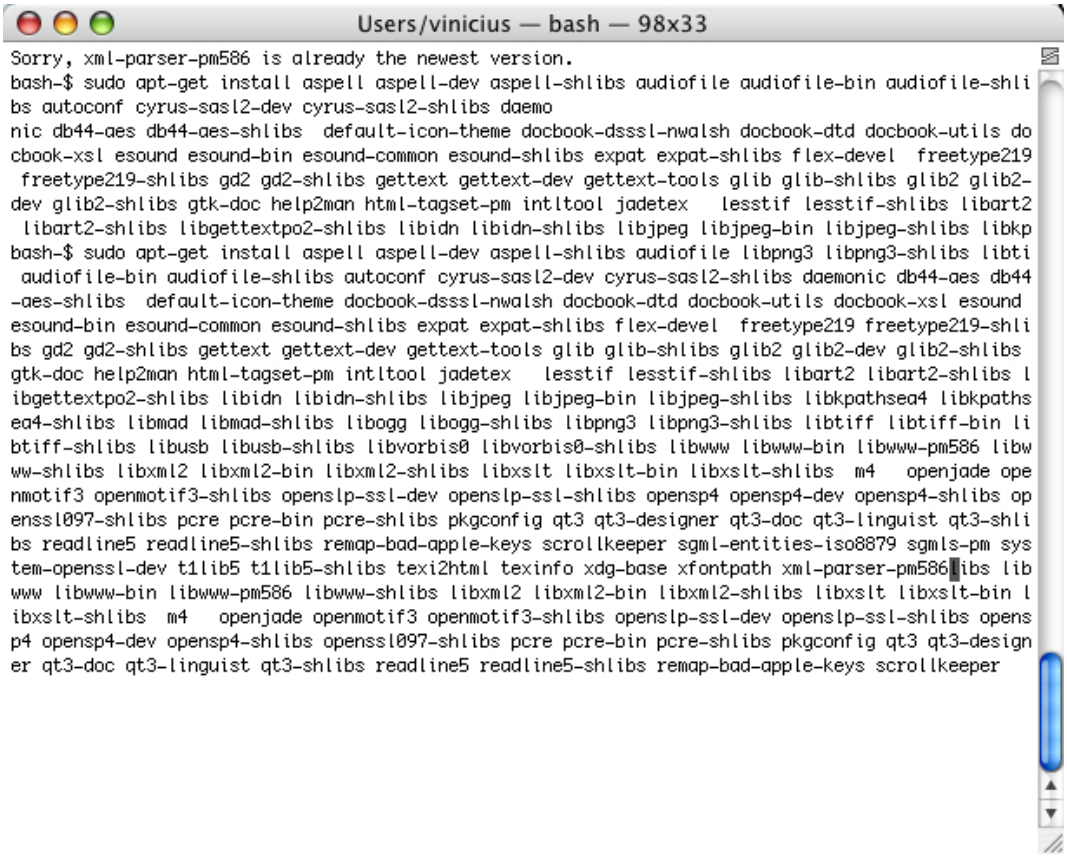
7. Emacs, vi or Xemacs editors have binary packages available! If you are familiar with them you can start writing right away!

8. Perhaps when you read this article new binary packages will have been made available! Fink is very a dynamic project.

9. This list will vary from system to system, and depends on many factors.

opensp4-dev opensp4-shlibs openssl097-shlibs pcre pcre-bin pcre-shlibs pkgconfig qt3 qt3-designer qt3-doc qt3-linguist qt3-shlibs readline5 readline5-shlibs remap-bad-apple-keys scrollkeeper sgml-entities-iso8879 sgmls-pm system-openssl-dev t1lib5 t1lib5-shlibs texi2html texinfo xdg-base xfontpath xml-parser-pm586

You can select each package on Fink Commander and choose binary → install or just "sudo apt-get install" the list above.



Setting up Kile will be faster after the packages are installed .

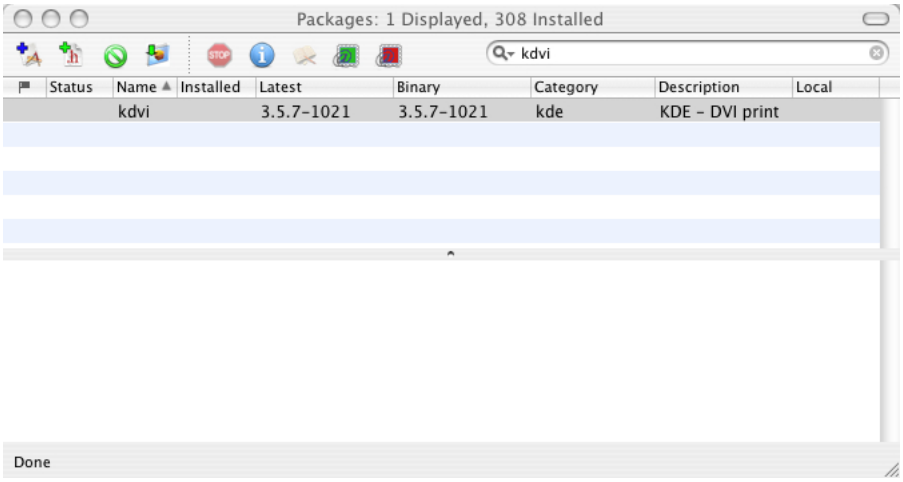
Select Kile from FinkCommander's packages list. Since it has no binaries we are forced to compile it and its missing dependencies. From the Source Menu choose the option Install. Answer the questions[10] and FinkComander will down-

10. If Fink asks you about teTEX or others, choose teTEX even if it is not the default choice.

load and compile Kile for you. It's also possible to achieve the same result by typing the following command:

```
fink install kile
```

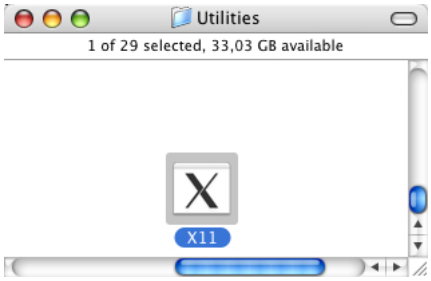
On both types of installations it will take a lot of time to complete the installation. Fink needs to download and compile a large number of KDE libraries. It also has to compile Kile. If Kile is not compiled on the first try, read the messages output by apt-get or by FinkCommander to understand why, and follow the hints to solve the problem[11]. It is also a good idea to install Kdvi and Kpdf; they are not required to run Kile but I am pretty sure you will need them. Without Kdvi, Kile will loose its preview capacity.



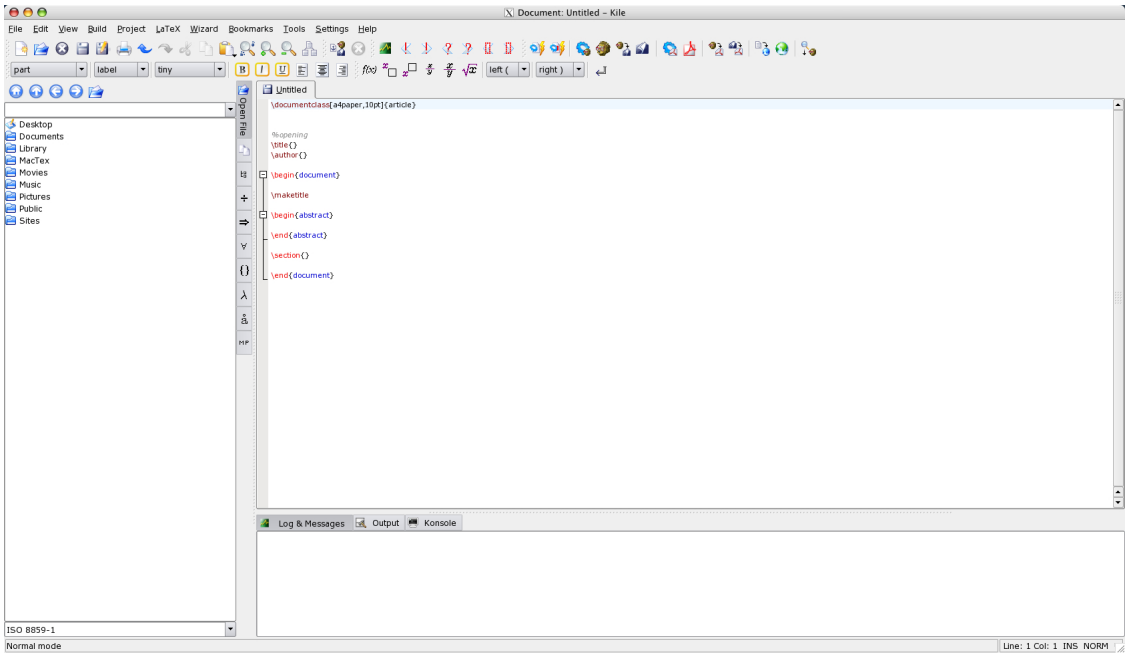
6 Getting Kile to run

First you will need to run X11. Go to the Finder, click Applications, Utilities and then double click X.

11. Most of time the problems you get are related to inconsistencies in the database that can be solved by typing the command "sudo apt-get update".



On the xterm window that will open you can run any X11 program. Type kile in the xterm window and it should start. It takes some time to load since all KDE libraries have to be loaded before Kile.



After all the downloading and compiling, the reward is that Kile up and running!

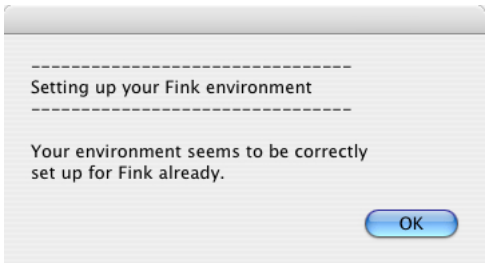
7 Troubleshooting

Sometimes Fink, X11 and Mac OS X integration do not run as smoothly as it should. Some of the main issues can be easily solved.

If your xterm complains about "command not found", set the path environment for Fink. On any terminal run the following command:

```
/sw/bin/pathsetup.sh
```

A window should pop up confirming that the PATH variable is now set[12]. All programs installed by Fink can now be accessed from the command line. Remember that if the application you want to use needs X11 to run, like Kile, you should start X11 first, and then start any other X program.



The system is configured, but right now you can only start Kile from X11's xterm. When it runs from Terminal.app a DISPLAY error appears. A DISPLAY variable needs to be set for Terminal.app to be aware that X11 is running. To get more power and be able to start Kile directly from Mac OS X terminal.app[13], you

12. if you still have problems please visit Finks Documentation page: http://finkproject.org/doc/users-guide/install.php#setup
13. Please remember that X11 must be running for Kile to work!

can create two little files in your home directory : `.bashrc` and `.bash_profile`[14], that will be used to define an environment variable to tell Terminal.app where X is running.

Create `.bash_profile` on your home directory with any text editor with the following content:

```
echo "Reading ~/
.bash\_profile"
source ~/.bashrc
```

Then create `.bashrc` with the following content:

```
echo "Reading ~/.bashrc"
# Initialize FINK if needed
if [[ ! -x $(which fink) && -d /sw/bin ]];then
source /sw/bin/init.sh
fi
# Set the DISPLAY variable -- works for Apple X11 with Fast User Switching
if [[ -z $DISPLAY && -z $SSH_CONNECTION ]]; then
    disp_no=($( ps -awx | grep -F X11.app | awk '{print $NF}' | grep -e ":[0-9]"))
    if [[ -n $disp_no ]];then
        export DISPLAY=${disp_no}.0
    else
        export DISPLAY=:0.0
    fi
    echo "DISPLAY has been set to $DISPLAY"
fi
```

8 Conclusions and alternatives

Setting up this environment is not easy nor fast, but the reward is a perfect working Kile/LaTeX2ε environment.

14. Bash code variation from http://xanana.ucsc.edu/xtal/x11.html. Please visit the page for other unix shell examples. You cannot forget the dot in front of the file names, otherwise it will not work.

It's likely that in the near future the same result can be accomplished in a simpler way, since KDE4 applications are being ported natively to Mac OS X and will run without the need of Fink[15]. Hopefully, Kile will soon be available on a "download and install" basis.

Running LaTeX2ε on a Mac can also be achieved with a combination of TUG-MacTeX[16] and TexMaker[17]. This is an easy way to install an environment, with major similarities to teTeX and Kile. I like this option and think that most people familiar with Kile will choose this as the natural transition path from the Linux/KDE environment to Mac OS X.

If you want to contact me, please visit one of my blogs: http://provenzano.wordpress.com (english) or http://camelomanco.com (portuguese).

Happy writing!

15. For more information visit:http://dot.kde.org/1168899755/

16. The LaTeX environment maintained by TeX Users Group. It is easy to install through a .dmg package. Please visit http://www.tug.org/mactex/ for more information.

17. TexMaker was the first name of Kile, and its developer, Pascal Brachet, was the original Kile developer. TexMaker runs on Mac, Linux, and Windows and is almost as powerful as Kile. Please visit http://www.xm1math.net/texmaker/ for more info.

THE PracTEX Journal

Will TeX ever be wysiwyg or the pdf synchronization story

Jérôme Laurens

Abstract

Why editing Plain, LaTeX or ConTeXt documents can be such a pain ? Of course, we can use dedicated text editors and environments that are very handy, but we are far from the efficient graphical user interface of a modern word processor. In this article, we will point out some of the reasons why TeX has no *wysiwyg*(What You See Is What You Get) user interface and we will discuss the possible remedies. One of them is the pdf synchronization implemented in the *pdfsync* package. This technology will be explained, we will see its benefits and its limitations. Finally, we consider routes towards a better user experience. Here, we are mainly concerned with LaTeX and PDF (for Portable Document Format) is a technology invented by Adobe® in 1990 to create and share documents. output.

After a PhD in applied mathematics, the author teaches mathematics in the french university of burgundy. He uses TeX both for personal and mathematical purposes, while pestering against its primitive user interface. So, he contributes to improve the TeX user experience on Mac OS X, spending too much time according to his wife and two young children. You can reach Laurens at jlaurens@users.sourceforge.net

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

Will TEX ever be *wysiwyg* or the pdf synchronization story

Jérôme Laurens

Email jlaurens@users.sourceforge.net
Website http://itexmac.sourceforge.net/pdfsync.html

Abstract Why editing Plain, LaTeX or ConTeXt documents can be such a pain ? Of course, we can use dedicated text editors and environments that are very handy, but we are far from the efficient graphical user interface of a modern word processor. In this article, we will point out some of the reasons why TeX has no *wysiwyg*[a] user interface and we will discuss the possible remedies. One of them is the pdf synchronization implemented in the `pdfsync` package. This technology will be explained, we will see its benefits and its limitations. Finally, we consider routes towards a better user experience. Here, we are mainly concerned with LATEX and pdf[b] output.

[a] What You See Is What You Get

[b] PDF (for Portable Document Format) is a technology invented by Adobe® in 1990 to create and share documents.

1 Time and memory are our enemies

When TEX was designed by D.E. Knuth some 30 years ago, personal computers were rather limited, their memory and CPU speed measured in kilobytes and megahertz. At that time, instantaneous typesetting was a nonsense unless we drop a great amount of both quality and efficiency. Indeed, TEX is well known for its very clever algorithms to break lines into paragraphs and paragraphs into pages, but these do need complicated calculations. Also, the macro feature is absolutely essential to manage sectioning commands, table of contents for example, but all this consumes lots of RAM. Long time TEX users will certainly remember the minutes spent to typeset just couples of pages, when patience was the rule. In fact, we can understand that to make the TEX program reasonably usable, things had to be dramatically optimized and there was no other choice than completely ignore graphical user interfaces (even if they were not yet popular).

Now that gigabytes and gigahertz are the standards, are we still limited by time and memory?

2 Typesetting can be extremely complex

Macro and page breaking illustrate how complicated can be the typesetting mechanism. It is obvious that changing a macro used throughout a TEX input file `foo.tex` involves changes throughout the output document (in general `foo.pdf`). In the same way, changing just a word can also have repercussions much more important than expected at first glance. If the size of the word changes, the lines might break differently, then the paragraphs, then the pages. The figures, tables, and more generally all the floating material, might also see their location modified. If the input file uses references to page numbers in the output document, then both the first and last pages can see their contents altered, and all the other pages consequently by some chain reaction. Finally, a slight change in the input file can lead to modifications spread over many pages of the output document, before and after what is changed.

This clearly shows that we cannot assume changes to be local, any modification is potentially global, so the only solution to be up to date is to typeset the whole input file from scratch.

3 A *wysiwyg* editor would increase complexity

Suppose you want to correct a misspelled word and turn an "a" into a "z", you just would like to select the "a" into the displayed output document then press the "z" key. How could a *wysiwyg* editor handle this task? First it would have to know exactly what in your input file generated this "a". There stands the first problem: the traceability of each character, each macro. Unfortunately, when TEX translates the input material into its own private internals, it does not remember the whole necessary information: the "a" character is turned into something called a node, the container file name and the file number being recorded for debugging purposes[1], but not the column number. Recording this last information would require more memory and also some tricky character index management.

1. This is the information used when, in case of error, you ask TEX to edit by entering the "e" character in the terminal.

But this is not the only problem: the traceability information should be remanent, which means that inside the pdf output document, any single character should keep track of the input file name and character index it was originated from. Not being a part of the natural pdf features, this would require dangerous acrobatics.

So far we reached a first conclusion: in order to have a *wysiwyg* TEX editor, one would need a huge amount of memory, speed, and engineering. We are constrained to reduce the ambitions.

4 Visual editors

Editors like LyX[2] and Scientific Word[3] provide us with an advanced graphical user interface, which resembles the output document for the main points. The sections are properly numbered and highlighted, the mathematical stuff is displayed with a dedicated font, colors are supported. All these visual effects have more meaning than the underlying command, and it is possible to edit text in place: this is a real benefit.

But these editors are not able to manage all TEX input files: for performance reasons, they only support a strict subset of TEX or LATEX. In general, this is not really annoying, but when it comes to advanced typesetting techniques, all the benefits are simply lost.

Another design is used by AUCTEX[4], the emacs extension dedicated to LATEX. It has a partial preview mode, where some critical parts of the input file are replaced by the real output. For example, displayed mathematical equations are replaced by the image obtained when typesetting only the equation. Of course, only the original input file is editable.

One can say that visual editors are just standard text editors with more or less extended capabilities. Let us explore now a more general concept to allow easy navigation between text input and pdf output.

2. "LyX is an advanced open source document processor running on many Unix and some non-Unix platforms" according to http://www.lyx.org.

3. A commercial document processor largely based on LATEX running on Windows™. See http://www.mackichan.com/.

4. AUCTEX is an extensible package for writing and formatting TEX files in GNU Emacs and XEmacs. See http://www.gnu.org/software/auctex.

5 The pdf synchronization

The first approach is due to Piero d'Ancona who had the idea of exploiting rather recent features of Han The Thanh's `pdftex`[5]. Whereas it was not possible to have complete traceability of all the characters, he thought that traceability of critical parts would be helpful, just like former `src-ltx` package did regarding DVI. Then, with the author, he could elaborate a LaTeX package, `pdfsync.sty`[6], that would automatically insert some properly designed control sequences in any input file, and create anchors in the output document for which the trace would be recorded.

More precisely, when using the `pdfsync` package in `foo.tex`, the `pdftex` engine will create a `foo.pdfsync` file containing both geometrical and traceability information. Let us see how it works on a concrete `foo.pdfsync` example:

```
1   foo
2   version 0
3   l 0 13
4   l 1 19
5   l 2 24
6   l 3 33
7   l 4 35
8   l 5 13
9   [...]
10  l 431 134
11  l 432 134
12  [...]
13  s 1
14  p 430 2368143 54651247
15  p 398 21086469 3154577
16  [...]
17  p 431 2368143 402063
18  l 433 134
19  [...]
20  l 464 173
21  [...]
22  l 527 215
```

5. `pdftex` is a concrete implementation of TeX with pdf output.
6. See `http://itexmac.sf.net/pdfsync` for more details and a complete set of specifications.

```
23   s 2
24   p 525 2368143 54651247
25   [...]
26   p 464 7149061 19993810
27   p 475 14470540 19043538
28   [...]
```

For direct (or forward) pdf synchronization, we start from a line number in `foo.tex`, say 173, and find where is the corresponding output. From the 20th line of `foo.pdfsync`, we can see that the anchor numbered 464 was registered for that line 173. Then from lines 26 and 23 of `foo.pdfsync`, it corresponds to a point of the second page of `foo.pdf` with TEX coordinates 7149061 19993810.

For reverse pdf synchronization, we can see that the point with TEX coordinates 2368143 402063 in the first page of `foo.pdf` was created with material from line 134, due to lines 13, 17 and 10 in `foo.pdfsync`.

To generate `foo.pdfsync` when typesetting, each time `pdftex` encounters maths, sections, boxes and other critical material, it appends to `foo.pdfsync` a "`l`" line like "`l 464 173`" where 464 is a unique anchor identifier and 173 is the current line number. When a new page is completely laid out, a line like "`s 2`" is appended to `foo.pdfsync`. It was necessary to manage the page indexing because page labeling or numbering need not be continuous in LATEX. At that stage only, the geometrical information is available, so "`p`" lines like "`p 398 21086469 3154577`" are appended to `foo.pdfsync` for each previously recorded anchor. The package also has to properly manage included files to keep track of the container's file name, but this is not illustrated here.

Observing more precisely this example will convince even more that typesetting with TEX is a fairly complex mechanism. We can see that the "`l`" lines are written synchronously and increasingly, but the lines number they are refering to do not appear increasingly. This is most certainly due to macro extension which might not occur at parse time but sometimes later, at TEX's will. The "`p`" lines are not even written synchronously, just like if it were totally random. This is quite natural because all graphical objects in a laid out page need not be displayed in any particular order. We can also observe that sometimes no "`p`" line correspond to some "`l`" line, or multiple "`p`" lines do. Similarly, multiple "`l`" line can correspond to a unique location in a given page. Some temporary objects migh be created but never displayed, other might be used more than once. Thus pdf synchronization is not so simple, mainly because in case of doubt there are no criteria to make a good choice, this explains why the technology sometimes fails. However, people will appreciate the difficulties to

design and implement the pdfsync feature.

The forthcoming limitations are not safe. The fact is that postponing write orders in pdftex until the page is completely laid out sometimes changes vertical spacing, paragraph or page breaking. Moreover, pdfsync macros can break existing packages. For these reasons, pdfsync should only be used during production stage.

Actually, only few tools are supporting pdfsync technology, with various levels of efficiency. AUCTEX is capable of forward synchronization towards xpdf[7], with an accuracy in the range of ± 1 page. In the Unix and Linux worlds, no other tool supports yet pdfsync, neither for direct nor reverse synchronization. On Mac OS X[8], two popular PDF viewers support reverse and forward synchronization, Skim[9] and iTEXMac2[10]. TEXShop[11] also have some kind support for synchronization as long as no external editor is involved.

6 The pdf synchronization by text extracts

When the Tiger version of OS X was released, developers could easily extract text from pdf documents and inspect their contents. Then TEXShop provided a synchronization method based on exact concordance. If the same sequence of characters is found both in the input and in the output, then it can be used to navigated between them. It has been chosen a sequence length of approximately 20 characters, less would lead do multiple choices and dilemnas, more would suppress concordances and reduce the efficiency.

Some people think that such a technology can be a replacement for pdfsync. They are mistaking: both should be used concurrently, this one for pure text and pdfsync for math and macros.

7. xpdf is a popular PDF viewer on Unix.
8. Apple System™ operating system based on Unix
9. See http://skim-app.sourceforge.net/.
10. iTEXMac2 has the most advanced implementation, and this is not due to the fact that the author is also iTEXMac2 developer. See http://itexmac.sf.net
11. The most popular TEX environment on Mac OS X. See http://www.uoregon.edu/~koch/texshop.

7 The pdf synchronization by words

To improve even more both accuracy and efficiency, a new approach is also using the file contents to synchronize. It is based on a simple observation: the input file contents and text extracted from the pdf output are not so far from each other. The main differences concern macros, line breaks and math, but the words remain essentially the same.

Thus, how can we achieve synchronization ? First of all, one cannot use the `diff` utility as is due to performance reasons, but the idea can help. Given a word in `foo.tex`, we find the two preceeding words and the two following words, wiping out macros, line breaks, and maths, but taking accents into account. Then we find in the pdf output all the text extracts that contain these five words in exactly the same order, starting on the first and ending on the last. In case different extracts are found, only the shortest ones are kept. If there is still a choice to be made, the `pdfsync` technology helps. This is extremely efficient for direct synchronization, and works similarly from output to input.

This is the most accurate and efficient method for pdf synchronization, in both directions, and iTEXMac2 is the only application implementing it. Instead of asking to display the point for a given line in a given input file, you just ask to display the point for a given character in a given text extract of a given input file. Then, iTEXMac2 will parse the words and take care of everything to properly highlight the corresponding character in the pdf document.

8 Can't we do better?

Concerning `pdfsync`, the author already received significant contributions from very qualified people, mainly Hans Hagen and David Kastrup, in order to publish the package on CTAN. But the whole technology should definitely be embedded directly in the `pdftex` engine and the end user should not have to worry about it at all. This would definitely prevent polluting the pdf output with erroneous vertical spacing and any other undesirable or even fatal side effects. Moreover, this would make `pdfsync` available indifferently for TEX, LATEX, ConTEXt and any other format.

The work is in progress and will certainly be complete in a few months. There is not great risk in guessing that the tools on Mac OS X will quickly be adapted to this new kind of synchronization, but the same does not hold under Unix and Linux.

More generally, it is actually quite impossible to achieve the ultimate goal of complete *wysiwyg*. If extending TEX internals to really support character traceability is certainly feasible, TEX as a typesetting engine remains of an unbearable slowness. For that point, improvements would possibly come from incremental typesetting: when TEX processes an input file, it should remember what is has done before and modify only what really needs to be modified. This could be done by splitting TEX into a macro manager and a typesetting server. As a consequence, one would also have to separate typesetting macros that drive the layout from structuring ones like `\section{...}`.

All this implies radical changes in TEX internal architecture. Is it feasible? Only the future will tell us.

THE PracTEX Journal

TpX -- a drawing tool for LaTeX

Alexander Tsyplakov

Abstract

This article describes TpX, a lightweight, easy-to-use graphical editor for Windows platform, presents guidelines for its use and discusses some features and limitations.

Alexander Tsyplakov teaches microeconomics and econometrics at the Economic Department of Novosibirsk State University. His interest in LaTeX was inspired by writing a textbook in microeconomics.

You can reach Alexander at tsy@academ.org

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

TpX — LaTeX-oriented graphical editor

Alexander Tsyplakov

Email tsy@academ.org
Website http://www.nsu.ru/ef/tsy/about_en.htm
Address Novosibirsk, Russia

Abstract This article describes TpX, a lightweight, easy-to-use graphical editor for Windows platform, presents guidelines for its use and discusses some features and limitations.

1 Introduction

TpX is a lightweight, easy-to-use graphical editor for Windows[1] [4]. The tool allows users to draw and include their drawings in LaTeX files. In addition, the tool can be used as a stand-alone editor for vector graphics.[2]

The tool supports both the (pdf)LaTeX→DVI→PS and pdfLaTeX→PDF ways of producing documents. The choice between two regimes is governed by the `ifpdf` package. Thus the same drawing can be used in a LaTeX document in both regimes.

TpX can import EMF/WMF pictures created by other Windows applications, including many applications producing scientific graphs. It also can import simple SVG pictures. So TpX can be used as an EMF-to-any and a SVG-to-any converter.

2 How it came to be

I am engaged in writing a large microeconomics textbook (more than 1000 pages). It is a long-term project of two co-authors and myself which has continued for

1. See below about the cross-platform version of TpX which is under development.
2. Vector graphics use geometric shapes based on analytical formulas to represent graphical information, which is different to bitmap or raster graphics that represent arrays of pixels.

several years and is not finished yet. At one stage a decision was made to switch from Word to LaTeX because of the high typesetting quality of LaTeX and the preferences of a co-author. Consequently, there were two main problems. The first one was to convert the documents from Word to LaTeX. The second one was to include a large number of graphical illustrations into the resulting LaTeX document in a form which allows further editing.

The graphic edition workflow using MS Word was relatively fast and easy. Double click the picture, edit with the mouse, close and here you are. In LaTeX, it is a little bit different because it has no built-in WYSIWYG graphical capabilities like MS Word. Thus, I was looking for some tool which would allow me to edit illustrations in a similarly rapid manner with the mouse.

It turned out that there are many different choices of vector graphics programs around. Some of them I knew when starting TpX (TeXCAD [7], jPicEdt [8] and many others). Some alternatives came to my sight only recently (Ipe [9] and LaTeXDraw [10]).

There exist several drawing programs based on the standard LaTeX `picture` environment (sometimes enhanced by additional packages). This environment produces what can be called pseudo-graphics. By default, the result is very primitive and is not usually suitable for a published book. (However, it is quite portable, as it uses only LaTeX fonts for rendering graphics and do not need special drivers.)

At the other extreme there is Inkscape [6]. It is a powerful vector graphics program based on the SVG format, but it is large and not at all LaTeX-friendly.

jPicEdt is somewhere between these two extremes. As well as the LaTeX `picture` environment enhanced by the `eepics` package it can also output PSTricks code, which is suitable for use in a published book. However, jPicEdt is based on Java, which is not very convenient (for example, it precludes use of the program on computers which do not have Java installed). And most importantly, it is not possible to use graphics produced by jPicEdt directly with pdfLaTeX to get PDF.

Thus, I decided to write my own tool. The following principles were behind the development of TpX:

- tight integration with LaTeX (for example, use of LaTeX for rendering text labels),
- support for multiple LaTeX-related formats (including formats compatible with the pdfLaTeX→PDF way of producing documents),

– small size and fast loading with a small memory footprint,

– usability,

– simple and open format for storing drawings,

– easy import and export from/to important formats of vector graphics.

The development of TpX was very helpful for writing of the above-mentioned textbook. TpX writes graphics in various formats and is very usable. After all, converting the illustrations from MS Word to TpX[3] has provided advantages for preparing the textbook for publication because MS Word encapsulates pictures in the document. This is a problem when you need to unify their style and appearance. I have used TpX for many other purposes. For example, I have included statistical diagrams into beamer presentation or printed pictures for my little daughter to colour.

3 The direct use of TpX

The TpX graphical user interface is similar to other Windows programs. Figure 1 shows a TpX screenshot. TpX allows you to do the most common vector drawing tasks (like creating, moving, copying, reshaping of graphical objects, changing colour and so on).

TpX is not fully WYSIWYG. The screen representation of a drawing is somewhat different from the resulting graphics included into LaTeX document. For example, mathematical formulas would be seen only in the LaTeX document. Having a system font with a rich set of symbols one can make text labels more similar to LaTeX output by including `&#xHHHH;` code into the label text where `HHHH` is hexadecimal unicode representation of a symbol.[4] Please, observe Greek letters and degree symbols in Figure 1.

A TpX drawing is stored in a file (with extension .TpX), which is intended for inclusion into a parent LaTeX document with the `\input` command. Figure 2 is the output from TpX drawing. It was included into this PracTeX article using

3. The illustrations were saved as RTF using a Visual Basic script. Then a Python script was used to convert RTF to some XML representation and eventually to TpX.

4. This is important when exporting TpX drawings to non-LaTeX formats like SVG or EMF.

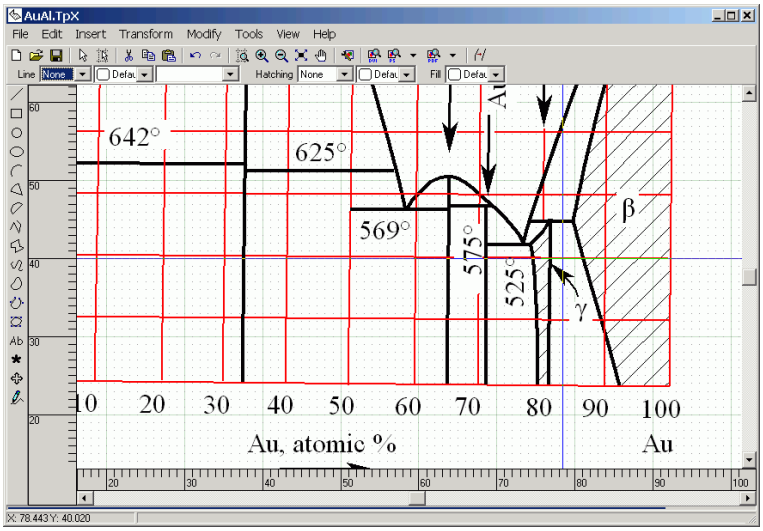


Figure 1: TpX screenshot

`\input{AuAl.TpX}`. Note that it uses the same URW Palladio font family as the article itself.

Let me describe the workflow to use TpX for creating a drawing from scratch. Suppose that one is going to create a drawing with the filename "Foo.TpX" in subdirectory `pics/`.

– In the text editor used to edit the parent LaTeX document add the following line to the document:

  ```
  \input{pics/Foo.TpX}.
  ```

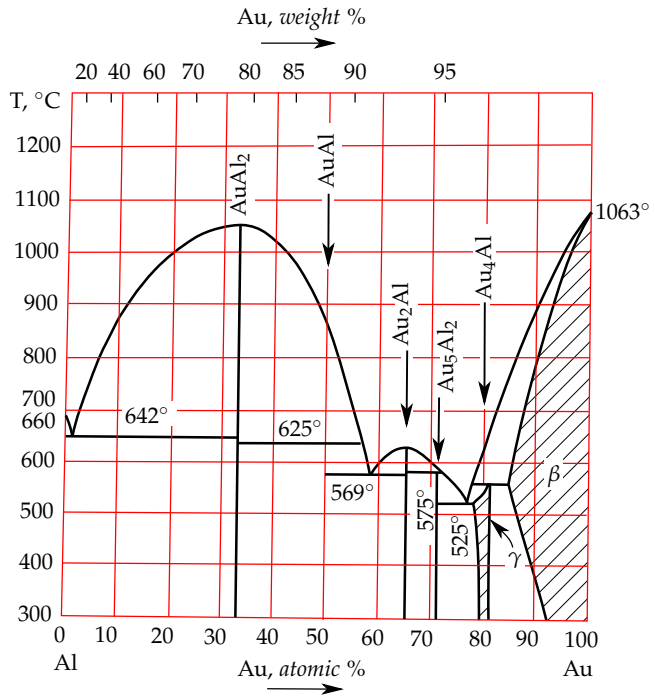
  Save the document.

– Move the cursor to the added line and press the hot key.[5] The TpX program will be started with the following command-line parameters:

  ```
  -i<filename> -l<line number>
  ```

  where <filename> is the name of the parent LaTeX document and <line number> is the current line number. TpX will scan the LaTeX document and find

5. I assume that there is already a hot key for running TpX. This requires an editor which can have a hot key for running an external program and which is able to pass the current line number as a command-line parameter to an external program. The TpX distribution includes support macros for the popular WinEdt editor.



Aluminium-gold compounds diagram

Figure 2: TpX output

the case of `\input{<filename>.TpX}` which is closest to the given line (that is, `\input{pics/Foo.TpX}`). This will open Foo.TpX in TpX.

– Fill the newly created drawing with graphical content and save it in the `pics/` subdirectory.

– Close TpX and return to the editor.

The most time-consuming part of the procedure is preparation of the drawing. The other operations are very fast. TpX in combination with a smart text editor saves time which is typically spent on finding a graphic on the disk and opening it in a graphical editor. Time savings are even more pronounced if only some minor corrections are needed for an existing drawing.

The program's own data are put into TEX comments (lines starting with "%") so that the drawing could be loaded into TpX and edited again. (The internal TpX format is based on XML and can be understood and edited easily). These

data are not seen by the LaTeX program. After the comments TpX writes normal commands that are seen by LaTeX. Depending on the required output format this code would include direct drawing commands (like commands from the `picture` environment, TikZ commands) or an `\includegraphics` link to an external file created by the program.

Code inside a typical TpX file looks like this:

```
%<TpX v="4" ArrowsSize="0.7" ... ...
%  <polygon fill="whitesmoke" ... ...
%  ... ... drawing data ... ...
%</TpX>
\begin{figure}
\centering \ifpdf
  ... ... code for PDF ... ...
\else
  ... ... code for DVI->PS ... ...
\fi
\end{figure}
```

The output format is chosen separately for PDF and DVI→PS. TpX provides several output formats:

- Formats for which graphics are drawn using direct LaTeX code: LaTeX `picture` environment, PSTricks [13], PGF and TikZ [14].
- Formats for which graphics from an external file generated by TpX is included into the document with the help of the `graphics` package [5]: EPS, PDF, MetaPost EPS [15] and PNG.[6]

EPS and PDF are TpX defaults for the DVI→PS and PDF regimes respectively. With these output formats LaTeX text labels are overlaid above included EPS (PDF) graphics using the LaTeX `picture` environment.[7] This allows using the same font for text labels as is used in the parent document. See TpX help for more information about output formats.

6. Please note that PNG is, as opposed to all other formats mentioned, a bitmap (or raster) format rather than a vector format.
7. This is similar to how the `overpic` package [11] works.

4 Porting graphics into LaTeX using TpX

It is common for a Windows program which produces some kind of vector graphics to allow copying a metafile image (EMF) to the clipboard and/or exporting it as an EMF file. TpX can import this (though not 100% correctly) for subsequent inclusion into LaTeX document. Old-style Windows Metafiles (WMF) are imported by converting them to EMF. In most cases the result of importing is nice, though often the imported picture needs some manual editing. TpX can also import SVG images. Note that the SVG format is extremely rich (see [1]), so TpX can understand only a basic subset of it.

For example, in order to include an Excel diagram into a LaTeX document one can do the following:

- Copy the Excel diagram to the Windows clipboard. This copies also an EMF image representing the diagram.
- Use TpX to capture the EMF image from the clipboard ("Tools" > "Capture EMP") and save it to an EMF file.
- Import this EMF file into TpX.
- Edit the picture. Sometimes scaling of the picture's physical size is needed (for example, in order to satisfy page size constraints).

An example of such a use is given in Figure 3.

Of course, there are other ways to include external graphics into LaTeX document. The most popular procedure is printing graphics to an EPS file with a PostScript printer driver. However, the TpX way gives important advantages:

- TpX can add TeX formulas to graphs.[8]
- It is possible to edit imported images and correct their deficiencies. For example, TpX can help to shift misplaced text labels.
- The graph imported through TpX uses the same fonts as LaTeX document itself, which adds to the typographic quality of the document. (For example, Figure 3 uses the sans serif font which is implied by the PracTeX article style[9]).

8. Well-known alternatives are `overpic` [11] and `psfrag` [12] packages which allow adding text labels to included EPS files by means of LaTeX commands.
9. The graph was included with `{\sffamily\input{images/ExcelGraph.TpX}}`.

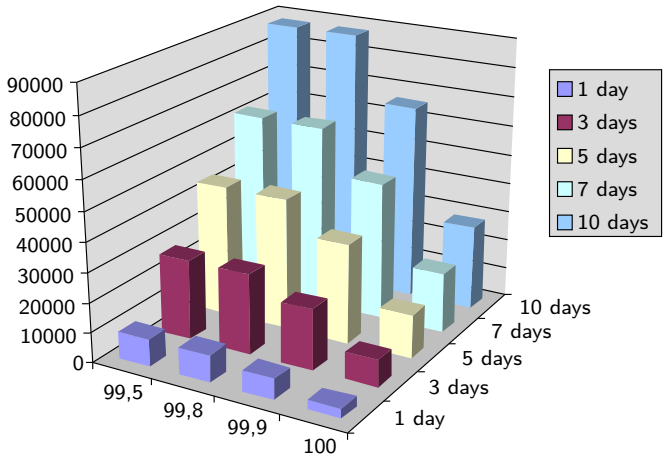


Figure 3: Excel graph imported into TpX

To see what the TpX capabilities and limitations are in this respect, have a look at 'Demonstration of TpX import capabilities' [2]. This document provides examples of import from many well-known as well as lesser known Windows programs.

5 Drawing with the mouse vs. drawing by code

Some people prefer to draw visually and interactively using the mouse while others prefer to draw logically (programmatically) by providing coordinates of graphic objects directly (or using geometric concepts like line crossing). TpX is intended for people of the first type. TpX is most useful when one has some visual scheme in one's head and wants to implement this scheme quickly as a sketch.

Figure 4 shows the typical task which is performed using the mouse and which requires the human eye to control the accurateness. This is unlike a clever graphical programming language which could make it possible to say: "Draw a line of this length through these two points".

Note that TpX's viewport can be zoomed easily using the mouse wheel or hot keys. This allows accurate visual placement of lines for most common purposes. The human eye cannot distinguish small nuances in the positions of graphical

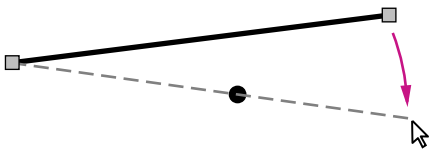


Figure 4: Passing a line through a point

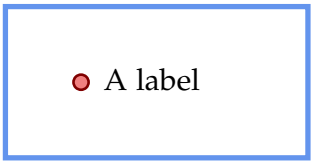


Figure 5: Simple drawing for TikZ example

objects anyway so it is not usually necessary to place them 100%-exactly.

For more accurate placement with the mouse additional features can be used: snap to grid and angular snap. Snap to grid mode allows to restrict oneself to "round" coordinates. Angular snap mode helps to restrict the angle when moving something with mouse (to a multiple of 45° in TpX).

The fact that TpX is mouse-driven does not mean that it is impossible to specify exact coordinates in TpX. Open the properties window of a graphical object, then press the "Points" button. This will open a table editor for editing coordinates. Consider, for example, generating coordinates in a spreadsheet application and then pasting them into this table editor.

An interesting point is that there is actually no antagonism between using TpX and drawing by code. TpX can be used to generate PSTricks, METAPOST or TikZ code. Generated code can then be included into LaTeX documents and edited with the keyboard in a text editor together with the document.

Figure 5 shows a simple example drawing. When saved in a TikZ format it gives the following code:

```
\begin{tikzpicture}[x=1.00mm, y=1.00mm, inner xsep=0pt, inner ysep=-1.2pt]
\path[line width=0mm] (-2.00,8.00) rectangle +(44.00,24.00);
\definecolor{L}{rgb}{0.392,0.584,0.929}
\path[line width=0.70mm, draw=L] (0.00,10.00) rectangle
+(40.00,20.00);
\definecolor{L}{rgb}{0.502,0,0}
```

```
\definecolor{F}{rgb}{0.941,0.502,0.502}
\path[line width=0.35mm, draw=L, fill=F] (10.00,20.00) circle
(1.00mm);
\draw(13.00,20.00) node[anchor=west]{\fontsize{11.38}{13.66}\selectfont
A~label\strut};
\end{tikzpicture}%
```

Of course, this is not very suitable for logical editing as it is. One would prefer to do some further editing. First, it is better to replace `\fontsize...\selectfont` with something like `\large`. Second, one should prefer the use of logical commands for line width (like "very thin" or "ultra thick") to the direct declaration of physical size (like `line width=2mm`). See [3] for more on TikZ.

TpX's own data are in XML format and are written inside TEX comments as plain text. This makes it straightforward to program a script to generate a TpX drawing. I wrote a simple Python module, TpXpy, which can help generate TpX drawings. Here is a sample TpXpy code:

```
pic = TpXpic()
pic.addPolyline(((10,0), (30,-20), (15,-30)))
pic.fill('mintcream')
pic.lw(1.5)
for i in range(5):
  pic.addLine(i*10 + 50,-0.7,i*10 + 50,0.7)
  pic.li('dash')
pic.SaveToFile('Foo.TpX')
```

Any other scripting language would be as good for the task as Python. Of course, any of the formats produced by TpX can be generated using a script. However, generating TpX code has some advantages. First, it is possible to obtain output in many different formats from the same source. Second, the result can be further edited by mouse if needed.

6 Ideas for automating editing tasks

If a document contains just a couple of illustrations then modifying them is not a terrible problem. However, a document with dozens of illustrations requires special attention. For example, one can come upon a need for uniform scaling of the illustrations to fit a different paper size or changing colours throughout.

This section gives some ideas, from my own experience, to help you maintain illustrations in such documents.

- One can write a script gathering all `\input{<filename>.TpX}` commands in a temporary document. Then one can quickly revise all illustrations one by one switching between LaTeX editor, DVI (PDF) previewer and TpX.
- In order to make some uniform correction in a series of drawings all one needs to do is to use a simple search and replace utility. For example, one can change output format for DVI from PGF to PSTricks by replacing all cases of ␣TeXFormat="pgf" by ␣TeXFormat="pstricks".[10] The drawing has to be refreshed to reflect changes. This can be done by running the following command:

  ```
  TpX.exe -f<filename>.TpX -o
  ```

  A renew_all.py script which is distributed with TpX shows how one can refresh all drawings in a specified directory.
- A more advanced way of automatically correcting a series of drawings is to use a scripting language with an XML DOM library. As was already mentioned, the TpX format for storing drawings is based on XML. To read the corresponding XML document one has to read all lines staring with "%" from the TpX file and strip the "%" symbol from each line. This will give an XML document. The XML document is then converted to a DOM tree. After editing the DOM representation one has to go this way backwards by first getting XML document and then prepending each line by the "%" symbol. Finally, the resulting TpX file has to be refreshed.

7 Further Improvements

Currently TpX is a one-man project. However it is placed at SourceForge[11] under the GNU Public License and anybody with sufficient knowledge of Object Pascal is invited to participate.

10. Note the space char before TeXFormat; it is needed to distinguish TeXFormat and PdfTeXFormat.
11. http://sourceforge.net/.

Although TpX was originally created for the Windows platform using Delphi I recently reshaped it for Lazarus, a cross-platform clone of Delphi. The Lazarus variant of TpX is at alpha stage and lacks some functionality of the Delphi variant (like export of PNG and EMF images). Also it was not yet fully adapted for other platforms (Linux or OS X). In the future, this will hopefully result in a usable product.

I conclude this section listing some of the planned enhancements from my todo file: · group/ungroup; · grouping objects into compound paths (to get disjoint shapes and shapes with holes); · bitmap object; · diagram object for creating simple plots; · adding custom graphical objects from a library; · additional properties for graphical objects (miter limit, line caps, etc.); · simplifying of Bezier paths; · layers; · codepages.

References

[1] W3C. Scalable Vector Graphics (SVG). XML Graphics for the Web.
http://www.w3.org/Graphics/SVG/

[2] Tsyplakov, Alexander. Demonstration of TpX import capabilities, January 31, 2006.
http://tpx.sourceforge.net/TpX-Demo.pdf

[3] Tantau, Till. The TikZ and pgf Packages. Manual for Version 1.01.
http://sourceforge.net/projects/pgf

[4] TpX.
http://tpx.sourceforge.net/, CTAN:graphics/tpx/

[5] `graphics` and `graphicx` packages.
CTAN:macros/latex/required/graphics/

[6] Inkscape.
http://www.inkscape.org/

[7] TeXCAD.
http://homepage.sunrise.ch/mysunrise/gdm/texcad.htm

[8] jPicEdt.
http://jpicedt.sourceforge.net/site/index.php,
CTAN:graphics/jpicedt/

[9] Ipe.
http://tclab.kaist.ac.kr/ipe/

[10] LaTeXDraw.
http://latexdraw.sourceforge.net/

[11] `overpic` package.
CTAN:macros/latex/contrib/overpic/

[12] `psfrag` package.
CTAN:macros/latex/contrib/psfrag/

[13] PSTricks.
http://tug.org/PSTricks/, CTAN:graphics/pstricks/

[14] PGF/TikZ.
http://sourceforge.net/projects/pgf/, CTAN:graphics/pgf/

[15] METAPOST.
http://sarovar.org/projects/metapost/, CTAN:graphics/metapost/

THE PracTEX Journal

Tools for creating bibliographic databases for use with BibTeX

Duvvuri Venugopal

Abstract

By using BibTeX we can easily change the style of Bibliography/References according to the style of the journal. But creating bibliographic databases for use with BibTeX is very cumbersome. This article describes the various software tools available for creating bibliographic databases easily, particularly for the Windows platform.

Mr. Venugopal, a secretarial assistant by profession, fell in love with TeX, LaTeX in the year 2000. A former admirer of WordPerfect, his first encounter with TeX, LaTeX was as a result of a challenge from a colleague. His hobbies are exploring various TeX packages, experimenting with Omega/Lambda, MetaPost and MetaFont, and teaching and/or explaining the advantages of TeX, LaTeX over word processors. From time to time he conducts LaTeX training for research scholars in Banaras Hindu University. He has contributed three articles to TPJ. You can reach Mr. Venugopal at dvgtex@gmail.com

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/14

Tools for creating bibliographic databases for use with BibTEX

D.V.L.K.D.P. Venugopal

Email dvgtex@gmail.com
Address Senior Personal Assistant
Vice-Chancellor's Office
Banaras Hindu University
Varanasi 221 005, India

Abstract By using BibTEX we can easily change the style of Bibliography/References according to the style of the journal. But creating bibliographic databases for use with BibTEX is very cumbersome. This article describes the various software tools available for creating bibliographic databases easily, particularly for the Windows platform.

1 Introduction

One of the major advantages of using LATEX is ease of inclusion of bibliographies in conjunction with BibTEX. Various bibliographic style files (bst) are available for different journals and publishing houses. Using these bst files the format and citation style can be changed easily[1]. This facility is not available in commercial wordprocessors or desktop publishing systems. Creating BibTEX databases is also discussed in LATEX Tutorials[2] and by Parthasarathy[3].

Bibliographic databases for BibTEX can be created using ordinary editors but it is tedious work involving a lot of typing. But once created, the same data can be used many times and by many users.

A search on the Internet for BibTEX database managers gives a long list of free and commercial software[1]. To name a few: bibtex mode and Ebib (for Emacs), Pybliographer (using Python for Linux), gBib (for GNOME, Linux), Barracuda (for Linux), KBibTeX (for KDE, Linux), Sixpack (multi platform), JBibtexManager, Javabib and JabRef (multi platform using Java), BibDB (for DOS and Windows),

1. http://www.google.com/Top/Computers/Software/Typesetting/TeX/BibTeX/

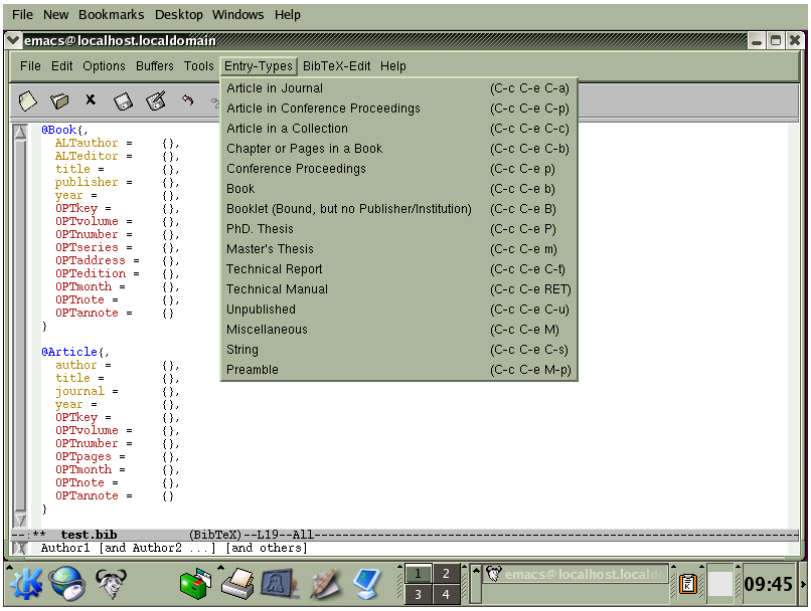


Figure 1: Editing BibTeX files in Emacs

BibTexMng (for Windows) and TkBibTeX (multi platform using Tcl/Tk) are some of them. In this article we discuss various software systems available for creating bibliographic databases.

2 Editor-based systems

2.1 Emacs

My first experience was with Emacs. Emacs is a multipurpose editor and, like TeX, is available for various platforms and is free[2]. When we save a file, it scans the extension and the menus and behaviour of Emacs changes. It has a BibTeX mode. (This is activated by 'bibtex.el'. By default it should be installed with Emacs. Otherwise one has to download and install it.) When this mode is present, the moment we save a file with the 'bib' extension the menu changes and a menu item 'Entry-Types' appears, with various types of bibliography categories. On selecting the category of a bibliography item a template will be added to the

2. http://www.gnu.org/software/emacs/

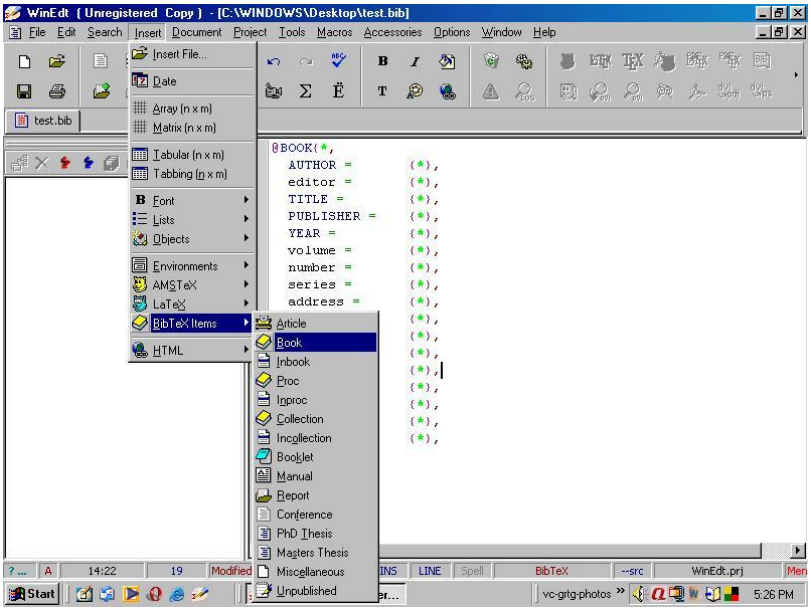


Figure 2: Creating bibliography data base in WinEdt

file. Now we just have to fill the various fields. The optional fields are indicated with 'OPT'. In addition the fields are also colour-coded for visual differentiation. Figure 1 shows a screen shot of Emacs, with empty fields ready to fill for a book, and the menu for bib.

We have not tried the 'Ebib' for Emacs. From the description given for this mode it appears it is better than the BibTeX mode of Emacs. Though the professionals in *nix world love Emacs due to its versatility, it has a steep learning curve.

2.2 WinEdt

WinEdt is a shareware editor distributed with MikTeX for MS Windows platform[3]. It also has a feature similar to that of Emacs. The only difference is it creates a star symbol where one has to replace it with actual data. It is advised to remove all these stars before saving the file or starting a new bibliographic entry. Otherwise these stars will create a problem when compiling the TeX documents. Figure 2

3. http://www.winedt.com

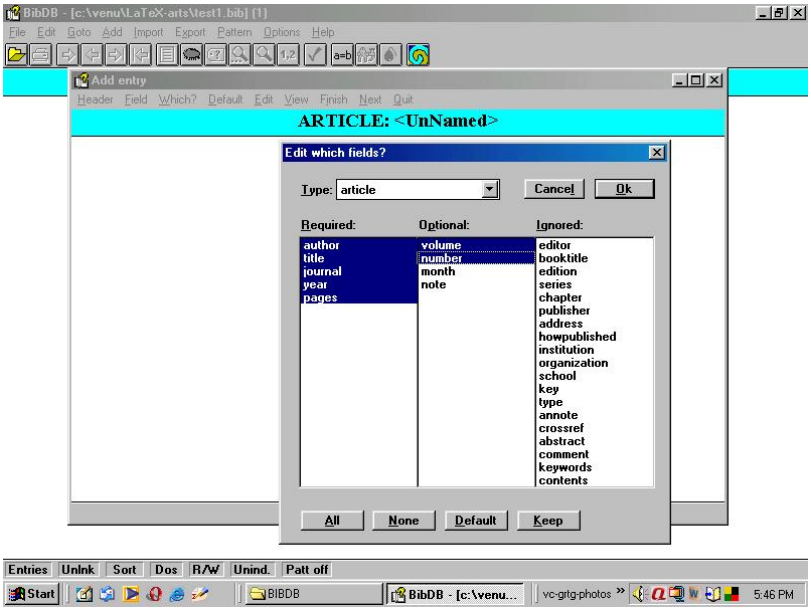


Figure 3: BibDB with fields selected

shows the screen shot of WinEdt ready for entering the bibliographic data. In WinEdt the compulsory fields are shown in capital letters whereas the optional fields are in lower-case letters. When starting a new entry care must be taken so that the cursor is outside the entry. Otherwise there may be problems when typesetting the document.

3 Dedicated bibliographic database software

In this category we have BibDB, TkBibTeX, BibTexMng, BibEdit, Pybliographer, gBib, Barracuda, KBibTeX , Sixpack, JBibtexManager, Javabib, and JabRef. Of these, BibTexMng is a shareware program. The programs available for Linux platform are not taken up, as they are more technically oriented. Java-based programs were not evaluated. The link for BibEdit happened to be a dead link. So only BibDB and TkBibTeX are discussed here.

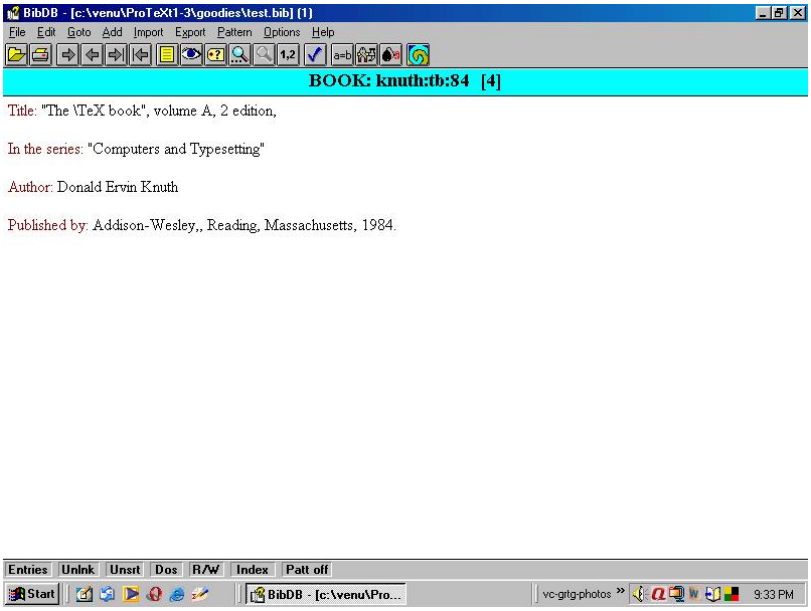


Figure 4: Bibliographic data in BibDB

3.1 BibDB

BibDB is a full fledged bibliographic database software for Windows. It is created by Eyal Doron[4] and is freely available from CTAN[4]. A search for BibDB found several software systems with the same name. The other BibDB is actually a BibTeX to DocBook translator.

BibDB was first created for MS DOS and later ported to MS Windows. The source code, in Turbo Pascal, is also available. MacKichan Software, makers of Scientific Word (a commercial version of LaTeX), also bundles BibDB along with their software.

After starting BibDB, the first step is to open a 'bib' file. Once this is done all the menu items are activated. To enter data, first select the 'add' menu, which will open another window 'Add entry'. Here chose the menu item 'Which?'. Then another window 'Edit which fields?' pops up (See Fig. 3). Select the type of entry, for example article, book, booklet, etc. The fields will change depending on the type of entry selected. For example, if `article` is selected, *author, title, journal,*

4. http://www.ctan.org/tex-archive/support/bibdb

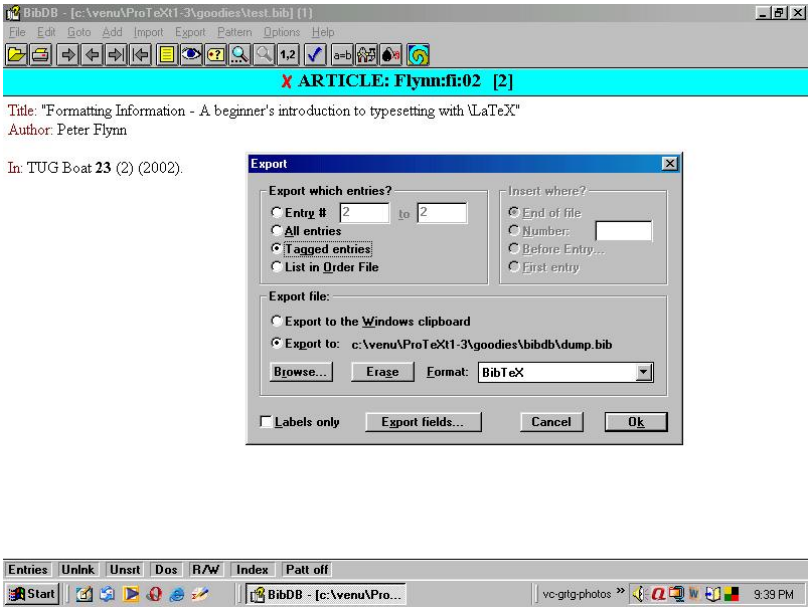


Figure 5: Exporting selected bibliographic data in BibDB

year, pages are the 'required' fields, *volume, number, month, note* are 'optional' fields and all other fields are under the category 'ignored'. Once the desired fields are selected, press 'OK' and then the actual input can begin. After filling a field, press 'Next' till all fields are filled. When all fields are filled the window goes back to 'Add entry' and shows all the fields as in Fig. 4. Now choose 'Finish', and it will create a key. Either accept it or change it. After this the entry will be added to the 'bib' database file.

The main advantage of BibDB is that it shows data in a clean reference card pattern (Fig. 4). One can search the database by key or any other expression. BibDB automatically sorts the database and arranges it in alphabetical order.

Sometimes the publisher may request the bibliographic data in a 'bib' file. With BibDB it is very easy. Just select the required references, then press 'Export'. A window will pop up as in Fig. 5. Select 'Tagged entries', select the format, and press 'OK'. The output will be saved as 'dump.bib'. Now the file is ready to send. Using BibDB, one can import or export the data to Tib, Refer, and Comma-delimited formats.

The other advantage with BibDB is customisation. It can be used as a Personal

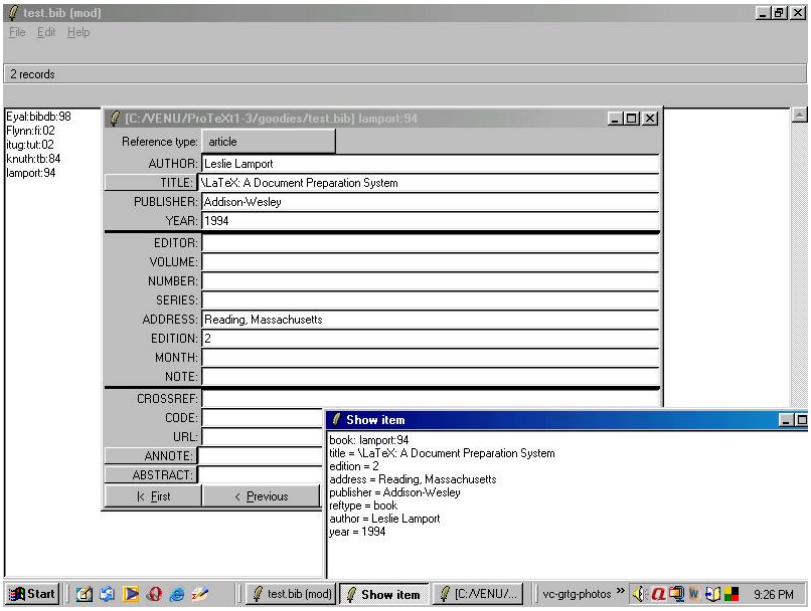


Figure 6: The interface of TkBibTeX

Information Manager (PIM) for keeping addresses and contact information. In conjunction with the package 'Directory', beautiful address books, address lists, telephone lists, e-mail lists can be created. These features will be discussed in a separate paper.

3.2 TkBibTeX

TkBibTeX was first created by Peter Corke and can be downloaded from CTAN[5]. The present version is modified by G. Milde. A reference to it was made by Flynn[5]. It is based on the Tcl/Tk (Tool Command Language and Tool Kit) created by John Ousterhout. Tcl/Tk is an interpreted language and like TEX it is also platform independent and is available for Win32, UNIX, Linux, and Mac platforms. Tcl/Tk can be freely downloaded from the internet[6]. The same script can be used on all platforms and the look and feel will be the same. Though a comprehensive help is provided with the GUI of TkBibTeX, the author's home

5. http://www.ctan.org/tex-archive/biblio/bibtex/utils/tkbibtex
6. http://www.tcl.tk/software/tcltk/

page gives the various ways of using it for better results[7].

In TkBibTeX two screens are provided. The first screen shows the various reference items by key. On selecting a reference item, or on selecting 'New Entry' in the 'Edit' menu, the other screen opens. This is further divided into three parts. The first part contains the compulsory fields, the second part contains the optional fields, and the third part contains the ignored fields (Fig. 6). The difference with BibDB is, in BibDB the key is created in the last whereas in TkBibTeX it should be provided in the beginning itself.

4 Conclusion

In this paper only Emacs, WinEdt, BibDB, and TkBibTeX are discussed. The screen shots of other software systems shown on their respective web sites are enticing. However, the author was unable to install these programmes for a proper evaluation.

Though there are several methods of creating bib files and databases, the tools reviewed here are the ones the author found workable and best suited to his needs.

Acknowledgments

The author thanks the anonymous reviewers and Lance Carnes, the editor of this article, for their valuable suggestions in improving the quality of this paper.

About the author

Mr. Venugopal, a secretarial assistant by profession, fell in love with TEX/LATEX in the year 2000. A former admirer of WordPerfect, his first encounter with TEX/LATEX was as a result of a challenge from a colleague. His hobbies are exploring various TEX packages, experimenting with Omega/Lambda, METAPOST and METAFONT, and teaching and/or explaining the advantages of TEX/LATEX over word processors. From time to time he conducts LATEX training for research

7. http://www.cat.csiro.au/ict/staff/pic/tkbibtex.html

scholars in Banaras Hindu University. He has contributed three articles to The PracTEX Journal.

References

[1] Oren Patashnik. BibTEXing. Documentation for general BibTEX users. February 8, 1988.

[2] Indian TEX Users Group. LATEX Tutorials – A Primer. E. Krishnan (Ed.). 2002. http://www.tug.org.in

[3] S. Parthasarathy. Demystifying LATEX bibliographies. The PracTEX Journal, 2, 2007.

[4] Eyal Doron. BibDB for Windows – A BibTEX Database. 1998. http://www.tcisoft.com/tcisoft/bibdb.html

[5] Peter Flynn. Formatting Information. TUGboat, 23(2), 2002.

THE PracTEX Journal

Bib Manager and Word Citer: Bibliography Management and Citation Extraction

Fernando Sáenz-Pérez

Abstract

This article describes two tools for managing and using BibTeX bibliographies. The first tool, Bib Manager, allows the user to manage references and store them both as relational databases as well as the usual .bib text plain files. The second tool, Word Citer, allows the user to cite and list references, stored in a database or in a .bib file, in a MS Word document with the selected bibliography style. It relies on another open-source project, LaTeX2RTF, which we use to provide format to the list of references as well as to generate the citation text. Bib Manager is cross-platform, free, and open-source. Word Citer has the same features but, obviously, it is not cross-platform. Both of them will be distributed under GPL.

Fernando Sáenz-Pérez writes: I am an Associate Professor at the Department of Software Engineering and Artificial Intelligence at the Faculty of Computer Science of the Complutense University in Madrid (Spain). LaTeX was almost my first choice (after Chi-Writer) for writing technical documents on a workstation, when PCs were first introduced. Since then, I have used this document preparation system for writing papers, articles, books, slides, and my PhD thesis. My research interests include declarative programming (logic, functional, constraint programming), (deductive) databases, and natural language processing (ontologies). Other interests include videogames — I am involved in teaching in the class "Developing Videogames", and have developed an 80's videogame. I have been in close contact with companies in developing software for many years, and find this provides useful feedback from the real world as we continue to do university research. You may contact me at http://www.fdi.ucm.es/profesor/fernan.

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/28

Bib Manager and Word Citer: Bibliography Management and Citation Extraction

Fernando Sáenz-Pérez

Website http://www.fdi.ucm.es/profesor/fernan
Address Facultad de Informática, Universidad Complutense de Madrid, Spain

Abstract This article describes two tools for managing and using BibTEX bibliographies. The first tool, Bib Manager, allows the user to manage references and store them both as relational databases as well as the usual `.bib` text plain files. The second tool, Word Citer, allows the user to cite and list references, stored in a database or in a `.bib` file, in a MS Word document with the selected bibliography style. It relies on another open-source project, LaTeX2RTF [7], which we use to provide format to the list of references as well as generating the citation text. Bib Manager is cross-platform, free, and open-source. Word Citer has the same features but, obviously, it is not cross-platform. Both of them will be distributed under GPL.

Keywords: Tools, Free, Open-source, Cross-platform, BibTEX, Bibliography Management, Databases

1 Introduction

Our role in the university as researchers (hence, paper-writers) and lecturers (thus, manual and study-guide makers) requires us to deal with a few to numerous references in our documents. It is not uncommon to have BibTEX files of around 1MB. Several of us often use both LaTEX and MS Word documents depending on the document we are preparing. Sometimes, we use MS Word for rapid preparation, but when we need either professional-looking documents, or we are preparing large documents, or including lots of references, we resort to LaTEX and BibTEX. BibTEX provides versatile bibliography handling, and makes the listing of references an easy task. However, since references are stored as plain tex

files, when the number of references grows the task of managing them becomes more difficult. Therefore, we were looking for, first, a bibliography manager, and, second, a reference citer for MS Word which could use those bibliographies. I directed a team of students in developing such tools under my guidance. As a result, a project was started as part of their university studies to develop a bibliography manager as well as a citer for Word documents. The requirements of this project were several-fold:

Database-oriented A bibliographic (data) file is usually present but bibliographic databases are often missing. Databases, however, add several features which cannot be found in file-oriented applications. The most important feature for us is the concurrent access to databases. This allows concurrent updating by different users, who can seamlessly add references to a common bibliographic database. In fact, this is one of our current problems in writing collaborative documents, where each writer in the team has to add references. Therefore, using a database greatly simplifies our work. We focused on relational databases because they are quite commonly used nowadays, and their capabilities fulfill our requirements.

Simple We looked for, first, a simple, intuitive GUI. Second, we wanted to have the option of working with a database or simply with BibTeX plain files. Third, the tool should be easily installed, possibly requiring no installer. In fact, merely decompressing an archive and executing a file is enough to start working with Bib Manager. Unfortunately, in the current version of Word Citer we were not able to provide a simple installer; it requires a manual and cumbersome installation procedure. Obviously, this will be fixed as soon as possible.

Internationalized In order to have a wider audience, text files are used to define all the texts of the GUI, so that the system can be easily localized to different languages.

Open-source We do believe that the open-source alternative will produce better software and systems, and allow developers to contribute enhancements. This is fitting in a university context, so that students will have sources available to learn and practice with them. In an unusual step, the project's main application has been developed without using other software as a basis. While doing this would

have allowed more rapid development, starting from scratch gave the students a unique opportunity, one of the few they will probably have in their careers. They were able to rely on LaTeX distributions for lists of references and cross-translations between BibTeX and database entries. In addition, they used the free, open-source project LaTeX2RTF [7].

A Cross-Platform System Despite the basic need to develop the same system for several platforms, many of us use several different operating systems and therefore want to have the same applications available on several platforms. As a consequence, the selected development language was Java, which is widely known and works on any platform. Of course, this was not the only alternative, since one can use C#, for example. But for developing a cross-platform system that can be widely used and modified in an open-source project, Java was chosen.

Free One of our goals in the university is to share knowledge, and we make our work freely available to others. Further, providing free software allows for widespread usage. In addition, our university and several national projects support us as researchers and lecturers (see Section 5), so we do not need additional income from developing the software.

Before starting this project, other related systems were analyzed. Note that there are many bibliography managers and we only list here some representative ones. On the one hand, there exist some commercial or non-free database managers such as PAPYRUS (Windows and Macintosh, which now becomes free but not open-source) [9], ProCite (Windows, personal and networked) [10], Reference Manager (Windows and Macintosh) [11], EndNote (Windows and soon for Macintosh) [5], and RefWorks (a Web-based application to create personal bibliography from text files or online databases) [12]. We reviewed their features but did not consider them an alternative to free software.

On the other hand, among free software systems, we found SIXPACK [13], JabRef [6], and BibShare [3]. SIXPACK is open-source, cross-platform and manages file-oriented bibliographies. In addition, it does not make automatic citations or lists of references, and it seems to be a closed project (no Web page modifications since 2000). JabRef is open-source and cross-platform. It is also file-oriented and does not include a citer. BibShare (an evolution of BibWord [4]) is free but not

open-source (although it is open in the sense that it can be connected to different word processors by programming the interface), and only works for Windows.

In summary, after stating our original goals and reviewing related systems, we concluded that no system fully fits our requirements. LaTeX and/or MS Word users might find the tools we propose useful. Note, however, that our system is emerging and it is in the early stages of development. However its future development will be guided by the aforementioned objectives as well as user feedback.

This paper is concisely organized as follows. Sections 2 and 3 describe Bib Manager and Word Citer, respectively, from a user point-of-view. Some conclusions are summarized and some future work is pointed out in Section 4. Finally, acknowledgements are posed in Section 5.

2 Bib Manager

2.1 Features and Limitations

This tool is basically a bibliographic manager based on the BibTeX data format. Data are stored in databases, instead of files, so that concurrent access from different (remote) instances of Bib Manager is possible, therefore maintaining a consistent storing for shared bibliographies. MySQL and MS Access database management systems (DBMS) are supported, and others will also be supported soon (PostgreSQL, Oracle, Sybase, DB2, etc.). Note that although Access is a personal, file-oriented, passive (without triggers) database, it can be remotely accessed with a networked file or as a Web service. In addition, locking and transactions are supported through their APIs, so that concurrent accesses can be ensured to be isolated. Several databases can be opened in Bib Manager at a time. It can import BibTeX data from `.bib` files and export a database to a `.bib` file. Entries in a database can be created, modified, deleted and copied to any other open database. A key for a new reference can be automatically created following the alpha bibliographic style. BibTeX display is always possible either for single references or for the complete database. (Relational) table views can be filtered by a search filter for usual fields and for any fields. The table view can be ordered by any column, columns can be hidden or shown, and they can be moved and sized. For the modification of a single entry, a dialog is provided; in the table

view, direct modifications will be possible. A log is provided for development purposes.

As far as limitations, Bib Manager can only open ten databases a time, but this will be upgraded to open as many as needed. Only two DBMSs are supported, but other widely-used ones will be supported. The tool cannot import from other bibliographic sources. The main dialog cannot be resized. A Web-based application is not provided in the current version. The current status of the development is alpha, so that several bugs need to be fixed, as well as some non-intuitive behavior. In addition to this short list, a given user might find several other limitations. We have only highlighted the most noticeable ones (from our point of view).

2.2 Technology

Bib Manager has been completely implemented using Java under Eclipse. MikTex 2.5 [8] has been used for generating reference keys along with OS batch files.

2.3 Description of Bib Manager

Figure 1 shows the main GUI of Bib Manager, which consists of a row/column-based table view for several databases, organized in a tab form (left-hand panel). The right-hand panel allows filtering of entries that contain the typed text in any of the fields Author, Title, Year, and Key. The same is also possible for the rest of the fields in the text box Other Fields. Note that the filtered entries contain the typed text somewhere in the required field, which is useful, for instance, if we are looking for the publications of an author who may also be a co-writer. Also, in this right-hand panel, the ordering of entries can be selected for any field, which is also useful for lookups. The number of displayed references (which depends on the applied filter) is counted in the text box Ref's. The bottom panel includes icons as shortcuts for frequent operations; from left to right: the magnifying glass for switching BibTeX and table views (see Figures 1 and 2), the trash can for deleting selected entries, the opened book for inserting a new reference, the selection of a book for modifying a single entry, the tools for configuring preferences, and the double arrow for copying selected entries between two databases. Since data read from the database are kept in main memory, updates from other users are not seen unless a view refresh is done (the button Refresh View is missing but

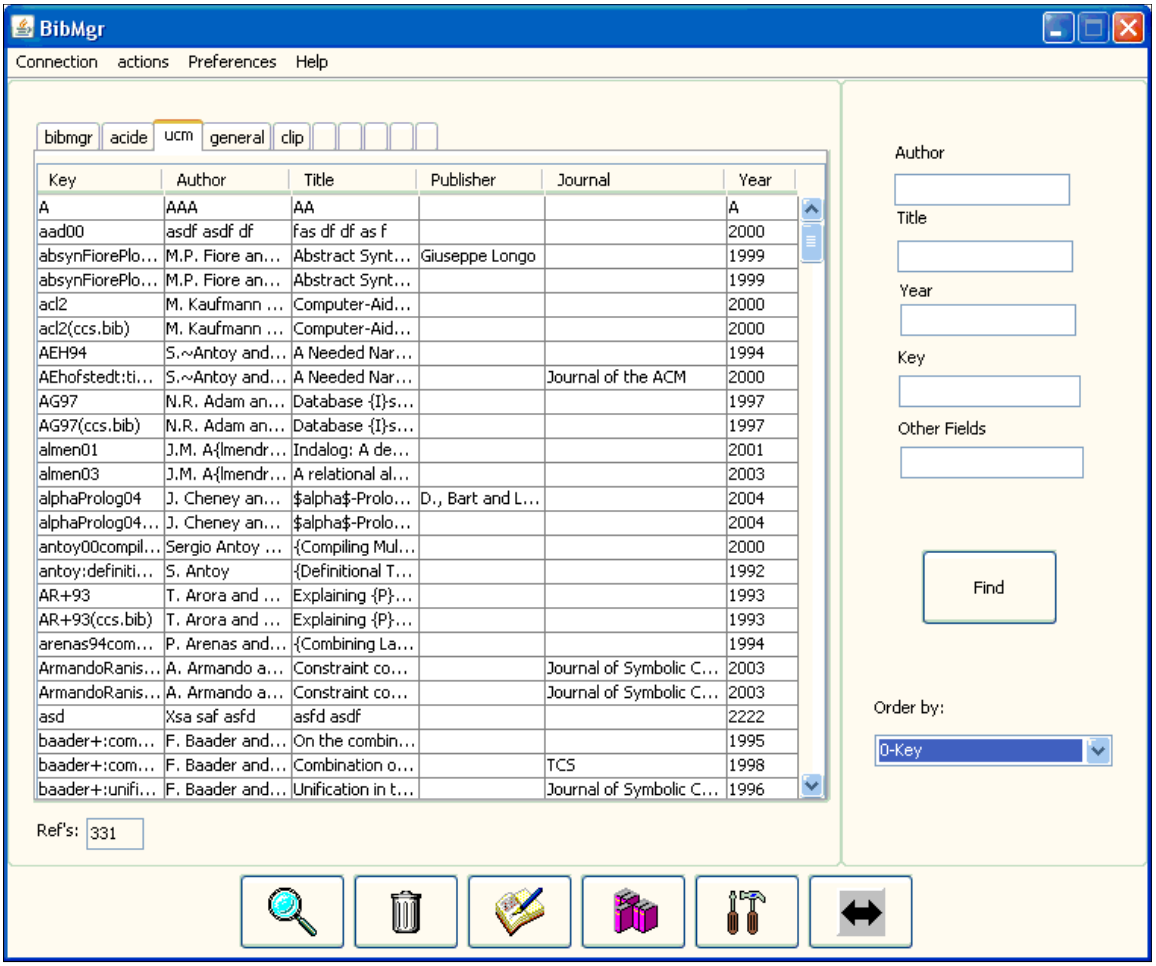


Figure 1: Bib Manager Main GUI

under development). In any case, updates from other users are supervised by the DBMS, therefore avoiding duplicates because the primary key of the database table is the BibTeX key.

The menu bar has several entries:

– Connection. For connecting to and disconnecting from both MySQL and Access databases. Connection names as used in Word Citer (see Section 3.3 should be added.

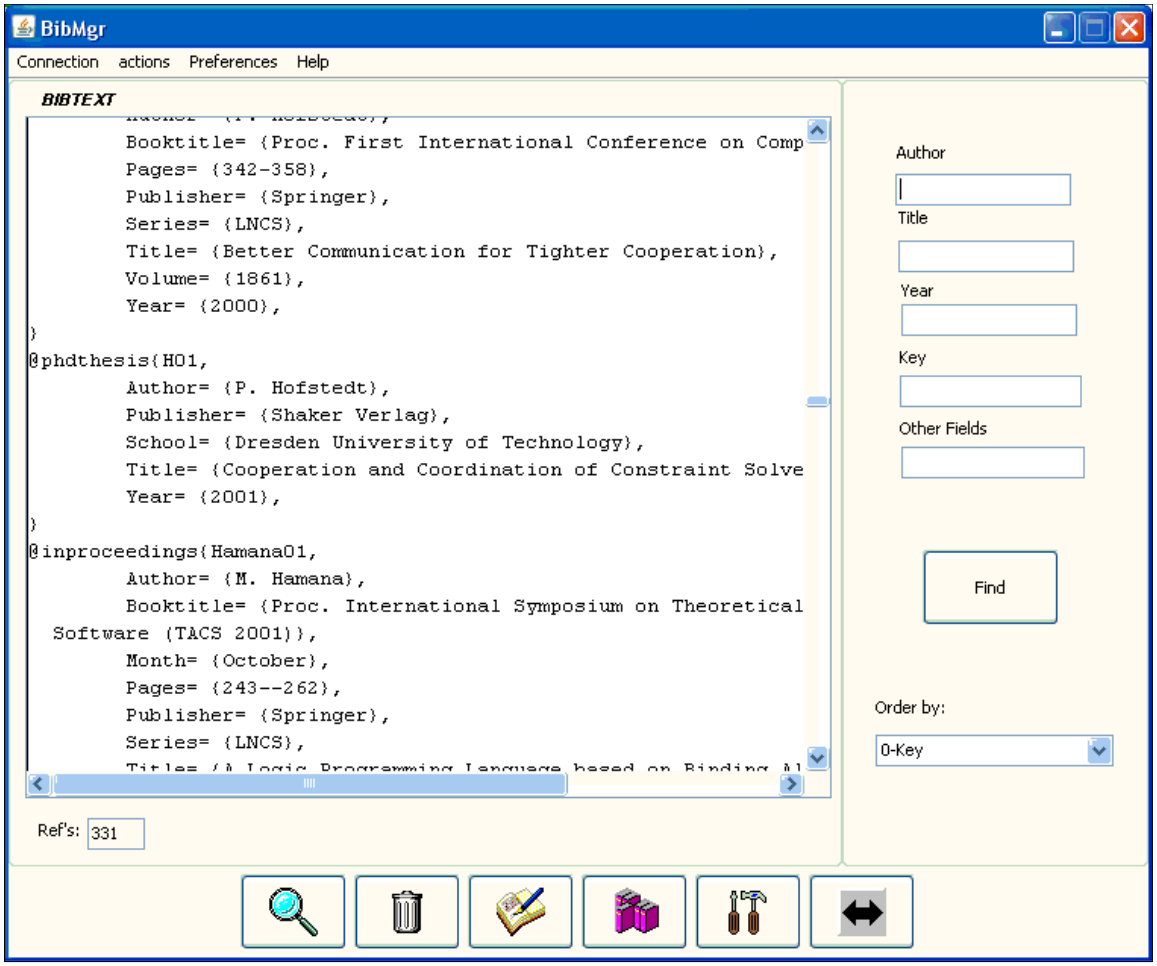


Figure 2: BibTeX View in Main GUI

– Actions:

  · Insert Reference. Opens a dialog box for inserting a new reference (see Figure 3). Depending on the entry type (book, article, inProceedings, etc.) that had been selected in the drop-down box, the dialog will show the corresponding mandatory and optional fields to be filled. It is possible to automatically generate a key from the citation label that LaTeX/BibTeX generate in terms of the filled fields (currently, only the style alpha is supported). For this, both a `.tex` document containing

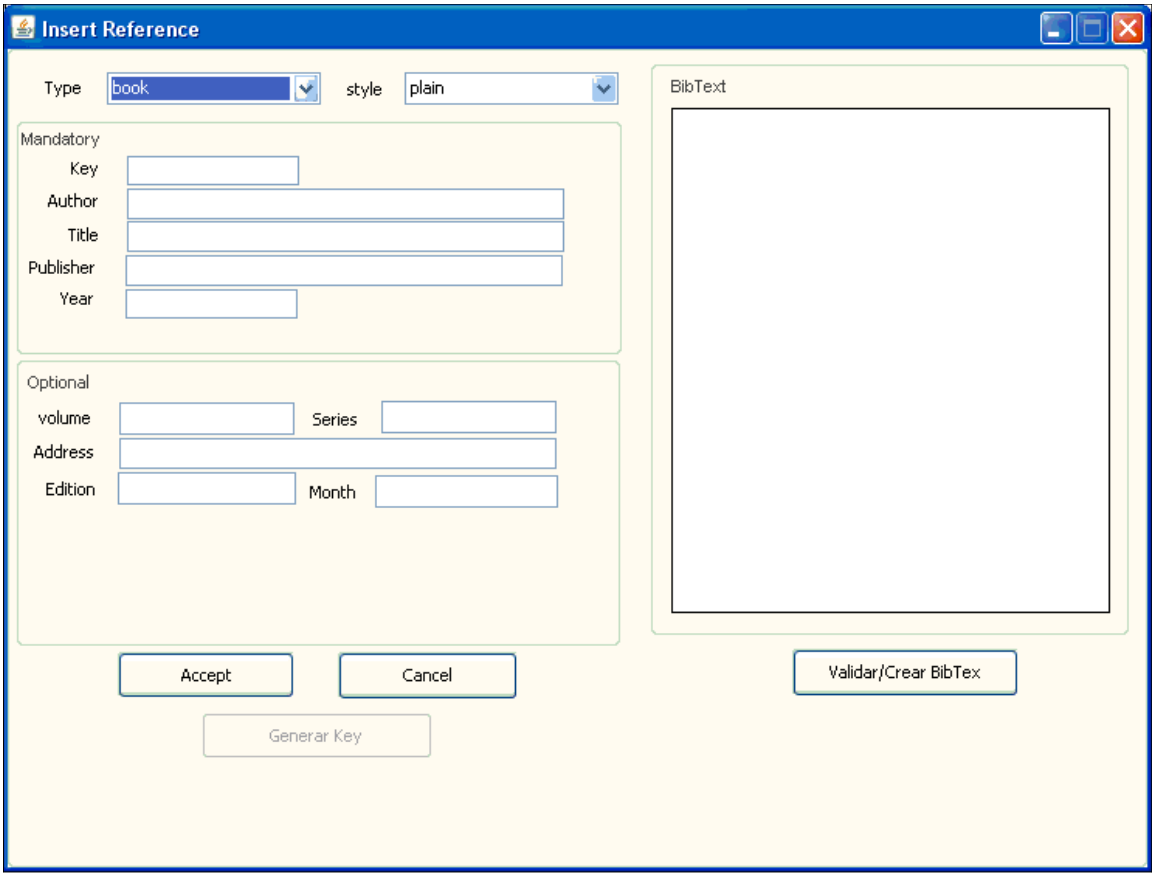


Figure 3: Inserting a Reference

a `\cite{gen}` and a `.bib` containing a BibTeX entry are generated and compiled. The result of the compilation contains the automatically generated citation label, which is extracted from the `.bbl` file.
The button Create/Validate Entry allows two operations. First, to display in the text box Bib Text the BibTeX text format corresponding to the filled fields of the reference (left-hand panel). Second, to automatically fill the fields of the reference from a manually typed entry (or pasted from another source) in the text box Bib Text (this last point is under development).

- Delete Reference. Deletes the selected references. The key Del is a short-

cut.

- Copy Reference. Copies selected references to another opened database, which can be from any supported vendor. Ctrl+C and Ctrl+V should be shortcuts, which is a future enhancement.
- Modify Reference. Opens a dialog box for modifying an existing reference (see Figure 4). The dialog is the same as the one for inserting

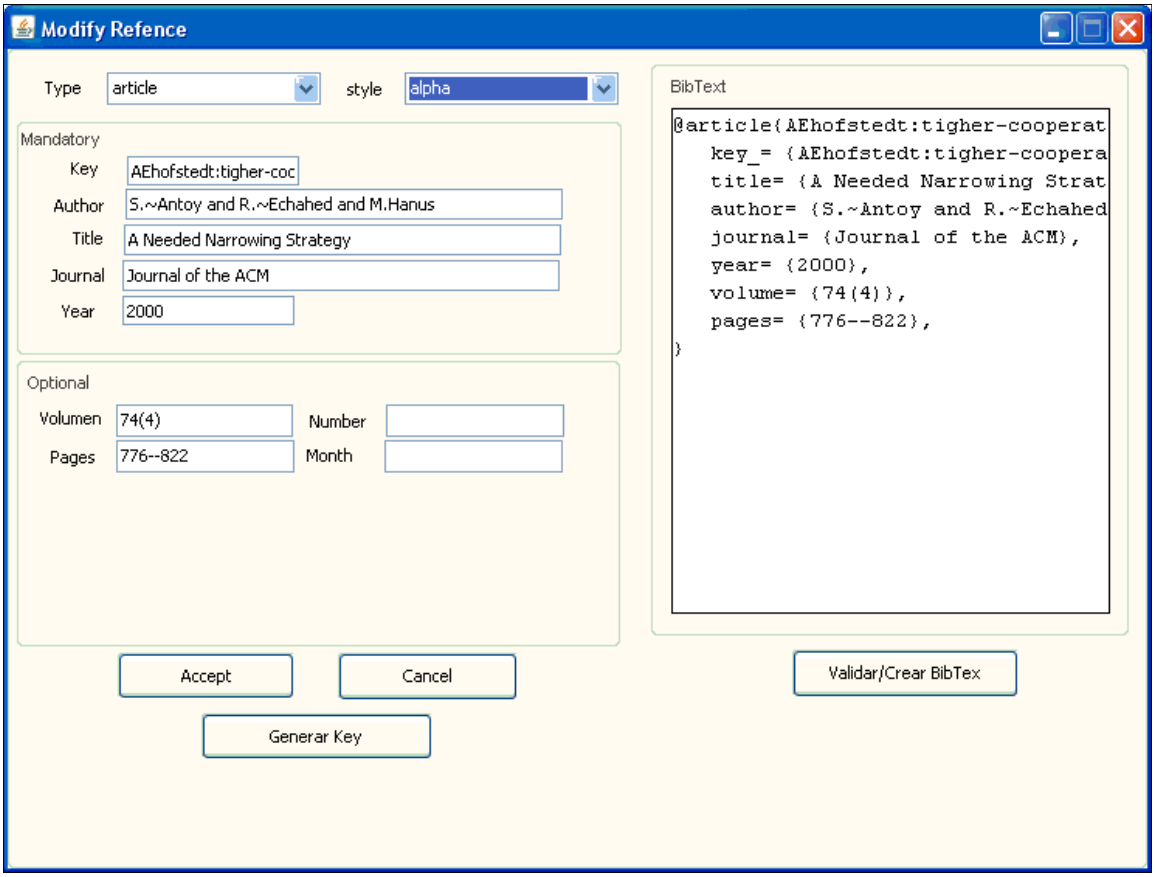


Figure 4: Modifying a Reference

a new reference, but here we can see other fields because a different reference type is selected. Also, double-clicking on the table view edits the contents of a cell.

- Create Database. Creates a MySQL or Access database.

    - Delete Database. Deletes (drops) a MySQL or Access database. The key Del should be a shortcut when the focus is in a tab.
    - Import BibTeX. Imports BibTeX entries from a `.bib` file.
    - Export to BibTeX. Exports the selected database to a new `.bib` file.
  - Preferences. Sets the user preferences for hidden/displayed columns, the GUI language, and the paths of the executables for LaTeX and BibTeX.
  - Help. Contains BibMgr Help, Log to show the log, and About Bib Manager.

This completes a short introduction to Bib Manager. The next section will describe a citer for MS Word documents, Word Citer, which can use the databases managed by Bib Manager.

3 Word Citer

3.1 Features and Limitations

Word Citer is a tool that resembles the synergy between LaTeX and BibTeX, with added capabilities. It allows inserting of references from different sources: Bib Manager databases and BibTeX files. There is provision for useful searching of references, similar to the search described in Section 2.3. When a new reference is first inserted in the Word document, its key is used as the unique identifier. Later, when listing the complete set of references, citation labels are resolved and the citations are updated with these. Citation labels appear following the desired bibliography style, whether predefined or custom. In this last case, any `.bst` valid file can be selected. In addition, the format of the references in the listing follows this bibliographic style.

There are several limitations in this tool. First, only MySQL databases can be handled, which has to be enhanced to support also MS Access, and several others in common use (PostgreSQL, Oracle, Sybase, DB2, etc.). Second, it cannot handle other bibliographic sources such as the ones that can handle EndNote [5], ProCite [10] or BibShare [3]. Similar to Bib Manager, there are several other limitations which we are tracking and hope to improve.

3.2 Technology

Word Citer has been completely implemented using Visual Basic for MS Word 2003. MikTex 2.5 [8] has been used for generating the list of references and citation labels. OS batch files have been also used.

3.3 Description of Word Citer

Figure 5 shows the new entry in the Word menu bar after the installation of Word

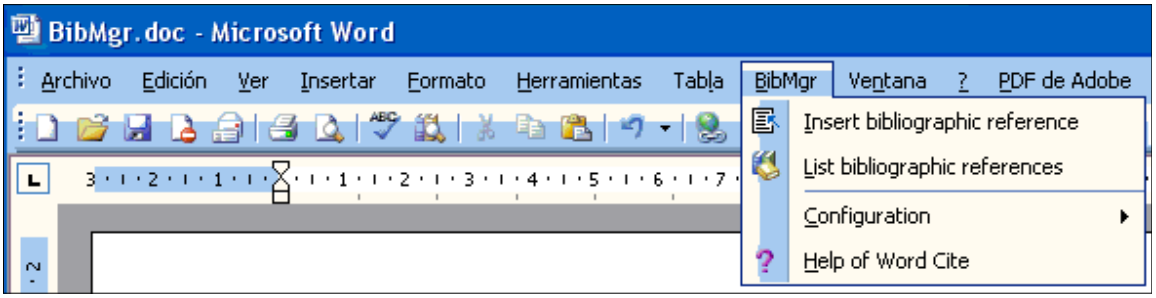


Figure 5: Menu Bar

Citer. The third menu item of BibMgr allows configuration of Word Citer (see Figure 6). The very first task a user will do with Word Citer is to configure the data

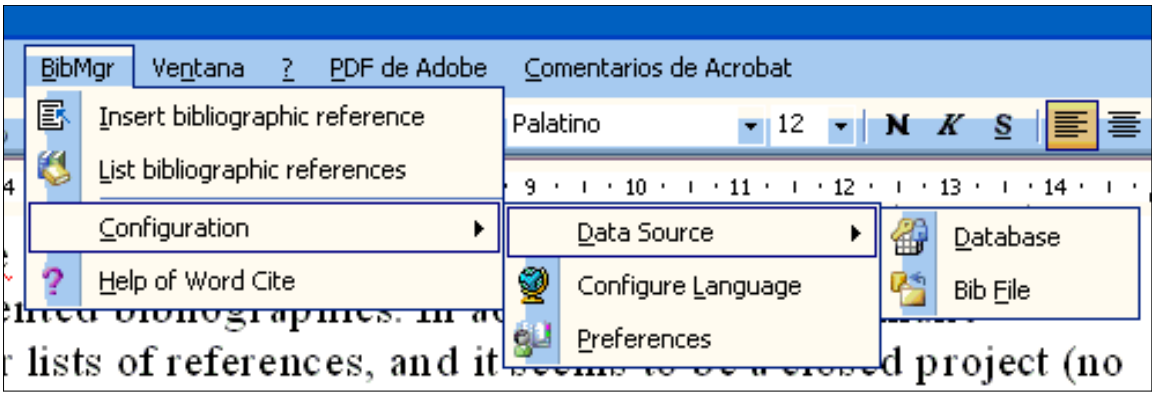


Figure 6: Menu Item for Configuring Word Citer

source, which can be a MySQL database or a `.bib` file, as shown in the figure. If the user selects the latter, an open-file dialog box appears for selecting the requested file. This operation will create an in-memory dummy database, hidden to the user, and handled as any other database regarding reference lookups and insertions. Selecting such a `.bib` file allows the user not only to use exclusively the references in this file, but also to add the new references to the existing ones, making it possible to handle several `.bib` files in the same Word document (as it is usual in LaTeX documents). If the user otherwise selects a database as a data source, the dialog box in Figure 7 appears. With this dialog, the user can manage

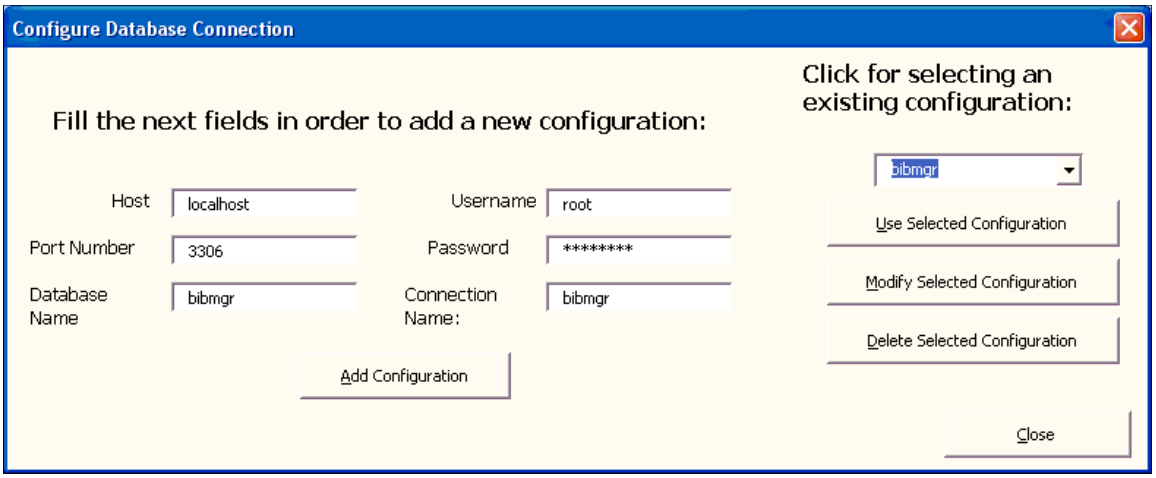


Figure 7: Selecting a MySQL Database Source

connections, which are identified with a connection name. The user can select any of the previous connections or create a new one. Whenever Word is started, the last connection is used, therefore saving time when continuing previous work. Connections can be managed with usual operations: create (Add Configuration), delete (Delete Selected Configuration), and modify (Modify Selected Configuration). To connect to a saved connection, the user pushes the button Use Selected Configuration. The list of previous connections is stored for future use in the drop-down box next to the upper-right corner. A straightforward enhancement is to allow the use of several database sources to look for references at the same time.

The second task the user will usually do is to insert a new bibliographic reference using the menu item Insert Bibliographic Reference, which opens the dialog

shown in Figure 8. In this dialog, the complete set of references from the chosen

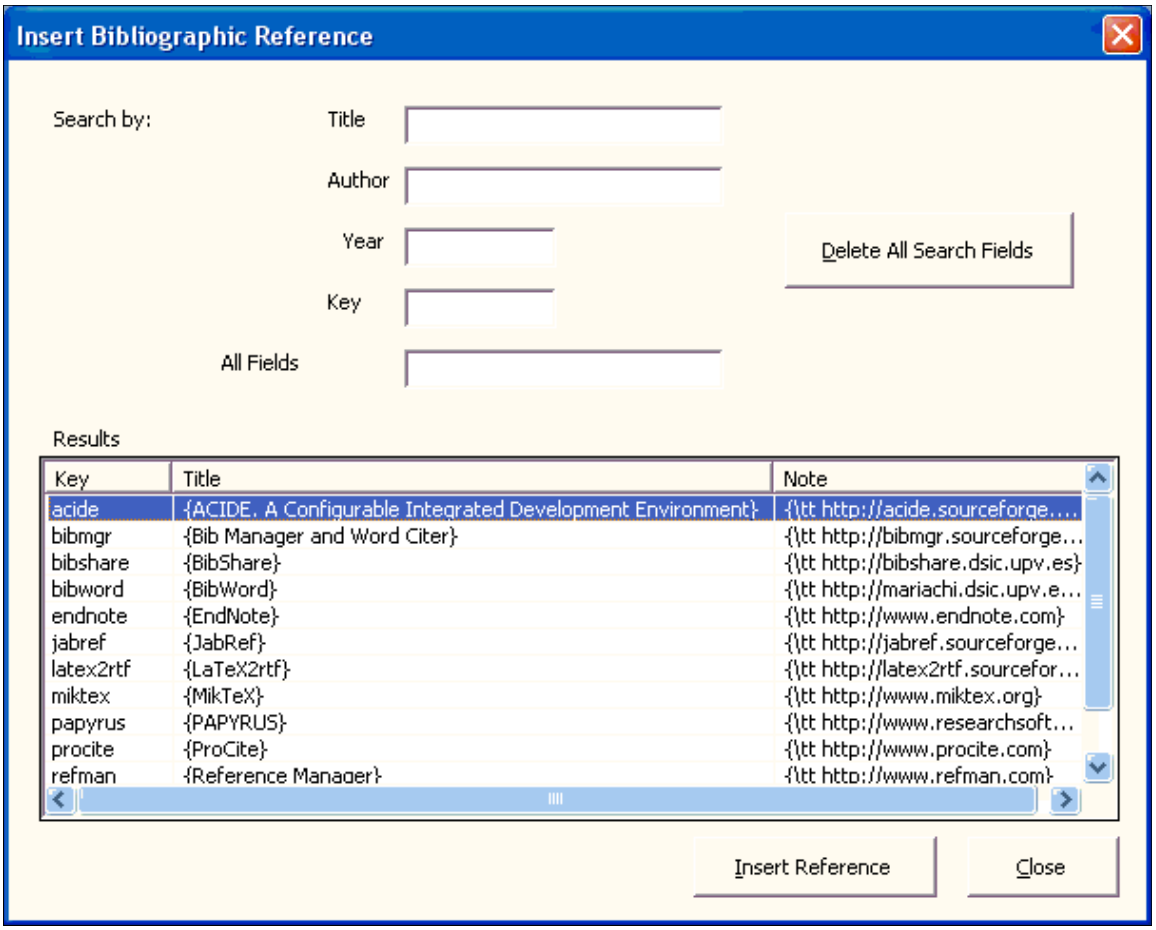


Figure 8: Inserting a New Reference

data source is shown in a table view. Filtering can be done similar to the method explained in Section 2.3 for facilitating the selection of the required reference. For instance, if we type `bib` in the Title search field, we get the result shown in Figure 9. Recall that the search text is looked for in any position of the database field.

Inserting a concrete reference amounts to inserting a new Word field at the current cursor position, which displays the BibTeX reference. When the list of references is created afterwards, this Word field will display the actual citation label generated by LaTeX and BibTeX. Figure 10 shows the result of inserting a

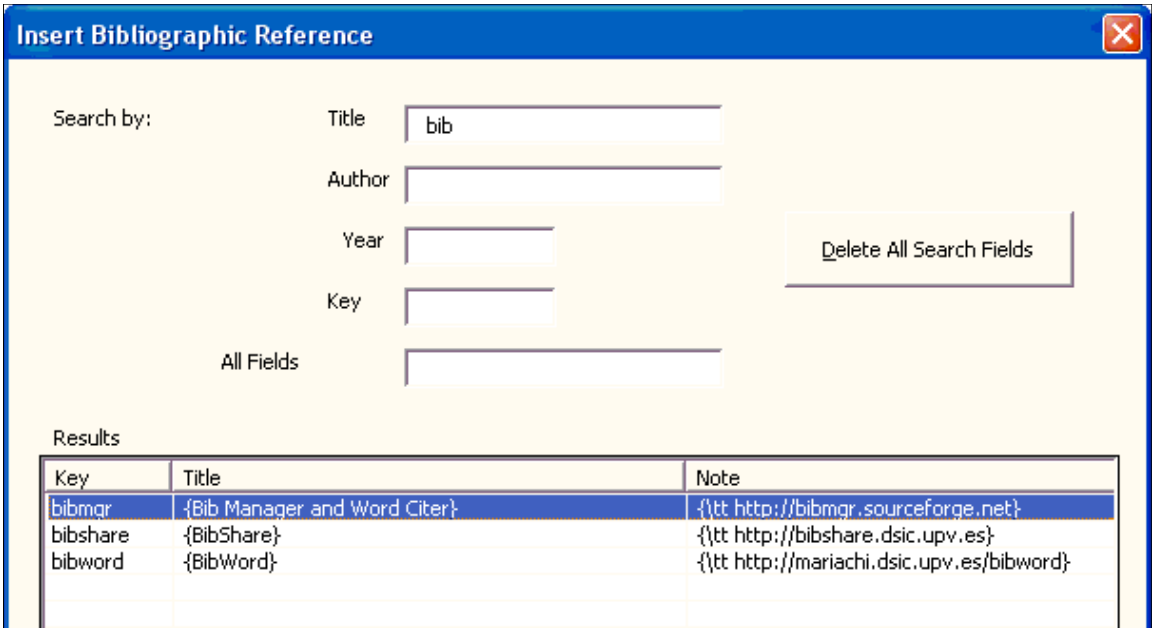


Figure 9: Filtering References

reference.

The last task is to list the references in the document, which is done with the menu item List Bibliographic References. Figure 11 shows the result, which is based on the `abbrv` bibliography style. Currently, this operation can be done only once, a known bug which will be soon fixed. LaTeX2RTF [7] is used to generate an RTF document which is the source for the listing.

Several configurations are available for Word Citer. As seen in Figure 6, in addition to configuring the data source, the GUI language can be selected from a drop-down list (resource files are used to define all the texts in the dialogs) as well as some preferences (Preferences, see Figure 12). These refer to the columns that the user wants to display when inserting references in the table view, the paths for LaTeX and BibTeX executables, and the bibliography style (a built-in or any valid `.bst` file).

This concise introduction to Word Citer closes this section. In the next section some conclusions and future tasks are outlined.

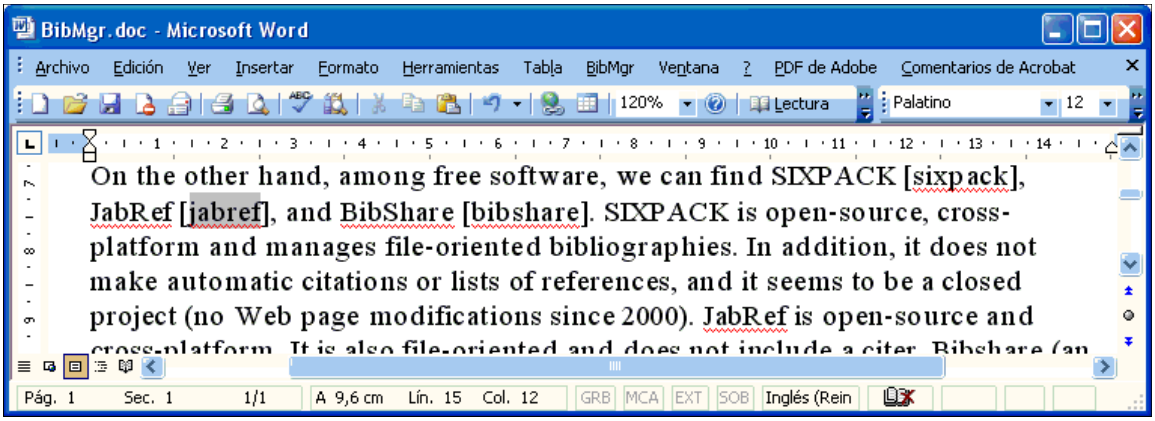


Figure 10: The Result of Inserting a Reference

4 Conclusions and Future Work

This paper has presented a bibliographic manager and citer, which are alternatives to other available tools, with features meeting our initial requirements. They cannot be thought of as complete tools since they are emerging and in an alpha development status. They should be seen as tools that, in time, might provide more and more features following our requirements. There are other tools in the free market today, but Bib Manager and Word Citer might be competitive in future development stages.

There are many goals to be achieved, some of which have been posed in the sections describing the tools, and others which have been indicated or have become evident after each tool description. Apart from fixing numerous bugs (including operating system dependent calls, which makes Bib Manager work only for Windows, up to now), we have to implement other interesting features such as the handling of other databases and even on-line access to bibliographies on the net, according to standards. Despite the assumed limitations, we hope that both Bib Manager and Word Citer tools will consolidate into friendly and powerful applications, amenable to its targeted users; in particular, to LaTeX/TeX and/or MS Word users. This project may be downloaded from `http://bibmgr.sourceforge.net`. We hope to upload the version described in this paper as soon as possible, and have a better and stable implementation.

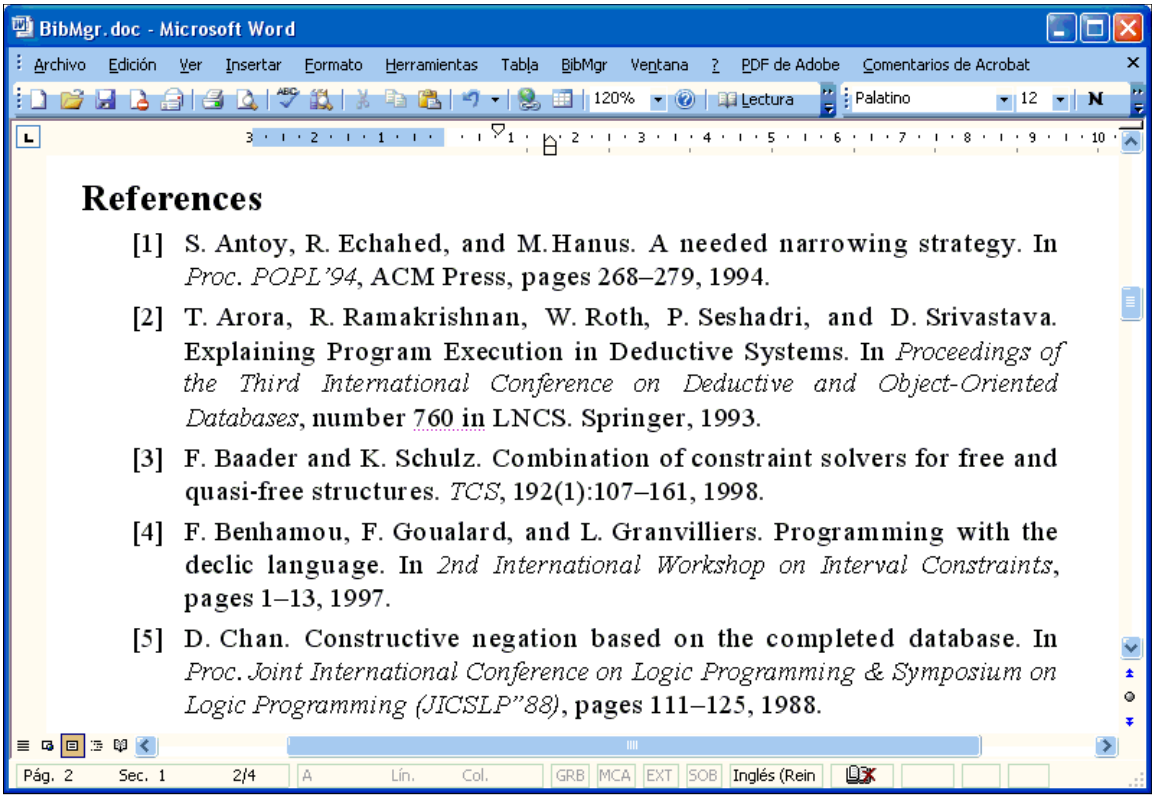


Figure 11: Reference Listing

5 Acknowledgements

I am grateful to the students who developed this system under my guidance, which I hope will be useful for them in the development of their own future projects. These students are Antonio Contreras Varas, Enrique Palomar Santamaría, and Javier Prats Robledillo, who developed this project in the computer science course "Sistemas Informáticos" (Computing Systems) for graduates, a course intended for the fifth year of Computer Engineering studies of the Computer Science Faculty of the Universidad Complutense de Madrid, Spain. Also, thanks to the projects TIN2005-09207-C03-03, and S-0505/TIC0407 which supported this work. Finally, thanks to the free, open-source project LaTeX2RTF [7] which we use in our project, as well as to the Java and Eclipse initiatives. This

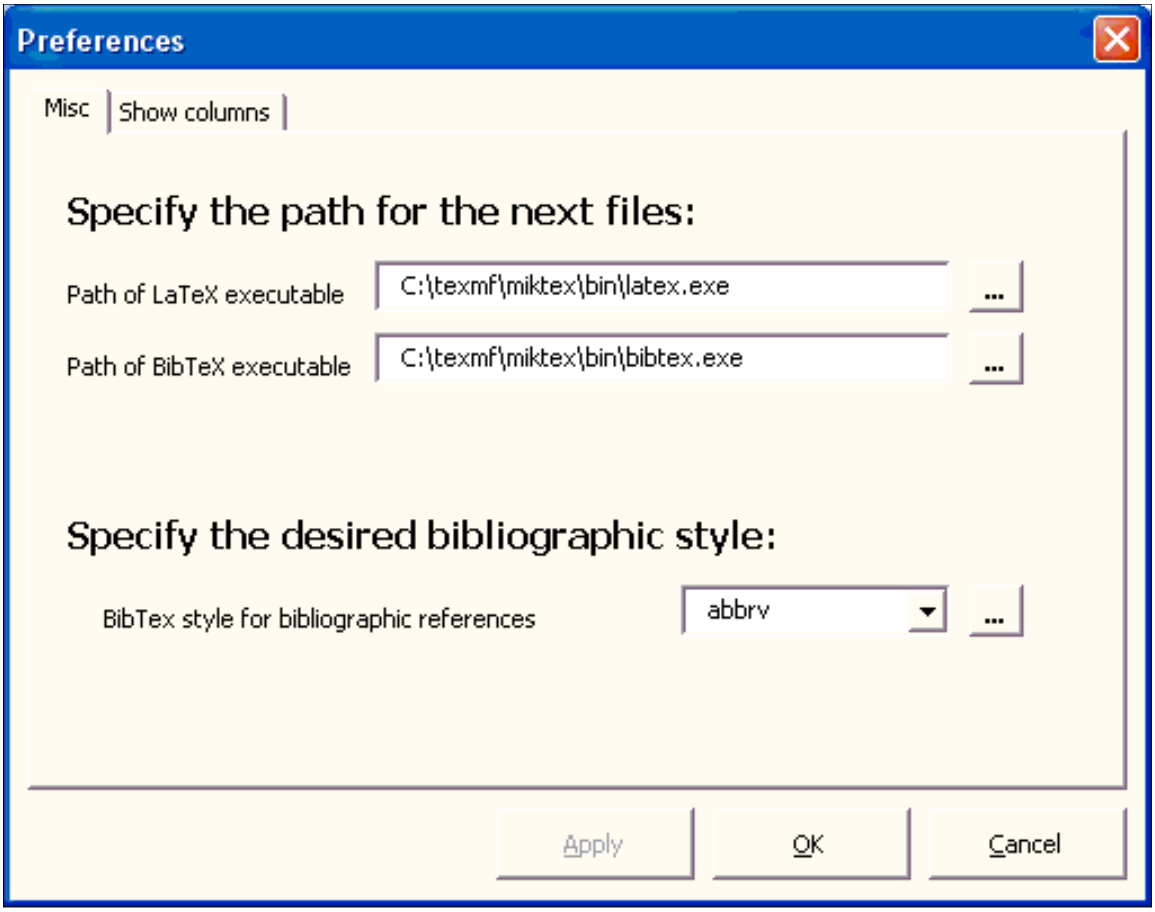


Figure 12: Word Citer Preferences

article was prepared with MikTeX 2.4 [8], ACIDE [1], and BibMgr [2].

References

[1] ACIDE. A Configurable Integrated Development Environment. `http://acide.sourceforge.net`.

[2] Bib Manager and Word Citer. `http://bibmgr.sourceforge.net`.

[3] BibShare. `http://bibshare.dsic.upv.es`.

[4] BibWord. http://mariachi.dsic.upv.es/bibword.

[5] EndNote. http://www.endnote.com.

[6] JabRef. http://jabref.sourceforge.net.

[7] LaTeX2RTF. http://latex2rtf.sourceforge.net.

[8] MikTeX. http://www.miktex.org.

[9] PAPYRUS. http://www.researchsoftwaredesign.com.

[10] ProCite. http://www.procite.com.

[11] Reference Manager. http://www.refman.com.

[12] RefWorks. http://www.refworks.com.

[13] SIXPACK. http://sixpack.sourceforge.net.

THE PracTEX Journal

ACIDE: An Integrated Development Environment Configurable for LaTeX

Fernando Sáenz-Pérez

Abstract

This article introduces the configurable integrated development environment ACIDE, which is an ongoing development project currently in alpha status. It is cross-platform, open-source, and free, and will be distributed under GPL. Although targeted to any programming language environment, including compilers, interpreters, and database systems, in particular it is well-suited to tasks required for LaTeX and TeX document preparation systems. It manages projects, is useful in dealing with multifile documents, allows configurable menus, has a toolbar for executing commands and lexical tokens for syntax colouring, and even grammars to identify programming (syntactical) errors on the fly.

Fernando Sáenz-Pérez writes: I am an Associate Professor at the Department of Software Engineering and Artificial Intelligence at the Faculty of Computer Science of the Complutense University in Madrid (Spain). LaTeX was almost my first choice (after Chi-Writer) for writing technical documents on a workstation, when PCs were first introduced. Since then, I have used this document preparation system for writing papers, articles, books, slides, and my PhD thesis. My research interests include declarative programming (logic, functional, constraint programming), (deductive) databases, and natural language processing (ontologies). Other interests include videogames — I am involved in teaching in the class "Developing Videogames", and have developed an 80's videogame. I have been in close contact with companies in developing software for many years, and find this provides useful feedback from the real world as we continue to do university research. You may contact me at http://www.fdi.ucm.es/profesor/fernan.

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/12

ACIDE: An Integrated Development Environment Configurable for LaTeX

Fernando Sáenz-Pérez

Website http://www.fdi.ucm.es/profesor/fernan
Address Facultad de Informática, Universidad Complutense de Madrid, Spain

Abstract This article introduces the configurable integrated development environment ACIDE, which is an ongoing development project currently in alpha status. It is cross-platform, open-source, and free, and will be distributed under GPL. Although targeted to any programming language environment, including compilers, interpreters, and database systems, in particular it is well-suited to tasks required for LaTeX and TeX document preparation systems. It manages projects, is useful in dealing with multifile documents, allows configurable menus, has a toolbar for executing commands and lexical tokens for syntax colouring, and even grammars to identify programming (syntactical) errors on the fly.

Keywords: Tools, Free, Open-source, Cross-platform, Editor, IDE, LaTeX, TeX

1 Introduction

As a system implementor of declarative languages (e.g., the constraint functional logic language Toy [16], and the Datalog Educational System DES [17]), I needed an Integrated Development Environment (IDE) for each of them. Because of the need to work with several languages, developing a configurable IDE was almost a necessity. Therefore, during the course "Computing Systems" (which belongs to a postgraduate study), I directed a team of students in developing such a system which we called A Configurable IDE (hence the acronym ACIDE). The benefits are not only for system implementors who need an IDE for their systems, but also for other users, such as database users, and more importantly for readers of this journal, who can use it as already configured for LaTeX and TeX (cf. the suggestions in Section 4).

The requirements of this project were several-fold:

Configurable The main objective of this system is to be as highly configurable as possible, keeping the configurations easy and portable by means of text files. This configuration includes menus, a toolbar for executing commands, lexical tokens for syntax colouring, and even grammars to identify programming errors on the fly.

Simple We looked for, first, a simple, intuitive GUI. Second, a system that can be easily configured for a requested programming system, both the language lexicon as well as its grammar. Third, it must be easily installed, even requiring no installer. In fact, only decompressing an archive and executing a file should be enough to start working with the system. And, fourth, to have a simple multifile editor for basic users, which can also be used by advanced users who need extra features (projects, language configurations, etc., see Sections 2 and 3)

Internationalized In order to have a wider audience, text files are used to define all the texts of the GUI, so that the system can be easily localized to different languages.

Open-source We do believe that the open-source alternative will produce better software and systems, and allow developers to contribute enhancements. This is fitting in a university context, so that students will have sources available to learn and practice with them. In an unusual step, the project's main application has been developed without using other software as a basis. While doing this would have allowed more rapid development, starting from scratch gave the students a unique opportunity, one of the few they will probably have in their careers. Nonetheless, another free and open-source project ANTLR [2] has been used for parsing.

A Cross-Platform System Despite the basic need to develop the same system for several platforms, many of us use several different operating systems and therefore want to have the same applications available on several platforms. As a consequence, the selected development language was Java, which is widely known

and works on any platform. Of course, this was not the only alternative, since one can use C#, for example. But for developing a cross-platform system that can be widely used and modified in an open-source project, Java was chosen.

Free One of our goals in the university is to share knowledge, and we make our work freely available to others. Further, providing free software allows for widespread usage. In addition, our university and several national projects support us as researchers and lecturers (see Section 6), so we do not need additional income from developing the software.

Before starting to develop this project, other related systems were analysed. On the one hand, there are multi-file editors such as JEdit [9], which is free, cross-platform, configurable and with many nice features. However, JEdit is targeted as a file editor, not as an IDE, and lacks features such as parsing and shell. Crimson Editor [5] is a Windows editor also configurable and allows writing commands to send to a shell in order to do code compilations and run executables. However, it also lacks parsing capability, and moreover it is not cross-platform.

On the other hand, there are several IDEs which can be categorised into specific or general purpose. Specific IDEs can also be configured in various ways. For instance, WinEdt [14] is a Windows editor highly targeted to LaTeX and TeX documents, and it is quite powerful for this document preparation system. However, it is neither cross-platform, nor free, nor open-source. Other free IDEs are LaTeX Editor [10], WinShell [15], and TexnicCenter [13] (all of these only for Windows), Texmaker [12] (free and open-source), LyX (a WYSIWYM document processor for *nix platforms), and several others. JBuilder [7] and JCreator [8] are examples of commercial software for programming IDEs that also may provide hints for detected functionalities requested by users. But maybe the best exponent of an open-source, free project is Eclipse [6], which is a development platform comprised of extensible frameworks, tools and runtime libraries for building, deploying and managing software, and it comes with many programming environments already configured. The main drawback is that it cannot be easily configured to deal with programming environments other than the ones it uses.

Therefore, having stated our original goals and reviewed related systems, we conclude that no system completely fits our requirements, and that our proposal might be of interest for other users. In particular, LaTeX/TeX users might find an

alternative to existing editors. Note, however, that our system is emerging and is in a very early development stage, but its future development will be guided by the aforementioned objectives as well as user feedback.

The next sections describe the system. First, Section 2 states the concrete features of the system as well as its main current limitations. Section 3 describes ACIDE from a user point-of-view. Section 4 summarises an instance of the system configured for LaTeX. In Section 5 some conclusions and future work are noted. Finally, acknowledgements are offered (Section 6).

2 Features and Limitations

This section first describes the inherent features of ACIDE, although not all of them or their functionalities are fully operational in the current development stage. In addition, some limitations of the system are noted.

2.1 Features

- **Multi-File Editor**. Many files can be opened and two views for the same file and window are possible. The usual copy, paste, undo, and redo operations are possible via contextual menus and application menus. File printing is also provided. Syntax, programming errors, and delimiter pairs (parentheses, square brackets, ...) are highlighted.
- **Configurable.** Menus, toolbar, syntax highlighting (which includes tokens, delimiters, and remarks), parsing (from an EBNF language description), compiler, shell, and GUI language (currently localised to English and Spanish). Tool variables are provided to configure the commands assigned to the toolbar. For instance, `$activeFile$`, which keeps the complete file name of the selected file in the tool.
- **Text-based Configuration.** This tool allows configuration of all of its parameters by the use of text files, which permits using the provided dialog boxes without the need to resort to manually encode the data in the plain text files, as well as editing these files with a text editor, and even generate them with applications. For instance, one could generate the lexicon of a language from its formal description (e.g., EBNF).

- **Project Management.** Logical views of projects arranged in folders, which contains files or other folders. A concrete file can be selected as compilable or as a main file for a compilation. Also, several files can be selected for compilation by specifying the file extension.
- **Logging.** For implementation purposes, a log has been added, which helps in developing the tool.

2.2 Limitations

- **Macros.** They are quite helpful for defining scripts inside the tool. Currently, this task has to be passed to the operating system. That is, we can define a command as a system call, which can be a batch or script file. Toolbar commands might use these macros.
- **Debugging.** An almost indispensable feature for programming, but with current limitations: the tool can only handle debugging an interpreter with input/output via the standard streams. It can be used with toolbar buttons, therefore easing the task. Think, for instance, of the debugging model of Prolog.
- **Menu Commands.** There is a prefixed set of menu commands which can be enabled or disabled, but it is not possible to define new menu commands in the same way they are defined for the toolbar.
- **Command History.** This is quite useful for the shell, and also provides an autocompletion feature.
- **Dictionary**. A feature which operates the opposite of syntax highlighting, in that unrecognised words are highlighted, and in addition suggestions are provided for fixing errors.
- **Editor Facilities.** Other features can be useful, such as shorthand methods for completing text fragments (e.g., type `ltx` and get `\LaTeX`), text autocompletion, case changing, word wrapping, patterns (e.g., LaTeX environments as `itemize`, therefore automatically filling `\begin{itemize}`, `\item`, and `\end{itemize}`), autoindent, font selection for editor windows and console, and many other features.

In addition to this short list, it is possible to find many more limitations. Here, the most important ones, in our view, have been listed.

3 Description

3.1 Technology

The implementation of the tool has been completely done using Java under Eclipse. Version control was kindly provided by Berlios [3]. Students prepared the documentation using Rational Rose for UML diagrams, MS Project for Gantt charts, and MS Word. It seems that our graduate students do not feel the need for more powerful text editing when working with LaTeX.

3.2 The Main GUI

Figure 1 shows the main GUI of ACIDE. It consists of three main panels. The left panel shows the organisation of the current project, the MDI windows in the right are the opened files which may belong to the project (files may be opened without assigning them to the project). Below, the shell panel is shown, which allows user interaction. The case shown is a command console of Windows XP. Both the shell and project panels can be hidden. Moreover, there is no need to work with projects if this flexibility is not needed; a regular user may use the system as is. The status of the GUI is remembered for the next time the tool is executed. If the tool opens a project, its status when it was last saved is restored.

The menu bar includes some common entries:

- File. For file-related operations (New, Open, Close, Close All for closing all opened files, Save, Save As, Save All, Print and Exit). Missing useful options: Recent Files.
- Edit. For clipboard-related operations, search, undoing changes, and go to line number. Missing useful options: Select All, Insert Comment, Remove Comment.
- Project. For project-related operations (New Project, Open Project, Close Project, Save Project, Save Project As, Add File to the current project, Remove File from the current project, New Folder in the project structure, Delete Folder from the

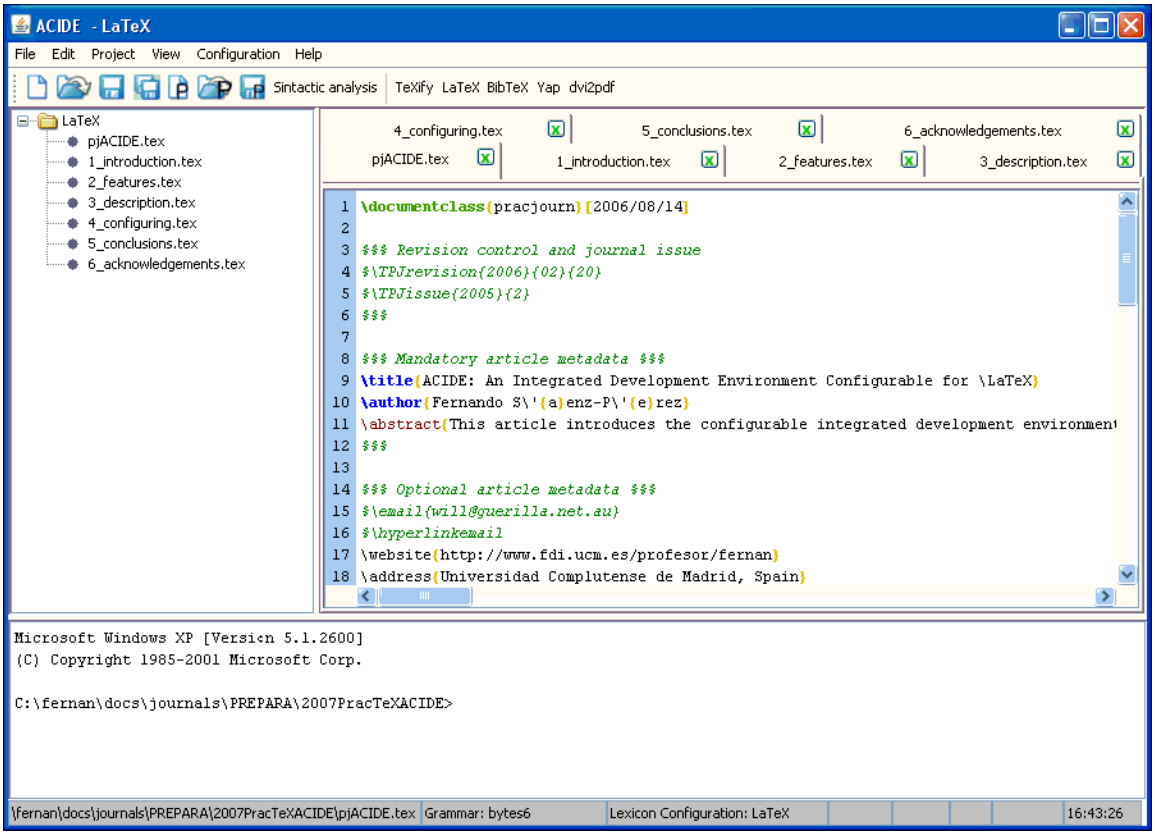


Figure 1: ACIDE Main GUI

project structure, Compile the specified files (see Section 3.5.3), Execute the result of the compilation, Set File as Compilable, Unset File, and Set as Main File). Missing useful options: Recent Projects.

– View. For showing/hiding project and shell panels, and displaying the log. Window arrangements are not possible up to now, but usual features are cascading and tiling windows both vertically and horizontally. Changing the order of the MDI windows (tabs) is not possible.

– Configuration. This entry allows to configure Lexicon (for syntax highlighting), Grammar (for parsing on-the-fly), Compiler (for compiling the project), Shell (the shell in the bottom panel), Language of the GUI, Menu entries that are displayed, and Toolbar for the commands, which can be displayed ei-

ther as icons or textual descriptions. Tooltips for toolbar commands can be configured.

- Help. This entry contains Show Help, and About ACIDE.

In addition, there is a fixed toolbar which includes common buttons for file and project-related basic operations: New, Open, Save, and Save All (this last one only for files). For development purposes, a button with label Syntactic analysis is included, and it is intended to test text parsing. Currently, a fixed grammar is included, but it will be specified by the user (see Section 3.5.2). Next to the fixed toolbar, there is the configurable toolbar (see Section 3.5.5).

Finally, the status bar gives information about some items: The complete path of the selected file, the selected grammar and lexicon, the line and column numbers, Caps Lock, Scroll Lock, Num Lock, and current time.

3.3 ACIDE System Variables

There are two system variables which are used for configuring commands in the toolbar, and also for configuring compiler options and the executable file:

- `$activeFile$`. Contains the complete file name of the current file (the one in the active MDI window).
- `$mainFile$`. Contains the complete file name of the main file (the one manually marked with Set as Main File).

In both cases, it is also useful to access only the name of the file or the extension, an issue to be addressed soon.

3.4 Projects

A project contains the whole status of a session, which is defined by all the possible configurations as well as the current display status. It consists of files arranged in folders (with any tree depth), all the configurations for the session (lexicon, grammar, compiler, shell, language, menu, and toolbar), main GUI arrangement (panel sizes, and opened files in the project), and file attributes. File attributes identify a file in a project as compilable and/or main. If a file is compilable, then the compiler configuration can be set to compile each of these files.

If a file is a main file (there is only one in the project), then it can be used in the compiler configuration or in the toolbar commands. For instance, a LaTeX project would compile the main file whether it is opened and active in the multifile panel or not. The filename of the main file can be accessed with the system variable `$mainFile$`.

The project structure shown in folders is a logical view which may coincide with the physical structure of the OS folders, but this is not needed. You can include in a given project a file belonging to another tree structure, therefore allowing to share files for different projects.

3.5 Configuration

Next, possible configurations are briefly described.

3.5.1 Lexicon

The lexicon for a programming or description language can be configured (as shown in Figure 2) for implementing a useful feature in text editors: syntax highlighting. Our tool allows defining the tokens and their associated format (colour, bold face and/or italics). Also, delimiters are needed to detect each token and these can also be declared. Line comments start with a given string; "%" in LaTeX, for instance. Also, a colour can be specified for comments.

3.5.2 Grammar

Both lexical categories and syntax rules can be defined to configure a grammar in EBNF (see Figure 3). Our tool allows the user to type or load those and create a parser, which is generated with the open-source, free project ANTLR [2] (ANother Tool for Language Recognition). With this tool the user will be able to activate parsing on-the-fly, a valuable aid in avoiding programming errors. This feature is still under development and it is expected to be completed by September 2007.

3.5.3 Compiler

A dialog box allows definition of the compiler to be used, its parameters, and the files to compile (see Figure 4). These files can be the files in the project that the

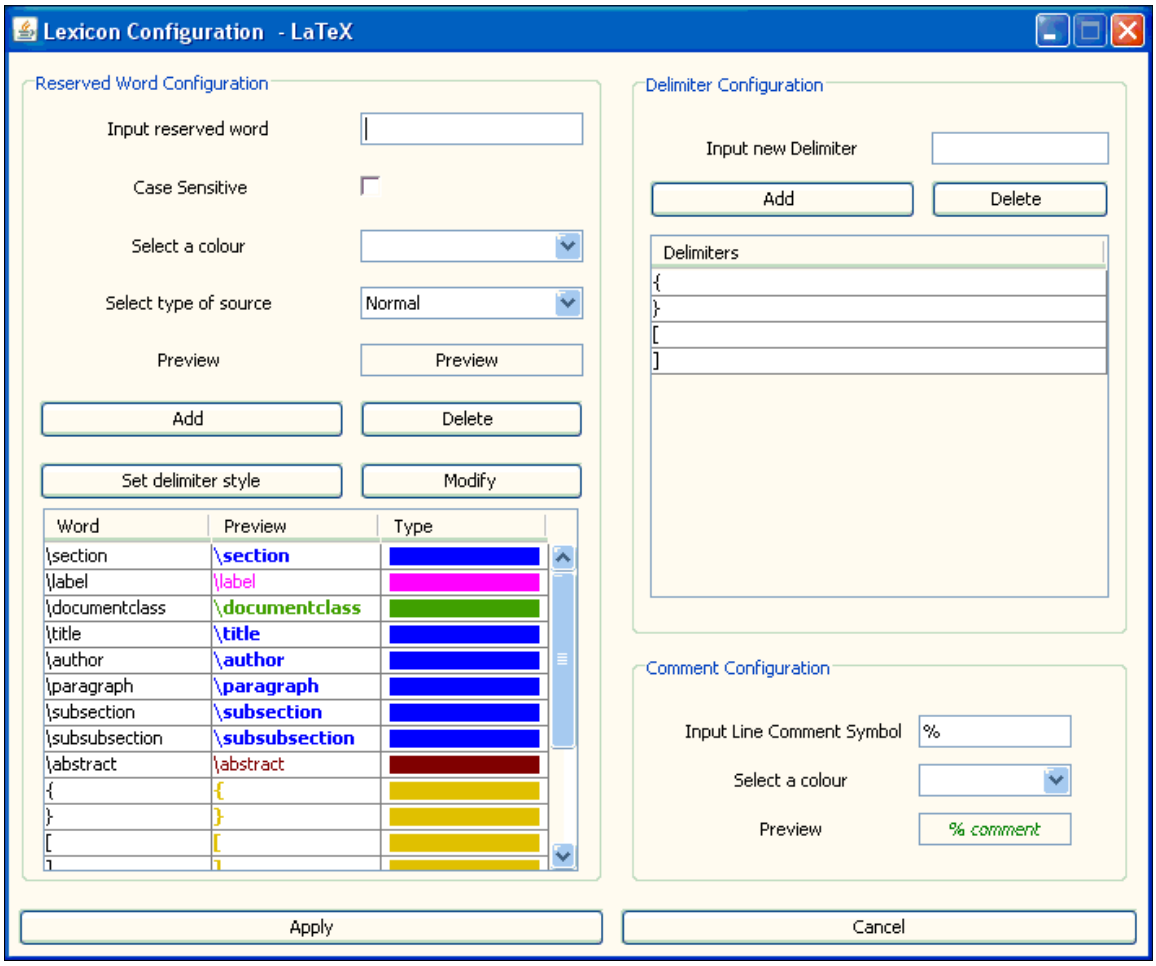


Figure 2: Lexicon Configuration

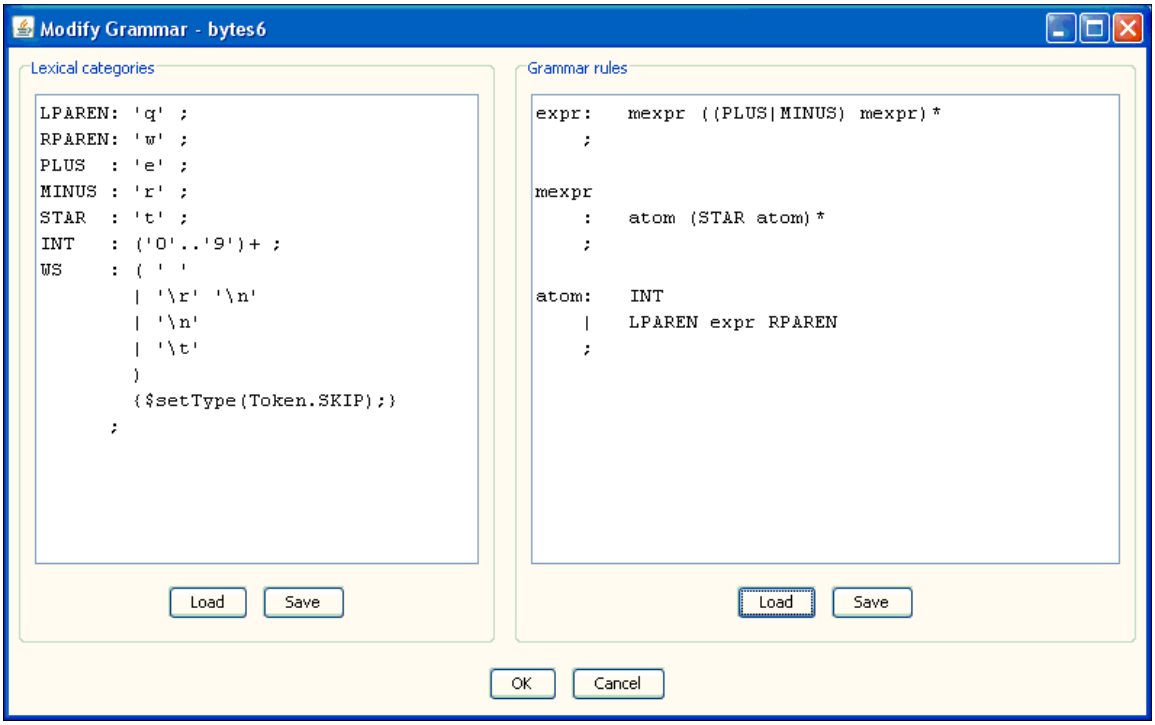


Figure 3: Grammar Configuration

user sets as compilable (a menu entry in the Project menu) or the files with a given extension (e.g., `.java`). Here, only the main file is compiled; there is no need to compile others. This dialog is useful for other programming environments in which several files must be compiled.

3.5.4 Executable

For programming projects that generate an executable, this dialog box allows the user to define the file's location as well as its arguments, if any (see Figure 5). The entry Execute in the menu Project will execute this file.

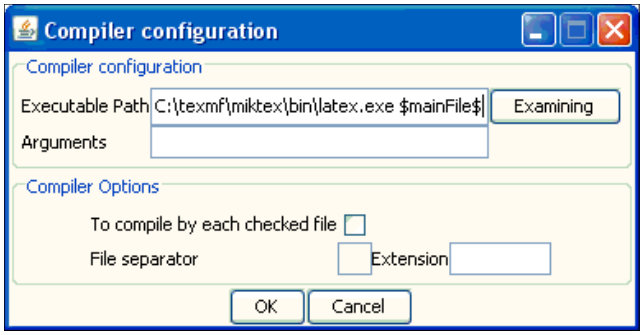


Figure 4: Compiler Configuration

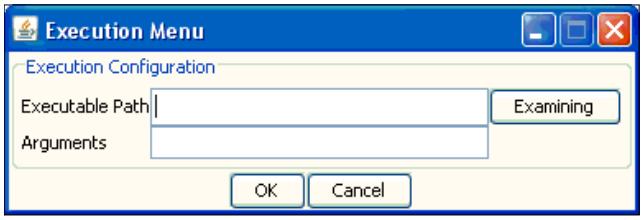


Figure 5: Executable Configuration

3.5.5 Toolbar

The toolbar can be configured by adding icons to the toolbar, which are associated with user-defined commands (see Figure 6). Currently, toolbar commands are sent to the console in the shell panel, but sending them as a separate process is also needed (a pending issue).

3.5.6 Menu

The menu bar can be configured by enabling or disabling predefined entries (see Figure 7). This configuration has to be extended to user-defined commands, as in the toolbar.

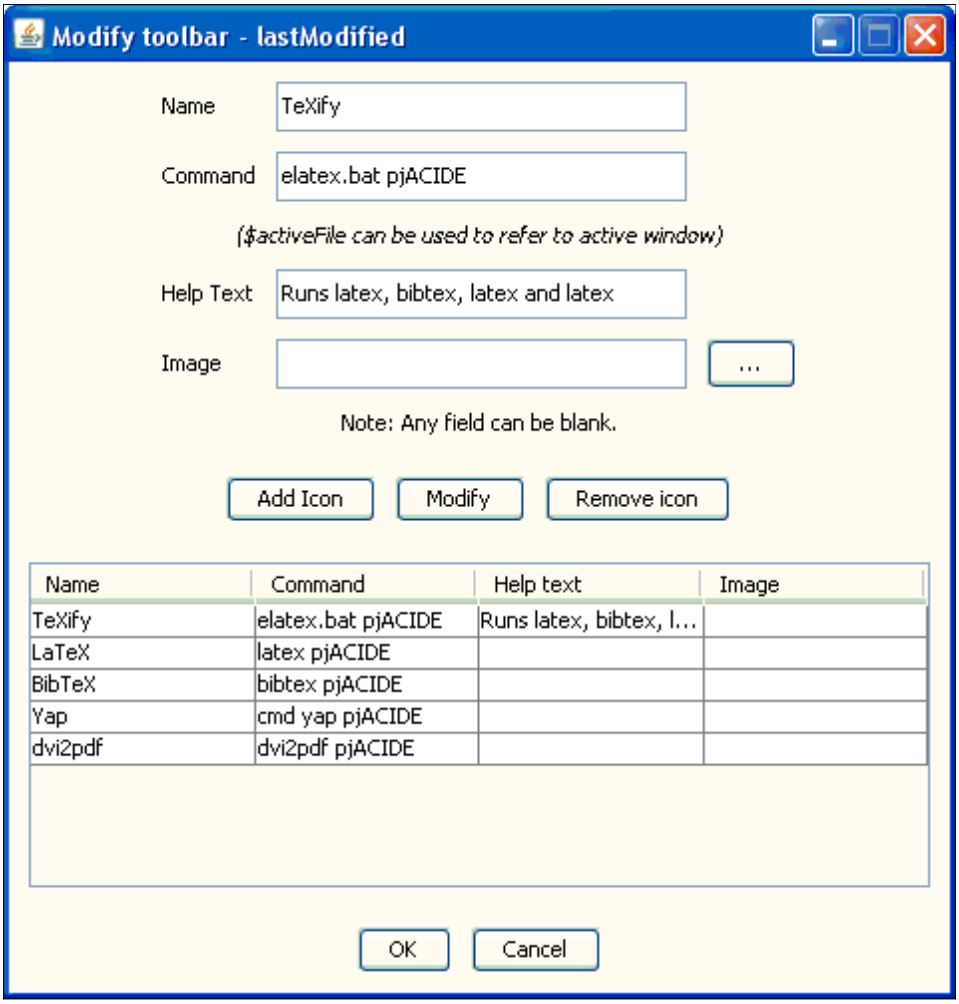


Figure 6: Toolbar Configuration

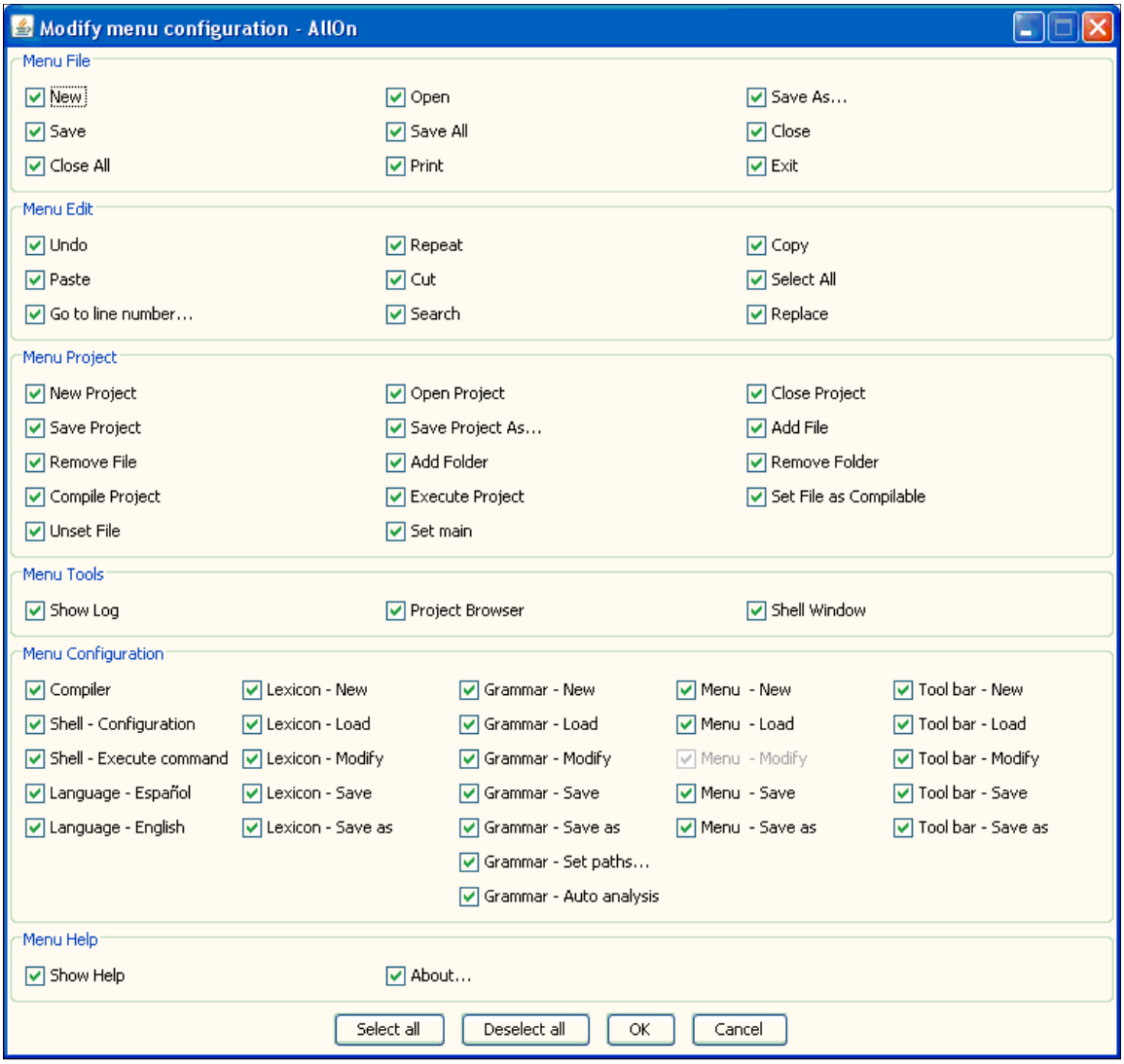


Figure 7: Menu Configuration

3.5.7 Shell

The shell can be configured with the dialog box seen in Figure 8. The check box is used for echoing the input command when the shell reads the input but does not output it.

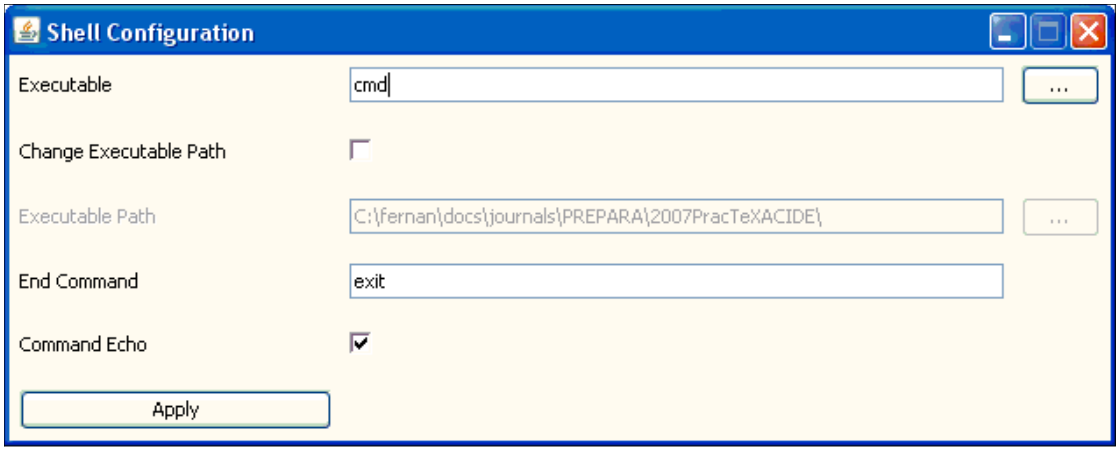


Figure 8: Shell Configuration

4 Configuring ACIDE to Work in a LaTeX Environment

Following the previous section, it becomes clear that ACIDE can be configured to work in a LaTeX/TeX environment. At least, in the current development stage, the following can be configured (actually, they are):

- Lexicon: For syntax highlighting as seen in Figures 1 and 2.
- Toolbar commands: TeXify (latex+bibtex+latex+latex), LaTeX (latex), BibTeX (bibtex), Yap, and dvi2pdf (dvi2pdf) among others (see Figures 1 and 6).
- Shell: For interacting with an operating system console (see Figures 1 and 8).
- Compile project: For compiling the main file (see Figure 4).

5 Conclusions and Future Work

This paper has presented ACIDE, an alternative to other available IDEs, with features meeting our initial requirements. It cannot be thought of as a complete tool since it is emerging and in an alpha development status. It should otherwise seen as a tool that, in time, might provide a steady stream of features following our requirements. For LaTeX/TeX users, there are much better tools in the free market today, but ACIDE might be competitive once it is more mature.

There are many goals yet to be completed. Some have been posed in Section 2 and others have been indicated in the following sections, while some have become evident after the tool description and configuration. Apart from fixing numerous bugs (including operating system dependent calls, which makes this tool work only for Windows, up to now), we have to implement state-of-the-art features in file editors, programming environments, and shell interactions. Despite the assumed limitations, we hope that this tool will consolidate into a friendly and powerful application, amenable to the users for whom it is designed; in particular, to system implementors, database users, document writers, and developers. It can be downloaded from `http://acide.sourceforge.net`. We hope to provide the version described in this paper as soon as possible, and to provide a better and more stable implementation by September 2007.

6 Acknowledgements

I am grateful to the students who developed this system under my guidance, which I hope will be useful for them as they develop their own projects in the future. These students are Diego Cardiel Freire, Juan José Ortiz Sánchez, and Delfín Rupérez Cañas, who worked on this project in the computer science course "Sistemas Informáticos" (Computing Systems) for graduates, a course intended for fifth-year students of the Computer Engineering studies of the Computer Science Faculty at the Universidad Complutense de Madrid, Spain. Also, thanks to the projects TIN2005-09207-C03-03, and S-0505/TIC0407 which supported this work. Finally, thanks to the free, open-source project ANTLR [2] which we use in our project, as well as to the Java and Eclipse initiatives. This article was prepared with MikTeX 2.4 [11], ACIDE [1] (cf. Figure 1), and BibMgr [4].

References

[1] ACIDE. A Configurable Integrated Development Environment. http://acide.sourceforge.net.

[2] ANother Tool for Language Recognition. http://www.antlr.org.

[3] Berlios. http://www.berlios.de.

[4] Bib Manager and Word Citer. http://bibmgr.sourceforge.net.

[5] Crimson Editor. http://www.crimsoneditor.com.

[6] Eclipse. http://www.eclipse.org.

[7] JBuilder. http://www.borland.com/jbuilder.

[8] JCreator. http://www.jcreator.com.

[9] JEdit. http://www.jedit.org.

[10] LaTeX Editor. http://www.latexeditor.org.

[11] MikTeX. http://www.miktex.org.

[12] Texmaker. http://www.xm1math.net/texmaker.

[13] TexNicCenter. http://www.toolscenter.org.

[14] WinEdt. http://www.winedt.com.

[15] WinShell. http://www.winshell.org.

[16] P. Arenas, A.J. Fernández, A. Gil, F.J. López-Fraguas, M. Rodríguez-Artalejo, and F. Sáenz-Pérez. $\mathcal{TOY}$. A Multiparadigm Declarative Language. Version 2.3.0, 2007. R. Caballero and J. Sánchez (Eds.), Available at http://toy.sourceforge.net.

[17] F. Sáenz-Pérez. Datalog Educational System User's Manual. Technical Report 139-04, Facultad de Informática, Universidad Complutense de Madrid, 2004. http://des.sourceforge.net.

THE PracTEX Journal

LaTeX and Subversion

Mark Eli Kalderon

Abstract

Subversion is a popular open source version control system. When writing complex LaTeX documents, it is useful to keep track of their development with a version control system such as Subversion. This article covers installation of Subversion, the creation of a local Subversion repository and working copy, the Subversion workflow, and how to get LaTeX to interact with Subversion with the svn-multi package.

I am Mark Eli Kalderon. I am an analytic philosopher teaching at University College London. I am also the current editor of the Aristotelian Society. I received my PhD from Princeton in 1995. I have taught at University of California Riverside, Princeton, UCLA, and Caltech. According to the present deformation professionnelle my work is in M&E (metaphysics and epistemology). My current research concerns color, consciousness, and metaethics.

You can reach Mark at eli@markelikalderon.com

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

LaTeX and Subversion*

Mark Eli Kalderon

Email eli@markelikalderon.com
Website http://markelikalderon.com
Address Department of Philosophy
University College London
Gower Street
London, WC1E 6BT
United Kingdom

Abstract Subversion is a popular open source version control system. When writing complex LaTeX documents, it is useful to keep track of their development with a version control system such as Subversion. This article covers installation of Subversion, the creation of a local Subversion repository and working copy, the Subversion workflow, and how to get LaTeX to interact with Subversion with the svn-multi package.

1 Introduction

Many of the tools that programmers use are, in fact, readily adaptable to the task of writing. Programmers and writers face at least one common problem, they need to track small changes over time. Programmers, being programmers, have written software to meet this particular need, software that can meet the writer's corresponding need. So-called *version control systems* (sometimes also called *source code management*), can help you keep track of the development of complex LaTeX documents. Further you can use it as collaborative tool.[1] Once you incorporate version control into your workflow, you will soon find it indispensable.

You might wonder what the big deal is. After all, many save their latest drafts in an *ad hoc* fashion, say, with multiple files. The problem is that this procedure is *ad hoc* and *unreliable*. Version control is not magic, it requires discipline, but it

*Thanks to Francisco Reinaldo for his help and encouragement.

1. See [2] for discussion of collaborative use of Subversion.

provides the framework for keeping track of revisions in a non-*ad-hoc* manner. It may well save you a lot of grief.

Subversion[2] is a popular open source version control system. In this article, I explain how to use Subversion in the production of LaTeX documents. In section 2, I describe the basic concepts of Subversion. In section 3, I explain how to install Subversion. In section 4, I explain how to create a local Subversion repository. In section 5, I explain how to check out a working copy. In section 6, I describe the basic Subversion workflow. Finally, in section 7, I explain how the svn-multi package[3] can help LaTeX interact with Subversion.

2 Subversion, Basic Concepts

In Subversion, a directory and its contents are kept in a Subversion *repository*. At the heart of a Subversion repository is a database that tracks not only the directory and its contents but, importantly, changes to that directory. The Subversion repository can reside locally on your machine, or, ideally, on a server.

Two observations are relevant.

First, when a change is committed to the repository, Subversion records four important pieces of information. Remember being taught journalism in high school? A good story must provide answers to four questions: "Who?", "What?", "When?", and "Where?" Or so we were taught. When you commit a change, Subversion records the "Who?", "What?", "When?", and "Where?" of the committed change:

1. Subversion records WHO committed the change
2. Subversion records WHAT the change was (a description of the change)
3. Subversion records WHEN the change was committed
4. Subversion records WHERE the change was made (which files or subdirectories were changed)

With each change, Subversion increments the revision number of the repository. The revision number represents the number of changes that have been made, or committed, to the Subversion repository.

2. http://subversion.tigris.org
3. CTAN:/macros/latex/contrib/svn-multi

Second, Subversion tracks changes to a *directory* and its contents. When a change is committed, the revision number of the *entire* repository is incremented. Whereas CVS[4], Subversion's predecessor, primarily tracks the history of revisions to individual files, Subversion primarily tracks the history of revisions to a directory.

Subversion uses a centralised model of version control. There is a central repository from which a local *working copy* may be *checked out*. You can think of a working copy as your own personal sandbox where you can freely modify the directory and its contents. Once you are satisfied with the changes you have made, you then *commit* these changes to the repository.

Some liken Subversion to a time machine. Wish you hadn't deleted that paragraph? With Subversion, the missing paragraph is just a `svn cat` away.

3 Installing Subversion

Subversion can be built from source or installed with a binary. Binaries exist for a number of operating systems including Red Hat Linux, Fedora Core, Debian GNU/Linux, FreeBSD, OpenBSD, NetBSD, Solaris, IBM i5/OS, Mac OS X, and Windows. A listing of these are available at the Subversion website[5]. One popular binary for Mac OS X is not listed on that page, Martin Ott's binary package[6].

To build Subversion from source, download the most recent distribution tarball from the Subversion website[7]. Unpack it, and use the standard *nix procedure to compile:

```
$ ./configure
$ make
$ sudo make install
```

Be sure to read the document "Install" first, though, to check for dependencies.

4. http://www.nongnu.org/cvs
5. http://subversion.tigris.org/project_packages.html
6. http://www.codingmonkeys.de/mbo
7. http://subversion.tigris.org/servlets/ProjectDocumentList

4 Creating a Local Subversion Repository

Creating a local Subversion repository is easy. It can be done with the following command:

```
$ svnadmin create /path/to/your/repository/
```

The command `svnadmin create` creates a new empty repository at the path provided as an argument. If the path provided does not exist, then Subversion will create it for you. So be careful: `/path/to/your/repository/` is just a dummy path, to be replaced with whatever you like.

The repository we just created is empty. We can *import* a directory and its contents into the empty repository as follows. First either copy a directory or create a directory in `/tmp/`:

```
/tmp/project/
```

(Again, `project/` is a dummy label.) Once you have populated `/tmp/project/` with whatever content you want, we are ready to begin. To import `/tmp/project/` into the newly created empty repository, we use the following command:

```
$ svn import -m "initial import" /tmp/project/
file:///path/to/your/repository/
```

The `svn import` command takes two arguments, the path of the directory to be imported and the `URL` of the repository. It can also take options. The option we have used, `-m`, is used to give a description of the change to the repository. The description follows the `-m` in double quotes. Since the change is an initial import of the directory, that is the description we have used. You can now delete `/tmp/project/` with:

```
$ rm -r /tmp/project/
```

but just to be safe, let's first *check out* the working copy.

5 Checking Out a Working Copy

Checking out a working copy is easy. And it only needs to be done *once*.

First, navigate to the directory where you want your working copy to be. Suppose you want to keep your working copy in your home directory, then use the command:

```
$ cd ~
```

If you want the working copy to reside elsewhere, simply replace the tilde with the desired path.

To check out a working copy, we now use the following command:

```
$ svn checkout file:///path/to/your/repository/ project
```

The command `svn checkout` has two arguments, the `URL` of the repository and the name of the working copy to be created by Subversion. Your working copy now resides at:

```
~/project/
```

Check the contents of your working copy with:

```
$ ls ~/project/
```

If the contents of your working copy are as expected, you can now safely delete `/tmp/project/`.

6 The Subversion Workflow

Okay. That was a bit of work setting things up. But now the fun begins.

There are four basic components to the Subversion workflow:

1. Updating your working copy
2. Making changes to your working copy
3. Reviewing your changes
4. Committing your changes to the Subversion repository

Let's review these in turn.

6.1 Updating

You should always begin your workflow by updating your working copy. This propagates any changes in the Subversion repository to your working copy. Updating is easy. At the root of your working copy simply issue the command:

```
$ svn up
```

Suppose that `foo.tex` has been changed since your last update, and it is the seventeenth revision of the repository, then the command will yield:

```
U foo.tex
Updated to revision 17
```

Thus, your working copy has been updated to reflect the changes to `foo.tex` and that it now reflects revision `17` of the repository. The letter code `U` is one of several codes that can be issued. These include:

- `U bar` — file `bar` has been updated;
- `A bar` — file `bar` has been added;
- `D bar` — file `bar` has been deleted;
- `G bar` — file `bar` has been updated but you have made local changes in your working copy not reflected in the repository but these changes do not conflict.

6.2 Changing

There are two kinds of changes that can be made in the working copy:

1. Changes to individual files;
2. Changes to the structure of the directory.

Changes to individual files are easy. Just make the changes in the appropriate editing program. You don't need to issue any Subversion specific commands.

Changes to the structure of the directory, however, require Subversion's help (so that Subversion knows what's happening to the directory). Fortunately, the relevant commands are easy to understand. To schedule the file or directory `foo` for addition or deletion from the repository the next time you commit, you use the following commands, respectively:

```
$ svn add foo
$ svn delete foo
```

To create a copy of `foo` called `bar` and schedule the addition of `bar` to the repository the next time you commit, you use the following command:

```
$ svn copy foo bar
```

To move `foo` to `bar`, you use the following command:

```
$ svn move foo bar
```

This creates a copy of `foo` called `bar` and schedules `bar` for addition and schedules `foo` for deletion the next time you commit it.

6.3 Reviewing

Before committing your changes, it is a good idea to review the changes you made. Helpfully, Subversion has provided some useful tools for doing this.

First up is the command:

```
$ svn status
```

If this command is issued at the root of your working copy it will yield a description of all the changes that you have made since last updating. The command yields a list of files and directories preceded by letter codes indicating the nature of the changes made. These include:

- `M foo` — `foo` has been modified since last updating;
- `A foo` — `foo` is scheduled for addition;
- `D foo` — `foo` is scheduled for deletion;
- `C foo` — `foo` conflicts with an update.

If `svn status` indicates that `foo` has been modified, you may want to examine these changes in detail in order to write a helpful commit message.

The command:

```
$ svn diff
```

run at the root of the working copy will print out a list of changes to modified files in `diff` format. Unfortunately, `diff` format displays *line* differences, not *word* differences. This is not great for prose, since paragraphs in text files are long lines so multiple differences in a paragraph will not be adequately represented. Fortunately, Subversion allows you to use external `diff` programs. To use an external `diff` command, preferably one that displays word differences, simply use the option:

```
--diff-cmd CMD
```

where `CMD` is the name of the external `diff` command. If the external `diff` command takes different arguments than `svn diff` you may need to write a wrapper script to use this option.

The command, `wdiff`,[8] is a frontend to GNU `diff` that displays word differences. It is also possible (and desirable) to use a GUI `diff` program. For example, on Mac OS X, if you install the developer tools, one gem that you will get is FileMerge, a descendant of NEXTSTEP's merge utility. FileMerge provides a visual comparison of text files. FileMerge can be invoked from the command line with `opendiff`. FileMerge, like GNU `diff`, captures line differences, but, like `wdiff`, it also highlights word differences. Unfortunately, `opendiff` and `svn diff` use different arguments. So to get `opendiff` to work with Subversion requires a wrapper script. Fortunately, Bruno De Fraine has done the heavy lifting. His script `fmdiff` is just such a wrapper. `fmdiff` can be downloaded (or checked out) from his website[9]. If you install `fmdiff` in `/usr/local/bin/` or `~/bin/` as is your preference and make it executable, you could then run the command:

```
$ svn diff --diff-cmd fmdiff foo.tex
```

to reveal in FileMerge the changes you have made to `foo.tex` since last committing, as can be seen in Figure 1.

6.4 Committing

So far, you have updated your working copy, made appropriate changes, and have reviewed these changes. If you are happy with these changes, it is time to com-

8. http://wdiff.progiciels-bpi.ca
9. http://ssel.vub.ac.be/svn-gen/bdefrain/fmscripts

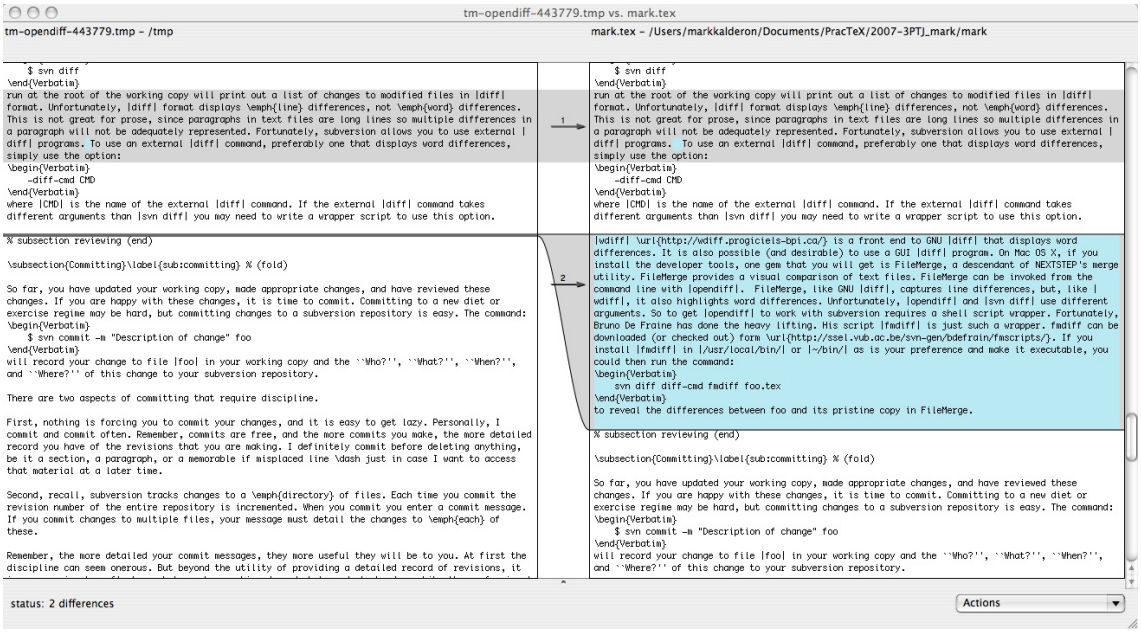


Figure 1: FileMerge.

mit. Committing to a new diet or exercise regime may be hard, but committing changes to a Subversion repository is easy. The command:

```
$ svn commit -m "Description of change" foo
```

will record your change to file `foo` in your working copy and the "Who?", "What?", "When?", and "Where?" of this change to your Subversion repository.

There are two aspects of committing that require discipline.

First, nothing is forcing you to commit your changes, and it is easy to get lazy. Personally, I commit and commit often. Remember, commits are free, and the more commits you make, the more detailed record you have of the revisions that you are making. I definitely commit before deleting anything, be it a section, a paragraph, or a memorable if misplaced line—just in case I want to access that material at a later time.

Second, recall, Subversion tracks changes to a *directory* of files. Each time you commit the revision number of the entire repository is incremented. When you commit, you enter a commit message. If you commit changes to multiple files,

your message must detail the changes to *each* of these.

Remember, the more detailed your commit messages, they more useful they will be to you. At first the discipline can seem onerous. But beyond the utility of providing a detailed record of revisions, it is an occasion to reflect on what you have achieved, and what needs to be done. Like the confessional for Catholics, or the weekly review for GTD geeks, the exercise of formal reflection can be good for you.

6.5 Summary

The Subversion workflow involves four basic components: updating, changing, reviewing, and committing. We have seen how to manage these on the command line. While the GUI's supplanting the command line represents the triumph of the Image over the Word, the GUI has its place — even in text editing. With respect to Subversion, there's cognitive utility in being able to visualise the structure of your repository or working copy. (For an extensive listing of GUI front ends on a variety of platforms see the Subversion website[10].) While there are GUI front ends for Subversion, none are perfect. A compelling solution to graphically representing Subversion repositories and working copies has yet to be found. And no GUI front end has the power, speed, and flexibility of Subversion on the command line. So take the time to learn the Subversion commands. The basics can be picked up in an afternoon.

7 The svn-multi Package

LaTeX is great for managing complex documents. When working on a complex LaTeX document, it is important, nay, *imperative* to keep it in some form of version control. (The necessity may not be apparent until you actually *use* a version control system such as Subversion — if you don't, well, just trust me, at least for now.)

If you are keeping your complex LaTeX document in a Subversion repository, it would be useful to include information about the current revision, the "Who?", "What?", "When?", and "Where?" of the last committed change, within your

10. http://subversion.tigris.org/links.html

document. You could enter this information by hand, but that would be tedious and unreliable. Fortunately, there are LaTeX packages that can help. There are three:

1. svn[11]
2. svninfo[12]
3. svn-multi (aka svnkw)[13]

The svn-multi package suits my workflow best. Here's why. The svn package doesn't work if your LaTeX document comprises multiple files. Since I sometimes work with a master document that includes several separate files, that's out. The svninfo and svn-multi packages do not share this limitation. The svninfo package, however, itself has an important limitation. It only registers revision information when you *check out* the document. Thus, if you create a LaTeX document, add it to your working copy, and commit the addition, the revision information, the "Who?", "What?", "When?", and "Where?", will never be registered. So that's out. That leaves svn-multi — which has worked well for me. Let me detail how I use it.

The svn-multi package does two things:

1. It registers revision information, the "Who?", "What?", "When?", and "Where?";
2. It displays this information in your LaTeX document.

7.1 Registering Revision Information

In the preamble, I have the following:

```
\usepackage{svn-multi}
\svnidlong
{$LastChangedBy$}
{$LastChangedRevision$}
{$LastChangedDate$}
{$HeadURL$}
```

11. CTAN:http://tug.ctan.org/tex-archive/macros/latex/contrib/svn
12. CTAN:http://tug.ctan.org/tex-archive/macros/latex/contrib/svninfo
13. CTAN:http://tug.ctan.org/tex-archive/macros/latex/contrib/svn-multi

The command:

```
\svnidlong
{$LastChangedBy$}
{$LastChangedRevision$}
{$LastChangedDate$}
{$HeadURL$}
```

registers WHO made the change, WHAT the change was, WHEN the change was made, and WHERE the change is — the `URL` of the changed file. (Strictly speaking, `{$LastChangedRevision$}` registers, not what the change was, but a revision number that represents this. If `n` is the revision number, a detailed description of what has changed will be given by the command `svn log -r n foo.tex`, where `foo.tex` is the name of your LaTeX source)

Now we need to get these commands to interact with Subversion. To do so, navigate to the directory containing your LaTeX source, `foo.tex` and enter the following command in the terminal *once*:

```
$ svn propset svn:keywords 'LastChangedBy LastChangedRevision
LastChangedDate HeadURL' foo.tex
```

This will return the following:

```
property 'svn:keywords' set on 'foo.tex'
```

When you next commit a change to your repository or receive a change by updating your working copy, this information will be registered.

7.2 Displaying Revision Information

Now that we have registered the revision information, we want to use this information, to display it somewhere appropriate in our LaTeX document. Fortunately, the svn-multi package provides commands to do just that.

We want to display the following information:

- the author of the revision;
- the revision number of the last change;
- the date of the last change;

– the URL of the file.

The command:

```
\svnauthor
```

displays the author registered by:

```
{$LastChangedBy$}
```

in the preamble. This is the author of the last committed change to the document. The command:

```
\svnrev
```

displays the revision number registered by:

```
{$LastChangedRevision$}
```

in the preamble. This is the revision number of the last committed change to the document. The command:

```
\svndate
```

displays the date registered by:

```
{$LastChangedDate$}
```

in the preamble. This is the date of the last committed revision in the repository. Finally, the command:

```
\svnkw{HeadURL}
```

displays the URL registered by:

```
{$HeadURL$}
```

in the preamble. This is the URL of the file as it resides in the Subversion repository.

Let's consider one possible use of these commands and some possible refinements. Consider the following minimal example:

```
\documentclass{article}
\usepackage{svn-multi}
\svnidlong
{$HeadURL$}
{$LastChangedDate$}
{$LastChangedRevision$}
{$LastChangedBy$}
\begin{document}
    Hello World!
\end{document}
```

Suppose we want to include the revision information in a final footnote of the article. We might append the following command to `Hello World!`:

```
\footnote{\svnauthor \svnrev \svndate \svnkw{HeadURL}}
```

That's okay, to a first approximation. But it would be useful to label this information. Thus, for example, the value of:

```
\svnrev
```

is just a number. A label would provide some useful context for understanding what that number represents. Indeed, labels would provide useful context for understanding each piece of information. Moreover, it would be useful to clearly distinguish this information, so let's separate them with semicolons. This gives us:

```
\footnote{Author: \svnauthor; Revision: \svnrev; Last changed
on: \svndate; URL: \svnkw{HeadURL}}
```

Two further refinements. If we load the hyperref package and use the `\url` command we can add a hyperlink to the file in the Subversion repository:

```
\footnote{Author: \svnauthor; Revision: \svnrev; Last changed
on: \svndate; URL: \url{\svnkw{HeadURL}}}
```

And if we load the url package, urls will be correctly formatted.

Our minimal example would then be as follows:

```
\documentclass{article}
\usepackage{svn-multi}
\svnidlong
{$HeadURL$}
{$LastChangedDate$}
{$LastChangedRevision$}
{$LastChangedBy$}
\usepackage{url}
\usepackage{hyperref}
\begin{document}
    Hello World!\footnote{Author: \svnauthor; Revision:
    \svnrev; Last changed on: \svndate; URL:
    \url{\svnkw{HeadURL}}}
\end{document}
```

Earlier, I mentioned that one advantage that the svn-multi package has over the svn package is its support for LaTeX documents that comprise multiple files. So, suppose you have a master file, `master.tex`, that *includes* the files, `chapter1.tex` and `chapter2.tex`. All three of these files need to be in version control, and so we need to register the revision information for *each*. Thus the command:

```
\svnidlong
{$HeadURL$}
{$LastChangedDate$}
{$LastChangedRevision$}
{$LastChangedBy$}
```

needs to occur in `master.tex`, `chapter1.tex`, and `chapter2.tex`. And for each of these documents you need to set the svn keywords. So you would need to run:

```
$ svn propset svn:keywords 'LastChangedBy LastChangedRevision
LastChangedDate HeadURL' master.tex
$ svn propset svn:keywords 'LastChangedBy LastChangedRevision
LastChangedDate HeadURL' chapter1.tex
$ svn propset svn:keywords 'LastChangedBy LastChangedRevision
LastChangedDate HeadURL' chapter2.tex
```

Please see the documents, `example_main.tex`[14], `example_chap1.tex`[15], and `example.pdf`[16] that come with the svn-multi package for a detailed example of usage.

7.3 Initial Workflow

Call our minimal example `example.tex`. To get the desired result, the following steps must be taken:

1. Add `example.tex` to your Subversion repository:

   ```
   $ svn add example.tex
   ```

   This should return:

   ```
   A example.tex
   ```

2. Commit the addition:

   ```
   $ svn commit -m "Added example.tex" example.tex
   ```

   This should return:

   ```
   Adding example.tex
   Transmitting file data.
   Committed revision n
   ```

   where `n` is a dummy letter representing the current revision number of your repository.

3. Set the Subversion keywords:

   ```
   $ svn propset svn:keywords 'LastChangedBy LastChangedRevision
   LastChangedDate HeadURL' example.tex
   ```

   This should return:

   ```
   property 'svn:keywords' set on 'example.tex'
   ```

4. Update the working copy:

14. CTAN:http://www.cam.ctan.org/tex-archive/macros/latex/contrib/svn-multi/example_main.tex
15. CTAN:http://www.cam.ctan.org/tex-archive/macros/latex/contrib/svn-multi/example_chap1.tex
16. CTAN:http://www.cam.ctan.org/tex-archive/macros/latex/contrib/svn-multi/example.pdf

```
$ svn up
```

This should return:

```
At revision n
```

5. Check the status of the working copy:

```
$ svn status
```

This should return:

```
MM example.tex
```

meaning that `example.tex` has been modified (since the values of the Subversion keywords have been added to the variables in the `\svnidlong` command).

6. Commit these changes:

```
$ svn commit -m "Svn keywords set on example.tex" example.tex
```

This should return:

```
Sending example.tex
Transmitting file data.
Committed revision n+1
```

7. Typeset `example.tex` *twice*. The first run generates an auxiliary file called `example.svn` whose content is added to the output of `example.tex` on the second run.

8 Conclusion

Some form of version control should really be at the heart of any writer's workflow. This article has discussed some of the basics of using one popular version control system, Subversion, with LaTeX. I have really just scratched the surface. There is a lot more useful functionality. For more information about Subversion see Collins-Sussman, et al. [1]. This book is available free online in both HTML and PDF format[17]. The package documentation for svn-multi is available

17. http://svnbook.red-bean.com

at CTAN[18]. And remember, commits are free — commit often.

References

[1] Ben Collins-Sussman, Brian W. Fitzpatrick and C. Michael Pilato, 2004. *Version Control with Subversion*. Sebastopol, CA: O'Reilly. URL: http://svnbook.red-bean.com/

[2] Charilaos Skiadas, Thomas Kjosmoen, and Mark Eli Kalderon, 2007. "Subversion and TextMate: Making collaboration easier for LaTeX users". *The PracTeX Journal*, 3.

18. CTAN:http://www.ctan.org/tex-archive/macros/latex/contrib/svn-multi/svn-multi.pdf

THE PracTEX Journal

Using Assembla in PracTeX Production

Mark Eli Kalderon

Abstract

A short introduction to the use of Assembla's tools and services in the collaborative production of your PracTeX article.

I am Mark Eli Kalderon. I am an analytic philosopher teaching at University College London. I am also the current editor of the Aristotelian Society. I received my PhD from Princeton in 1995. I have taught at University of California Riverside, Princeton, UCLA, and Caltech. According to the present deformation professionnelle my work is in M&E (metaphysics and epistemology). My current research concerns color, consciousness, and metaethics.

You can reach Mark at eli@markelikalderon.com

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/28

Using Assembla in PracTEX Production

Mark Eli Kalderon

Email eli@markelikalderon.com
Website http://markelikalderon.com
Address Department of Philosophy
University College London
Gower Street
London, WC1E 6BT
United Kingdom

Abstract A short introduction to the use of Assembla's tools and services in the collaborative production of your PracTEX article.

1 Overview

So PracTEX has accepted your proposed article or has invited you to write an article. Congratulations! From here on out, you will be working with the editor, a production editor, and any coauthors you may have to produce your masterpiece. A production editor will be working closely with you, offering advice about revisions and answering any technical questions you may have. This is a coordination problem of potentially international proportions—all of these individuals may very well be scattered quite literally across the globe. How are they to effectively coordinate to help you write your completed article?

Fortunately, PracTEX appreciates the challenges of working with a globally distributed team and has addressed these challenges by using Assembla's tools and services[1]. Having accepted your proposal, or issued you an invitation, the editor will send you an e-mail inviting you to join an Assembla team. This e-mail will contain a url leading to your start page where you can accept or decline your invitation, Figure 1. To accept the invitation simply click on **Accept**.

1. http://www.assembla.com

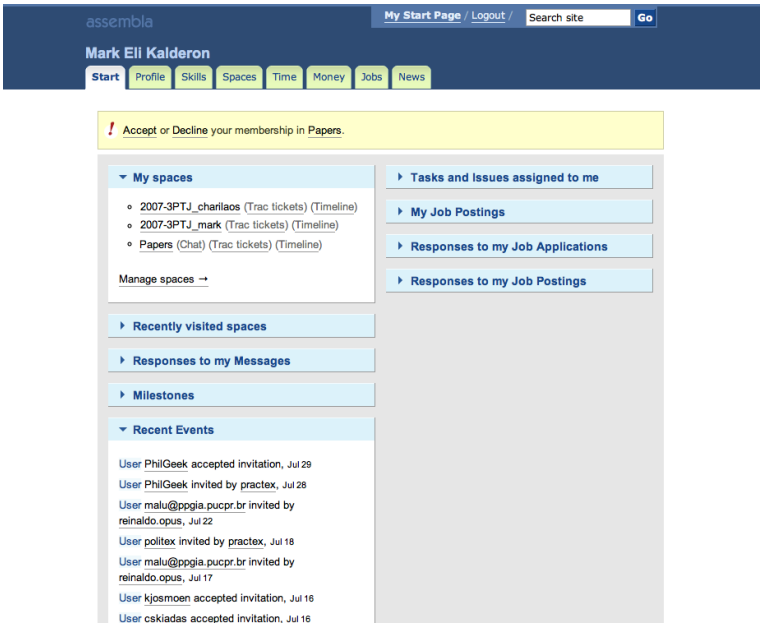


Figure 1: Assembla Start Page

On your start page, there is a portion called "My spaces". Think of a space as where your project lives. There will be a list of one or more project spaces, depending on how many PracTeX projects you are involved with. Each space is associated with three links. The first link leads to your project space's page, the second to a ticket list, and the third to a timeline. Sometimes, a chat tool is added. The ticket list and timeline are part of a trac tool to help you and your team track the progress of your article. Let's review these in turn.

If you click on your project space's page you can find a link to the trac tool, the ticket list, the timeline, and the Subversion url for the space. If activated by the admin, the chat tool can be accessed in My spaces or by a tab in the project space. As you may know, Subversion[2] is a free, open source version control system that keeps the history of the revisions of your document in an encrypted database that only you, your production editor, and any coauthors have access to. When a revision is committed to the Subversion repository, you, your production editor, and any co-author's will receive an e-mail notifying them of the committed

2. http://subversion.tigris.org

revision along with the commit message. This e-mail will contain a url of the changeset, or diff, representing the changes that have been made. If you click on this link, it will open this page in your browser, Figure 3. Changesets will be explained further later in this section.

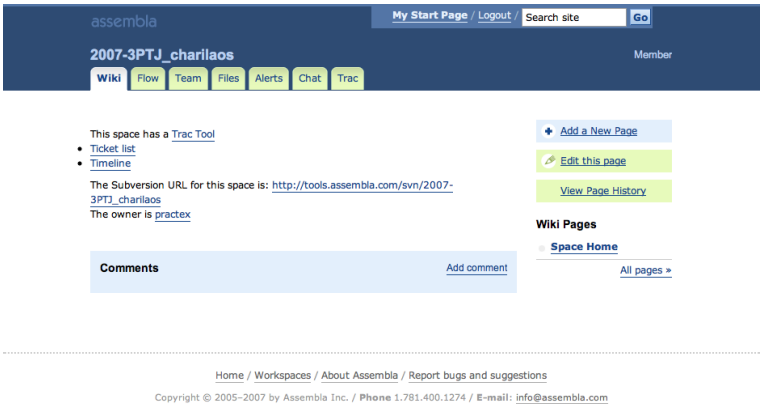


Figure 2: Project Space

The ticket list is a list of prioritized messages about what needs to be done or comments, more generally, about revisions of your paper. Tickets offers several actions, such as active (leave the ticket open—the problem or issue it raises still persists) or closed (the problem or issue is now fixed or otherwise resolved). The link from your space's page leads to the list of your active tickets. These are comments or queries that need to be addressed. To view the ticket simply click on an item on the list. The ticket system is an efficient way for the production editor to communicate with you or for you to communicate with your production editor or coauthors. To issue a ticket simply click on "New Ticket" on the trac tool and fill in the relevant fields. When a new ticket is issued, you will receive an e-mail notification.

The timeline is a list of project-related events ordered by date. These include tickets issued and changesets (a representation of the changes you or your coauthors have made to your paper since the previous committed revision). A changeset or diff only really works on plain text documents, so if you have committed

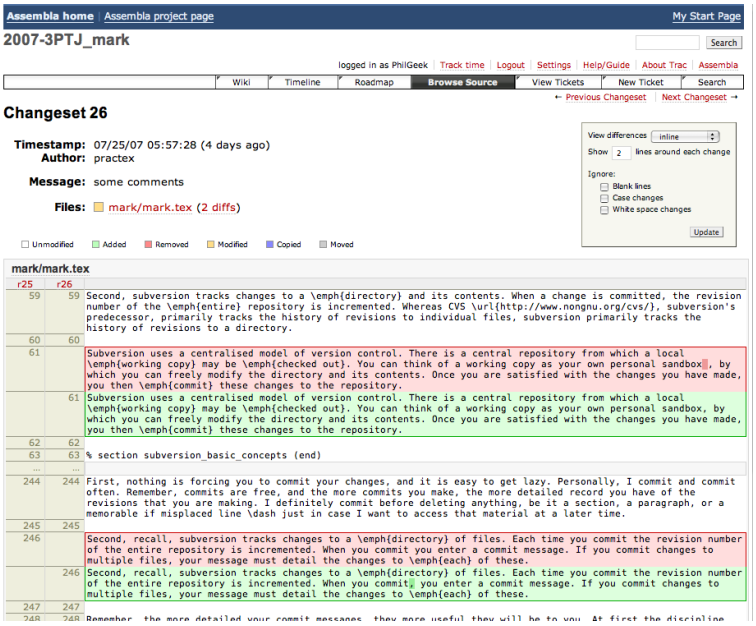


Figure 3: Changeset

the PDF generated by your LaTeX source, or any associated graphics, the results can look scary. But don't be put off, because the changesets for your LaTeX source are as informative as the changesets of binary files are ugly. (It is unnecessary an undesirable, anyway, to commit the PDF and auxiliary files such as .aux or .log files.)

The changesets can be accessed another way. In the trac tool, click on "Browse Source". You will see a folder representing your project materials. Clicking on this folder will reveal its contents. If you click on your LaTeX source, it will be displayed in your web browser. At the bottom of this page is a button called "View changes". This is a link to a page that asks you for two revisions. Say you want to compare the 29th and 26th revision. By clicking on "View changes" again, the changeset, or diff, will be displayed.

The chat tab on the project space's page will open a page where you can chat with the production editor or any coauthors, if they are online. See Figure 4. The right column lists who on your team is available to chat and a link to view the history of the chat sessions. To chat, simply type your message in the text

field and press return. The ability to chat online with your production team and coauthors is an incredibly useful tool that can speed up the production process.

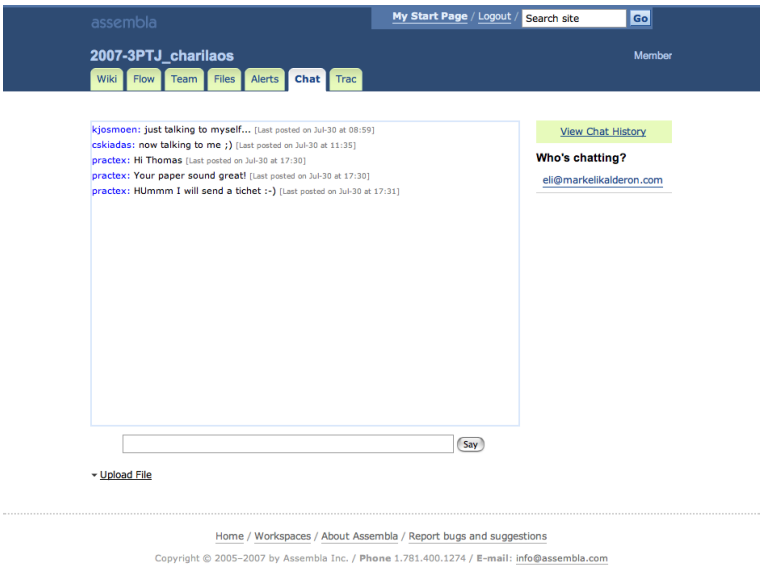


Figure 4: The Chat Page

The trac tool associated with your space is an efficient means for you and your global team to bring your masterpiece to fruition. There's no need for the UN to solve this international coordination problem. PracTEX and Assembla has solved it for us all.

2 Getting Started with Subversion

At the heart of Assembla's suite of tools is the Subversion repository that tracks the history of the revisions of your paper. Here are some instructions, organized by platform, about how to get started using Subversion in writing your PracTEX article.

As I mentioned to you before, the PracTEX production team will create your project's Subversion repository for you on the Assembla server. From this point what you need is to do is described in one of the three following subsections

depending on your operating system (Windows, Mac OS X, or *nix). Pick the appropriate subsection and perform the steps presented below to be synchronized with your team.

2.1 Windows

A Windows binary for Subversion is available online[3]. Get the best (that is, the latest) available version of Subversion. Please download and install this file. You should also download and install the latest version of TortoiseSVN[4]. TortoiseSVN allows you to access Subversion commands and view the status of your files all within Windows explorer. For more information see the TortoiseSVN website[5].

1. You need to create a folder for your working copy. Call it **PTJ-workplace**. This is where your project files must be.
2. Right click on **PTJ-workplace** folder and choose **Checkout** from menu. This action only needs to be done *once*.
3. In the dialog window paste the url of your project's repository. The url will be of the form:

   `http://tools.assembla.com/svn/2007-3PTJ_surname/surname-theme`

   and will be provided by the production editor. A link to it can also be found on your project's space (see Figure 2).
4. Click on the **Checkout** button.
5. A dialog box will ask you if the login and password can be saved for further access. Click on the **OK** button.
6. Enter your username and password previously created when you accepted the Assembla invitation.
7. Again, click on the **OK** button. Notice that the files stored in the Subversion repository are now being downloaded to your local **PTJ-workplace**.
8. Open your **PTJ-workplace** folder.

3. http://subversion.tigris.org/project_packages.html
4. http://sourceforge.net/projects/tortoisesvn
5. http://tortoisesvn.tigris.org

9. Open your LaTeX source in your LaTeX-aware editor, edit it, and save the changes. If you do not already have a LaTeX-aware editor, you might try LEd[6] or PCTEX[7].
10. Notice, after you have saved your changes, the file icon will be changed to **(!)**. This means that your source has changes not yet committed to the Subversion repository.
11. Right click over the file with **(!)** and choose **Commit**.
12. This will open a dialog window. Briefly describe the changes you have made to your text. This will be useful to you as a record of your revisions as well as to the production editor and your coauthors, if any, so that they know what you have done.
13. Click **OK**.

Congratulations! You have just committed your first revision to your space in Assembla's Subversion repository.

Before your next editing session be sure to update your working copy by right-clicking on **PTJ-workplace** and choosing **Update**. This will propagate any changes committed to the Subversion repository by the production editor or your coauthors to your working copy. After that, follow steps 8–13 to make further changes and commit these.

2.2 Mac OS X

A Mac OS X binary for Subversion is available online[8]. Get the best (that is, the latest) version. Unzip it, double click on the package installer, and follow the instructions to install. You should also download and install the latest version of SCPlugin[9]. The goal of the SCPlugin project is to integrate Subversion into the Mac OS X Finder. For more information about installation see the SCPlugin website[10]. If you are comfortable with the command line you can also follow the *nix instructions in subsection 2.3.

6. http://www.latexeditor.org
7. http://www.pctex.com
8. http://homepage.mac.com/martinott
9. http://scplugin.tigris.org/servlets/ProjectDocumentList
10. http://scplugin.tigris.org/installation.html

1. You need to create a folder for your working copy. Call it **PTJ-workplace**. This is where your project files must be.
2. Right click on **PTJ-workplace** folder and choose **Checkout** from the Subversion submenu. This action only needs to be done *once*.
3. In the dialog window paste the url of your project's repository. The url will be of the form:

   `http://tools.assembla.com/svn/2007-3PTJ_surname/surname-theme`

   and will be provided by the production editor. A link to it can also be found on your project's space (see Figure 2).
4. Click on the **Checkout** button.
5. A dialog box will ask you if the login and password can be saved for further access. Click on the **Always allow** button.
6. Enter your username and password previously created when you accepted the Assembla invitation.
7. Again, click on the **OK** button. Notice that the files stored in the Subversion repository are now being downloaded to your local **PTJ-workplace**.
8. Open your **PTJ-workplace** folder.
9. Open your LaTeX source in your LaTeX-aware editor, edit it, and save the changes. If you don't already have a LaTeX-aware editor, you might try TeXShop[11] or TextMate[12].
10. Notice, after you have saved your changes, the file icon will be changed to **(!)**. This means that your source has changes not yet committed to the Subversion repository.
11. Right click over the file with **(!)** and choose **Subversion Commit**.
12. This will open a dialog window. Briefly describe the changes you have made to your text. This will be useful to you as a record of your revisions as well as to the production editor and your coauthors, if any, so that they know what you have done.
13. Click on the **OK** button.

11. http://www.uoregon.edu/~koch/texshop
12. http://macromates.com

Congratulations! You have just committed your first revision to your space in Assembla's Subversion repository.

Before your next editing session be sure to update your working copy by right-clicking on **PTJ-workplace** and choosing **Update**. This will propagate any changes committed to the Subversion repository by the production editor or your co-author to your working copy. After that, follow steps 8–13 to make further changes and commit these.

2.3 Unices

To build Subversion from source, download the most recent distribution tarball[13]. Unpack it, and use the standard *nix procedure to compile:

```
./configure
make
sudo make install
```

Be sure to read the document "Install" first, though, to check for dependencies.

1. Create the directory `PTJ-workplace`:

   ```
   mkdir PTJ-workplace
   ```

2. Navigate to `PTJ-workplace`:

   ```
   cd PTJ-workplace
   ```

3. Check out your working copy:

   ```
   svn co --username username --password password http://
   tools.assembla.com/svn/2007-3PTJ_surname/surname-theme
   ```

   `username` and `password` were previously created when you accepted the Assembla invitation. The url will be of the form:

   ```
   http://tools.assembla.com/svn/2007-3PTJ_surname/surname-theme
   ```

   and will be provided by the production editor and a link to it can be found on your project's space (see Figure 2). The files in your Subversion repository are now being downloaded to your local `PTJ-workplace`. This only needs to be done *once*.

13. http://subversion.tigris.org/servlets/ProjectDocumentList

4. Open your LaTeX source in your LaTeX-aware editor, edit it, and save the changes. If you don't already have a LaTeX-aware editor, you might try Kile[14]; emacs and vim also both have excellent LaTeX environments.

14. http://kile.sourceforge.net

5. Commit your changes:

```
svn commit -m "your commit message" foo.tex
```

   `"your commit mesasge"` should briefly describe the changes you have made to the text. This will be useful to you as a record of your revisions as well as to the production editor and your coauthors, if any, so that they know what you have done.

Congratulations! You have just committed your first revisions to Assembla's Subversion repository.

Before your next editing session be sure to update your working copy by navigating to `PTJ-workplace` and entering `svn up`. This will propogate any changes committed to the Subversion repository by the production editor or your coauthor to your working copy. After that, follow steps 4–5 to make further changes and commit these.

THE PracTEX Journal

Subversion and TextMate: Making collaboration easier for LaTeX users

Charilaos Skiadas, Thomas Kjosmoen and Mark Eli Kalderon

Abstract

This article focuses on the Subversion version control system, and in particular its integration into the TextMate text editor. TextMate is a text editor, for the Mac OS X operating system, that has excellent support for working with LaTeX documents and projects, and its seamless Subversion integration makes it very easy to add version control to your workflow.

Version control systems have been used by computer programmers for many years for work on projects involving multiple authors. They have been invaluable in keeping track of the contributions of each party to the project, the changes that took place and when they happened, and so on. Subversion is one of the most popular version control systems today, and it is integrated into many text editors and online collaboration sites.

Charilaos (Haris) Skiadas is an Assistant Professor of Mathematics at Hanover College, and has been using LaTeX for all his writing ever since he found out about it ten years ago. For the last two years he has been very actively involved in the development of the LaTeX support for the TextMate text editor, one of the popular text editors for the Mac OS X operating system. His blog (http://skiadas.dcostanet.net/afterthought) contains a lot of information and screencasts related to LaTeX in TextMate.

Thomas Kjosmoen is a PhD candidate in the Department of Computer Science and Electrical Engineering at the University of Stavanger, Norway, where he works on bioinformatics and genomic signal processing. Ever since he discovered LaTeX about four years ago, it has filled most of his document needs. He particularly appreciates the open nature of the LaTeX documents and the fact that it makes them easy to keep under version control." Homepage: http://kjosmoen.org

I am Mark Eli Kalderon. I am an analytic philosopher teaching at University College London. I am also the current editor of the Aristotelian Society. I received my PhD from Princeton in 1995. I have taught at University of California Riverside, Princeton, UCLA, and Caltech. According to the present deformation professionnelle my work is in M&E (metaphysics and epistemology). My current research concerns color, consciousness, and metaethics.

You can reach Haris at cskiadas@gmail.com, Thomas at thomas.kjosmoen@uis.no and Mark at eli@markelikalderon.com

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

Subversion and TextMate: Making collaboration easier for LaTeX users*

Charilaos Skiadas, Thomas Kjosmoen, and Mark Eli Kalderon

Abstract This article focuses on the Subversion version control system, and in particular its integration into the TextMate text editor. TextMate is a text editor, for the Mac OS X operating system, that has excellent support for working with LaTeX documents and projects, and its seamless Subversion integration makes it very easy to add version control to your workflow.

Version control systems have been used by computer programmers for many years for work on projects involving multiple authors. They have been invaluable in keeping track of the contributions of each party to the project, the changes that took place and when they happened, and so on. Subversion is one of the most popular version control systems today, and it is integrated into many text editors and online collaboration sites.

1 Introduction

We often have to work on LaTeX documents with other people. There are a number of problems with such a collaboration:

- We need to exchange files with our collaborators and keep track of the date of each file, to avoid working on old files.
- We need to communicate with our collaborators what changes we have made to the current draft.
- We would like to keep a track record of all changes made to the document, and which person made each change, for future reference.
- We need some system to keep backups of the files, to avoid data loss.
- We may need to work on the same files on multiple computers, for instance at home and at work.

*Thanks to Francisco Reinaldo for his help and encouragement.

Most of us deal with these problems in awkward ways. E-mailing is probably the most usual way of exchanging versions of our files. Since each file has an associated *modification date*, we also have an elementary way of keeping track of when a file was last changed. We also often tend to name the files according to the date on which they were last modified. To communicate the changes made, we might write comments on the margins via `\marginpar`, or just comments in the LaTeX text, or simply a descriptive e-mail.

For many years programmers have battled with these problems, and a number of excellent solutions have emerged; the so-called *Revision Control Systems* or *Version Control Systems*. *Subversion*[1] is a popular and freely available version control system. The article by Mark Eli Kalderon in the current issue [2] offers a comprehensive introduction to the Subversion system and its benefits when working on LaTeX projects. TextMate as a LaTeX editor has also been discussed by the first two authors in another article in the current issue [3]. In this article, however, we will focus on TextMate's Subversion capabilities, and also discuss more advanced Subversion concepts such as branches and tags, and the process of merging.

In Section 2 we review the basic concepts of Subversion, discuss the problem of simultaneous commits and how Subversion addresses this, introduce the extremely useful concepts of branches and tags, and discuss the `svn merge` process. In section 3 we talk about TextMate's Subversion menu and the various commands in it. Section 4 discusses another very useful feature of TextMate, the TODO bundle, which allows you to keep track of tasks in a simple and convenient way.

2 Revision Control and Subversion

Version control systems are divided in two categories, depending on their decision of where to keep all the data. *Centralized* version control systems follow a client-server model, where the history of the project is stored in some central server, known as the *repository*. Changes made by an author must be committed to this central repository, and most of the operations asking for information about the history of the project need to access this server. Subversion is the primary example of free centralized version control systems, and it has to a large extent

1. http://subversion.tigris.org

supplanted the more traditional CVS[2]. This paper will focus on a discussion of Subversion, though most of the general concepts apply to other centralized version control systems as well.

Distributed version control systems on the other hand are based on the idea that each author has a local copy of the complete history of the project. The author can then go on to work on the project, and at some point his changes can be synchronized with the changes from the other authors. One of the main advantages of distributed version control systems is that they allow a lot more offline operations. Examples of distributed version control systems are Git[3], Mercurial[4] and SVK[5].[6]

2.1 The Repository and Working Copies

The central location where data is stored is known as the *Repository*. It is usually in a remote server[7] or in your local machine[8]; the data is stored in a database (*e.g.* Berkeley DB in Subversion) or as files (*e.g.* FSFS in Subversion), and could potentially be encrypted. This can offer a safe backup location, as servers are typically daily backed up. There are several good free services on the Internet offering access to servers that have Subversion set up.[9]

Let us say that the central repository has been set up and has the recommended repository layout (with trunks, branches, and tags—more on these in section 2.3). In order to protect the repository, users are not given direct access to it. Instead, they work with snapshots of the repository, known as *Working Copies*.

2. http://www.nongnu.org/cvs
3. http://git.or.cz
4. http://www.selenic.com/mercurial
5. http://svk.bestpractical.com/view/HomePage
6. For more information about the philosophy of these distributed version control systems, please see http://speirs.org/2007/07/19/a-subversion-user-looks-at-git and http://www.selenic.com/mercurial/wiki/index.cgi/UnderstandingMercurial. An extended list of version control systems can be found here: http://en.wikipedia.org/wiki/List_of_revision_control_software
7. http://www.assembla.com/space/2007-3PTJ_charilaos of this paper, or svn://tug.org/pracjourn/trunk of the PracTeX Journal.
8. file:///Users/Thomas/SVNrepository/Documents/trunk
9. Please see http://www.subversionary.org/hosting/hosting-services for a list of such services.

The working copy is a directory of files that the author gets from the repository for the first time in a process known as a *Check Out*. This is done *once*.

After the working copy has been checked out, the work process typically consists of the following steps:

- *Update* the working copy from the repository. This gets the newest version of the files from the repository, which includes the changes made by your coauthors.
- *Change* the working copy—either by editing individual files or by changing the directory structure by adding, deleting, or renaming files and directories.
- *Review* the changes made to the working copy.
- *Commit* the changes you made to the working copy. This sends the changes you made, usually along with a comment on the changes, into the repository, so that your coauthors can access them.
- If in the meantime another author has checked in their changes, then you have to update your copy and *merge* your changes with the other author's after *resolving* any conflicts (unless Subversion was able to resolve the conflicts itself).

2.2 The Problem with Simultaneous Commits

Collaboration is based on multiple people working on the same files, and this can cause some problems depending on the order in which these people read to and write from the files. Suppose, for example, that you started to work on a file from the repository, and made some changes to it. You then attempted to commit those changes to the repository. In the meantime, one of your coauthors made some changes to the same file, and committed his changes before yours. The version of the file that you were working on knew nothing of these changes, so when you commit your version of the file, with your changes in it, then his changes will be lost for ever.

There are more than one way to solve this problem. A method used by many systems is a "locking" mechanism. Using this mechanism, you *lock* the files you are going to change before making the changes. You then check the files out,

make the necessary changes, and subsequently commit the changes you made to the repository. After you are done, you unlock the file. The problem with this system is that nobody else can work on the file while it is locked. For a LaTeX document this might work reasonably well, but it is still a significant drawback.

Subversion uses a different method, which involves *merging* of files. The idea is quite simple. Authors check files out as usual, make the changes they want, and try to commit their changes. If, in the mean time, another author has made some changes, Subversion will check to see if those two sets of changes conflict with each other, namely whether both authors have made changes to the same line. If the changes do not conflict with each other, Subversion will *merge* the two sets of changes and update the file accordingly. Otherwise, it will inform the committing author of the problem and allow them to look at the conflicts and decide how best to fix them. Sections 2.4 and 3.4 discusses these issues more.

2.3 Revisions and Branches

Each time you commit something to the repository, the resulting state of the repository is called a *revision*. For instance the first commit made to the repository is revision 1, the second commit becomes revision 2 and so on. These revisions contain the entire history of your project. They each can contain a *commit message*, a useful and brief description of what was committed, or any other comments the author wants to make regarding the commit. The commit messages allow the other authors to get a quick summary of the various stages in the project. For this reason, it is in general advisable that each commit be "atomic", namely that it deals with only one change to the document. For instance, once you finish revising a particular section, you can commit those changes. This allows for more information to be stored in the revision.[10] Since revisions are "cheap", there is no reason not to have many of them. A classical motto is: "Commit early, commit often."

All these revisions allow the authors a lot of flexibility. For instance, you can ask Subversion to show you all the differences between two revisions: Which files were changed, and what changes were made to them. You can also ask it to show you the state of the repository at any revision you want.

10. As an example, this paper is in a Subversion repository, and has seen over 130 revisions in 4 days.

While revisions have a sequential structure, Subversion offers another way to proceed with the project, using what is known as *branches*. A branch, as its name indicates, is a new direction in the repository; very useful for doing major changes to files (Figure 1). A branch, then, is a directory of the repository representing

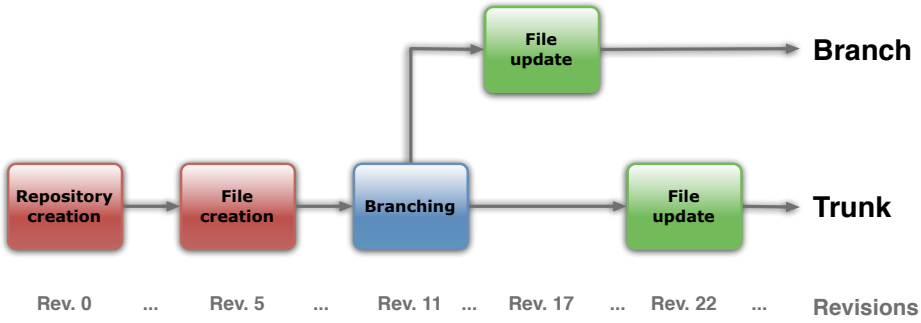


Figure 1: A branch is a new direction of development, moving parallel to the trunk.

a line of development of your project independent of its main or current line of development. In contrast, the main or current line of development resides in a distinct directory called the *trunk*. To create a branch of your document simply copy it into a branch directory and commence with its alternative line of development. If, at some point, you are satisfied that the branch supersedes the trunk as it stands, the changes to the branch can be merged into the trunk.

Suppose, for example, that you are working on some part of the document, on which you are planning to do major changes, which will take many steps and quite a lot of time to complete. At the same time though, your coauthors will need to work on some other parts of the same document, and your work really should be kept separate from their changes. The best solution then is to create a new branch of the repository, and make your changes in that branch, probably in many revisions. In the meantime, your coauthors make their changes into the trunk. Once you are finished with your changes, you can tell Subversion to *merge* the branch into the trunk (Figure 2).

However a branch doesn't have to end when its changes are merged into the trunk. It can continue to exist parallel to the main trunk, with merges happening

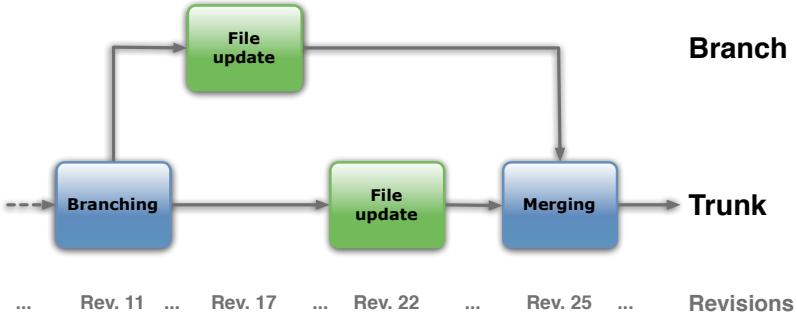


Figure 2: An author may create a temporary branch, to work on a file without interrupting the work of other authors on that file.

either from the trunk to the branch or the other way around (Figure 3). Imagine, for example, working on a textbook. Then the main trunk could correspond to the student's edition, and a branch could correspond to the teacher's edition.

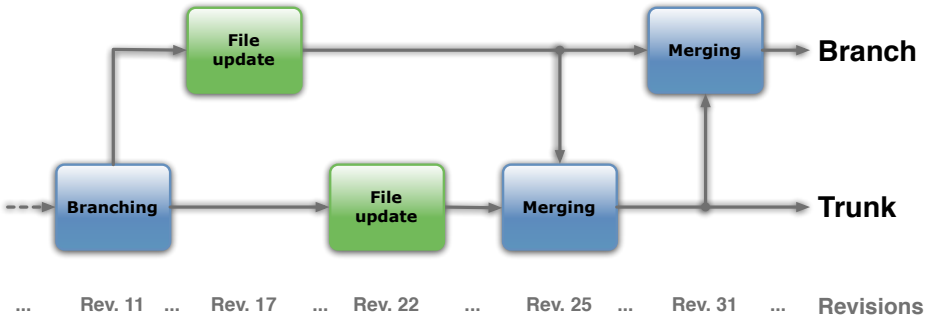


Figure 3: A branch can continue parallel to the trunk, and merging between the two can occur many times.

Another concept, possibly not of much use to LaTeX users but quite useful in, for instance, manuals or other business documents, is the concept of *tags*. A tag is simply a snapshot of the repository at a particular time, and it is meant to not be edited again. So it is just like a revision, except that it is *tagged* with a more meaningful name, and it is thought of as a "finished product". Tags can be used for instance for the various *versions* of a manual. If we were talking about a software, a tag would correspond to a stable release.

2.4 Svn Merge and Svnmerge

So far we have seen merging used to resolve conflicts and in branch management. In the case of conflicts, merging needs to be done by hand, since it requires the application of human judgement and consensus with your coauthors and neither human judgment nor consensus can be automated.[11]. But in the case of branch management, Subversion provides a useful tool, the `svn merge` command. Let's take a closer look at the `svn merge` command and its limitations. Unfortunately, for LaTeX users, these limitations are severe. Fortunately, these limitations can be overcome, in part, by the script `svnmerge.py` that can be integrated with TextMate's Subversion bundle. Moreover, these limitations of merge tracking are being addressed in the next major release of Subversion in a similar manner to `svnmerge.py`.

The `svn merge` command does two things, it compares the differences between two files or directories of files and applies these differences to the working copy. `svn merge` thus takes three arguments. The first two arguments can be repository urls or working copy paths (the locations of the two files or directories to be merged) and the third argument is a working copy path (the location where these differences are to be applied).

Let's consider a simple example to illustrate this. Suppose you are maintaining a list, say, of invitations, that you decide to branch. Perhaps you are considering inviting some more people, but you are unsure whether to invite them or not. Suppose your initial list of invitations, recorded in `trunk/list.txt`, is as follows:

```
Bill
Peter
Mary
```

And suppose your branched list, recorded in `branch/list.txt`, also includes Sally and John :

```
Bill
Peter
Mary
Sally
John
```

11. See the Subversion manual [1] for detailed instructions about resolving conflicts by hand

Suppose you decide that it is OK after all to invite Sally and John (they have been arguing with Bill, but they have made up). You now want to merge the changes you made in the branch into your trunk. This can be done as follows:

```
$ svn merge http://myrepository.com/svn/mylist/trunk/list.txt
http://myrepository.com/svn/mylist/branch/list.txt
mylist/trunk/list.txt
```

The file in your working copy, `mylist/trunk/list.txt`, now contains Sally and John as well. To propagate these changes to your repository, be sure to commit these changes.

Sounds easy. There are, however, limitations to the `svn merge` command that can create complications. The problem arises when you repeatedly merge changes. It can happen that you merge the same change twice over. Sometimes this is OK. If Subversion notices the file has the relevant change already it does nothing. But suppose the change has been made *and has been further modified*. We now have a conflict. Consider the list example again. Suppose John prefers to go by "Jonathan" and this change is made to `mylist/trunk/list.txt` and not to `mylist/branch/list.txt`. The branched list now has names of additional people that should definitely be invited. But it also has John, which svn does not identify as the same as Jonathan. Hence, running the above `svn merge` command again will produce a conflict. The problem arises because Subversion doesn't remember what changes have already been merged into the trunk and so attempts to merge these again.

Another limitation is the inability to exclude a previous revisions from being merged. Suppose you are writing a paper and have been invited to give a talk based on that material. It is natural to create a branch. It is also natural to eliminate some things from the talk like footnotes. You will do this with some commit, that corresponds, say, to revision 20. But suppose in writing the talk you hit upon things that should be in the paper. These are saved in some other revision, say, revision 21. You would now want to merge these changes into the paper, which is in the trunk. However, you cannot tell `svn merge` to merge only those changes, and to not merge the changes you made at revision 20. So with `svn merge` you would end up losing the footnotes in the original paper.

A final limitation is that the changes made when a merge occurs are attributed to the person that did the merge. So, for example, if there were more than one

person working on the branch, and one of those two ended up merging the branch to the trunk, then all the resulting changes are attributed to that person, even though some of them were made by someone else.

So to summarize, some of the main limitations of the `svn merge` command are:

- Subversion doesn't remember merged changes.
- Subversion can't exclude a change set from being merged.
- When a merge occurs, Subversion attributes all changes to the branch to the person merging.[12]

`svnmerge.py`[13] is a tool for automatic branch management that addresses these problems. It remembers which changes have already been merged and does the right thing by only merging the new changes. It lists changes available for merging so that some, but not all, of these can be merged. When merging, it includes the log of all the changes in the branch in the commit message so that changes are appropriately attributed.

To initialize the merge tracking support run the following command:

```
$ svnmerge.py init path/to/my/branch/
```

This only needs to be done *once* for each branch that you are managing. The command:

```
$ svnmerge.py avail
```

lists the revisions that are available for merging. And the command:

```
$ svnmerge.py merge
```

can be used to merge some, but not all, of the available revisions. One note of caution: Be sure that your commits are truly "atomic" (see section 2.3), since if a revision contains two changes, only one of which should be merged into the trunk, then this will need to be done by hand. After these changes have been merged, you can then commit them using `svn commit`.

12. For trenchant criticism of these and related limitations see Linus Torvalds' talk to Google about the advantages of Git and the distributed, as opposed to centralized, model of version control http://www.youtube.com/watch?v=4XpnKHJAok8.
13. http://www.orcaware.com/svn/wiki/Svnmerge.py

3 TextMate's Subversion menu

In this section we will discuss TextMate's Subversion menu, that allows you to do most Subversion-related tasks without ever having to leave TextMate. All Subversion commands in TextMate use the same shortcut, so one is typically presented with the options in Figure 4. The numbers next to the first ten commands mean that those commands can be accessed by simply pressing the corresponding number key. We will go in some detail through each command, but if you still have questions, the "Help" command in the menu will likely answer some of them. In particular it will provide details on the various environment variables you can set to customize how Subversion works.

The commands are divided in groups, depending on their functionality. The first group has to do with making and receiving changes in the repository. The second group has to do with reviewing information about the repository. A third group has to do with looking at the changes between revisions, and a fourth group consists of commands to deal with conflicts.

3.1 Working with Files

The first set of commands is all about working with files. The first command allows you to add files to the repository, and it essentially calls the `svn add` command for you. It will schedule for addition all the files that are selected in the project drawer. Similarly, the second command will schedule the selected files for removal from the repository. These actions will only take place at the next commit attempt, which we will discuss shortly.

The third option allows you to revert the file to its most recent repository state. What this means is that you will lose any local changes you've made to the file but had not had time to commit yet. Hence, TextMate warns you first before doing it (Figure 5).

The next command in the Subversion menu is used to update the working copy to the newest revision. It will only work on selected files, so you can update only part of the working copy if you prefer, or select the root folder and update the entire directory.

The "Commit" command is perhaps the most important command in the

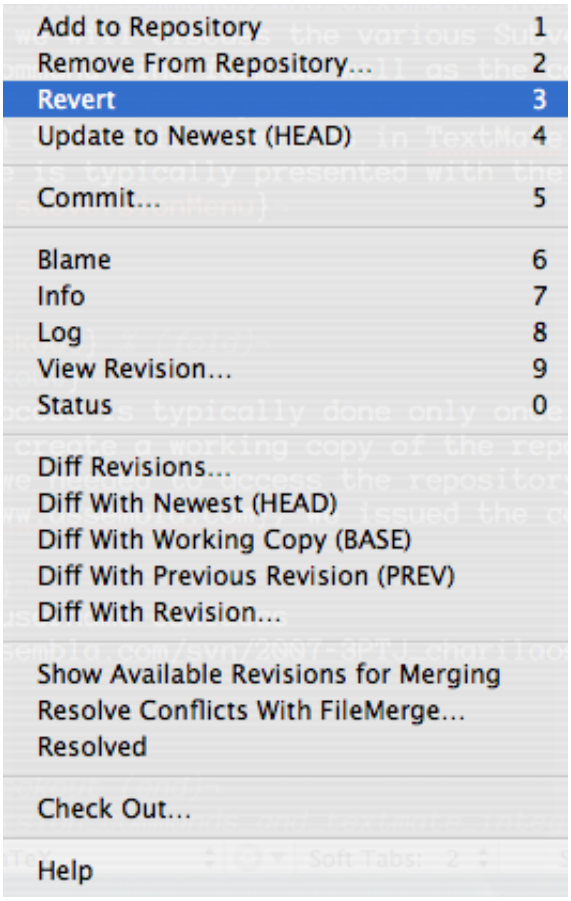


Figure 4: TextMate's Subversion menu offers quick access to all Subversion commands.

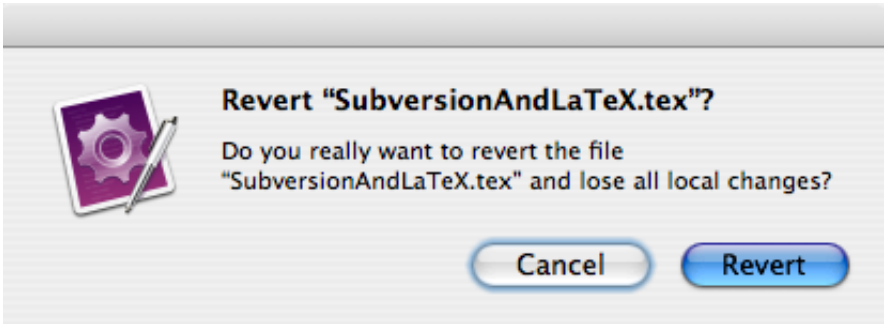


Figure 5: TextMate warns you before doing anything destructive!

menu. It is after all the command you would use to commit your changes to the repository, and thus move forward with the project. It has an elaborate menu, that allows you to review the changes you have made and decide what you want to commit, and what you don't want to commit yet. Figure 6 shows how this window looked like at some point in the progress of this paper.

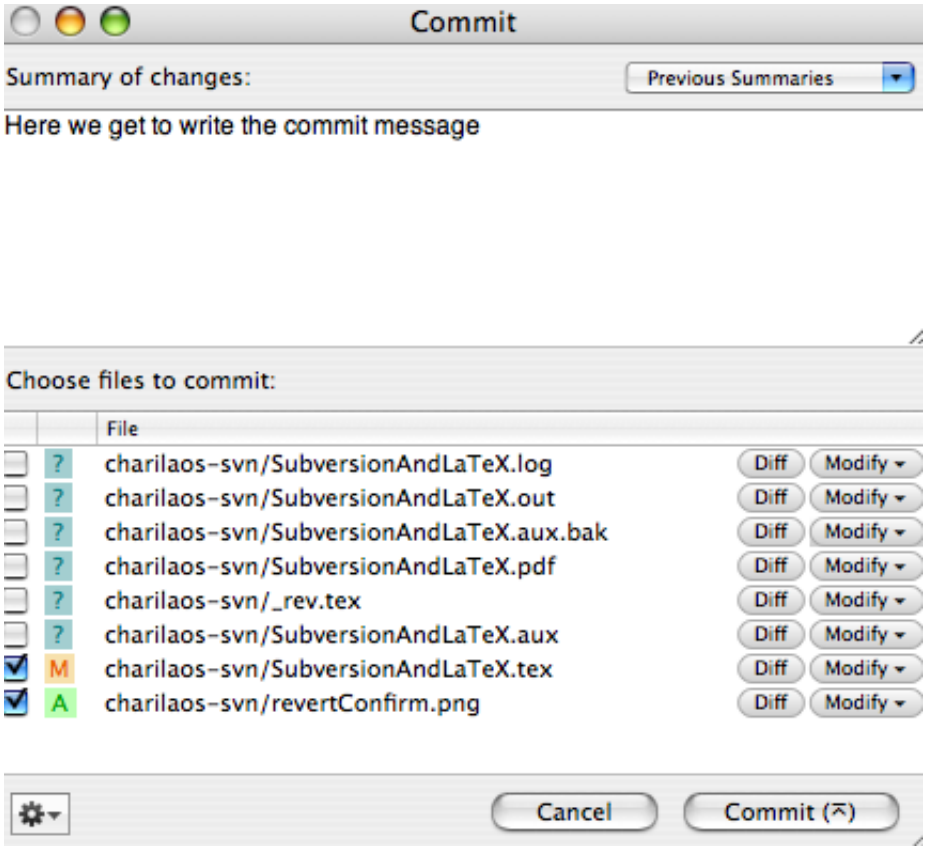


Figure 6: The Commit Window allows you to review your changes, and decide what to commit.

The top of the Commit Window contains an area where you write the commit message. The bottom part of the window is the most interesting part. Here you are provided with a list of all files related to your commit. Each file has a checkbox on the left, where you can decide whether you want this file to be committed or not. Next to the checkboxes, a colored character indicates the state

of the file. A question mark indicates that this file only exists in the working copy, and the repository knows nothing about it. In our case, these are files generated from the LaTeX file, hence we don't want them in the repository. The "M" next to the file `charilaos-svn/SubversionAndLaTeX.tex` indicates that this file has local changes, and the checked box indicates that we are about to commit our changes. Finally, the "A" next to the file `charilaos-svn/revertConfirm.png` indicates that this file is not yet in the repository, but we have scheduled for it to be added in the next commit.

Buttons on the right side of the window allow us to view changes, and to modify the state of the file. The "Diff" button is useful mainly for the modified files, and provides a screen with the differences between our copy of the file, and the copy in the repository that we are about to replace (Figure 7).[14] The "Modify" button provides you with options depending on the status of the file. For files unknown to the repository (those marked with a "?") it offers the option of adding them, for those that are modified the option to revert the changes, and so on.

3.2 Getting Information

The next set of commands in the Subversion menu deal with obtaining information from the repository, regarding the status of files, their history, etc. The first of these is the "Blame" command, which provides a view of the current document, with each single line marked according to when that line was last modified in the file, by whom and on which revision (Figure 8).

The next command is the "Info" command, which is just a wrapper for the `svn info` command. More useful information is obtained from the "Log" command, which shows a list of all the revisions where the current file was affected, the commit messages on each revision, and lots of other useful information (Figure 9).

The "View Revision..." command allows you to select one of the previous revisions of the current file, and have that version of the file open in a new window (Figure 10). Finally, the "Status" command shows a window with options

14. Diff files will be discussed more in section 3.3.

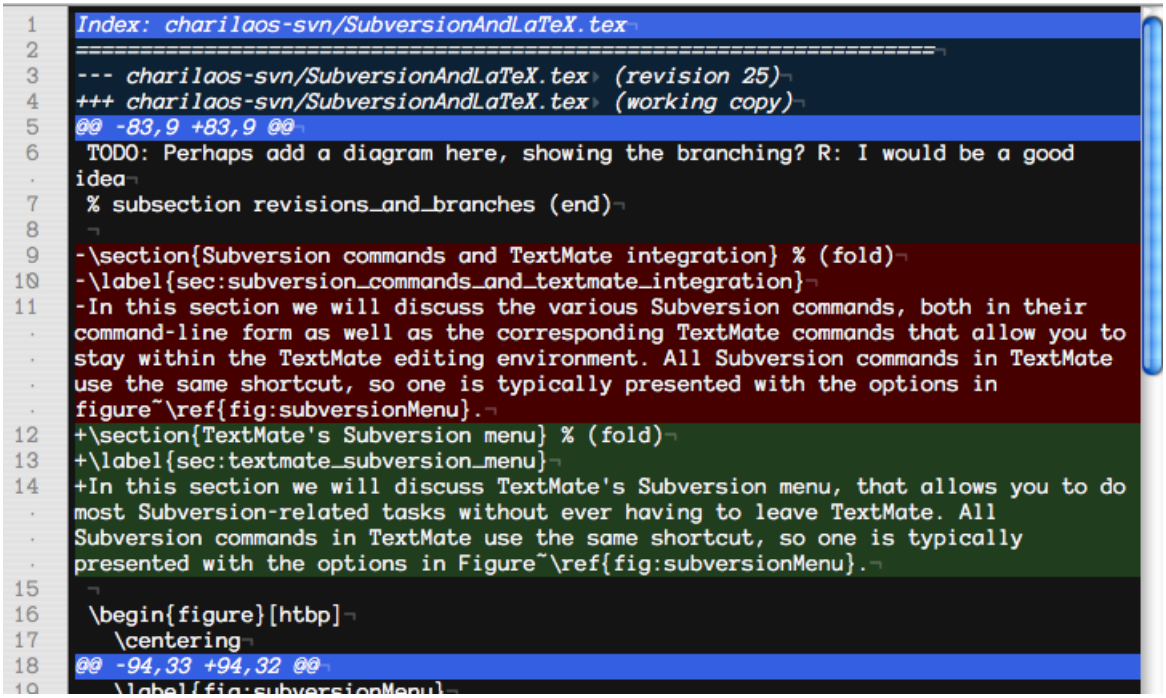


Figure 7: Subversion allows you to see the differences between the working copy of a file and the copy in the repository.

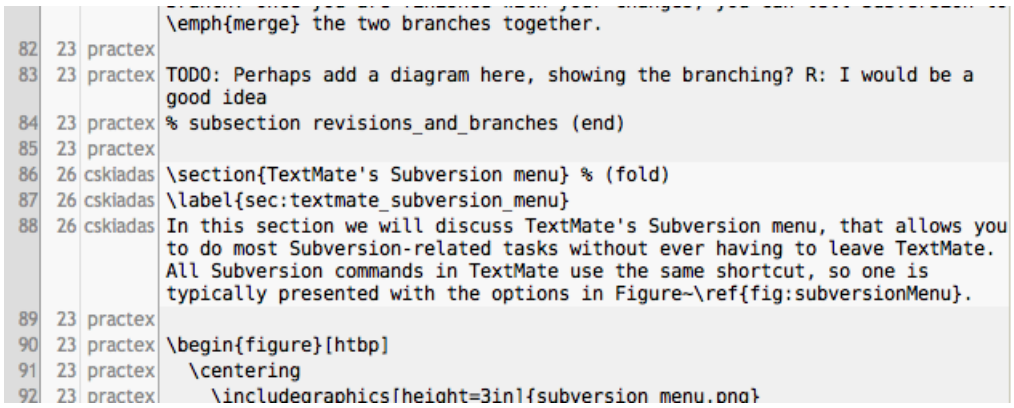


Figure 8: A portion of the Blame Window. Some lines were added by the author named `practex` in revision number 23, while others were added by author `cskiadas` in revision number 26. Hovering over a revision number makes the corresponding commit message for that revision appear.

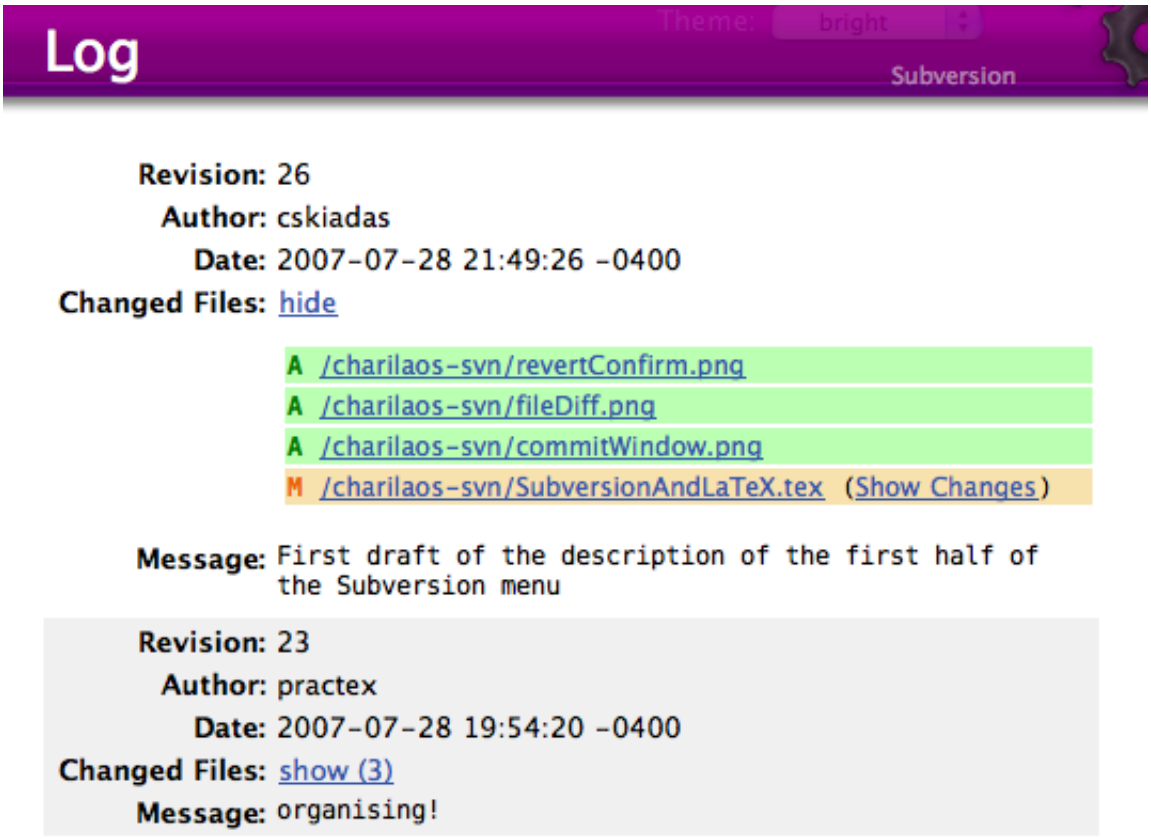


Figure 9: The Log Window gives us information on the history of the current file.

very similar to the ones shown in the Commit window. There are options for seeing the changes you would commit, options for adding files to the repository or removing files from it, and even an option to commit your changes.

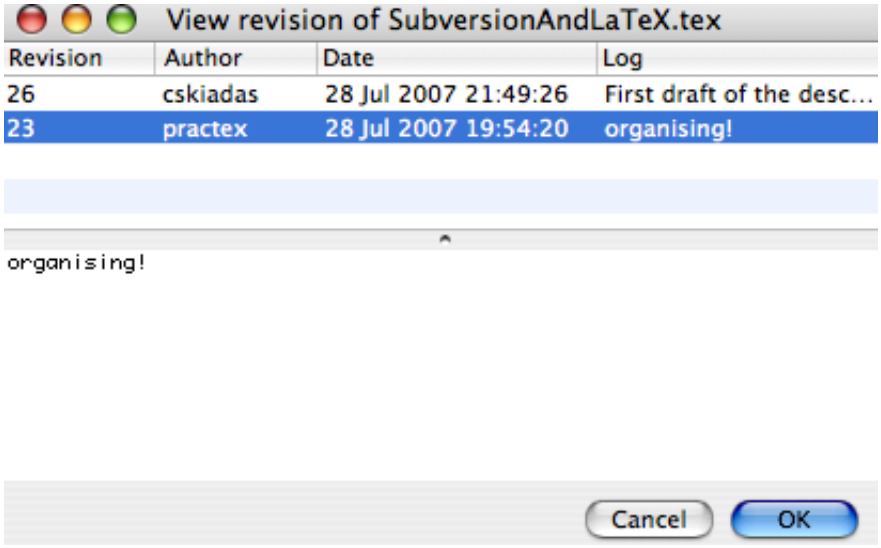


Figure 10: The View Revision window shows all previous revisions of a file, and allows you to open any one of those revisions in a new window.

3.3 Looking at Changes

The next set of commands in the Subversion menu are the five commands for producing various kinds of diff files. *diff* files are files that have a certain format, and their purpose is to show the differences between two other files. We already saw such a diff file in Figure 7. TextMate recognizes the particular format of diff files, and produces a colored output showing the differences between the two files. Unfortunately for LaTeX users, diff files look at *whole lines*, which means that if you compare two files that differ in a single character, then the diff file will indicate the entire line where that character is as the difference. In a LaTeX document, that line could happen to be a whole paragraph. Alternatives to this standard diff mechanism are discussed in [2], and TextMate can easily be made

to use those alternatives. In this section, we will talk about the diff commands assuming that the standard diff mechanism is used.[15]

All diff commands do essentially the same thing: They show you the differences in the selected files between two revisions. The commands differ as to whether and how those revisions are specified. They all produce a new diff file that contains the differences between the two revisions.

The "Diff Revisions..." command allows you to specify any two revisions from a simple and very informative list (Figure 11). The "Diff with Newest" command is perhaps the most often used of the diff commands. It shows the differences between your current version of file and the newest version of the file in the repository. This allows you to see the changes that you would make to the repository version if you were to commit the file right now.

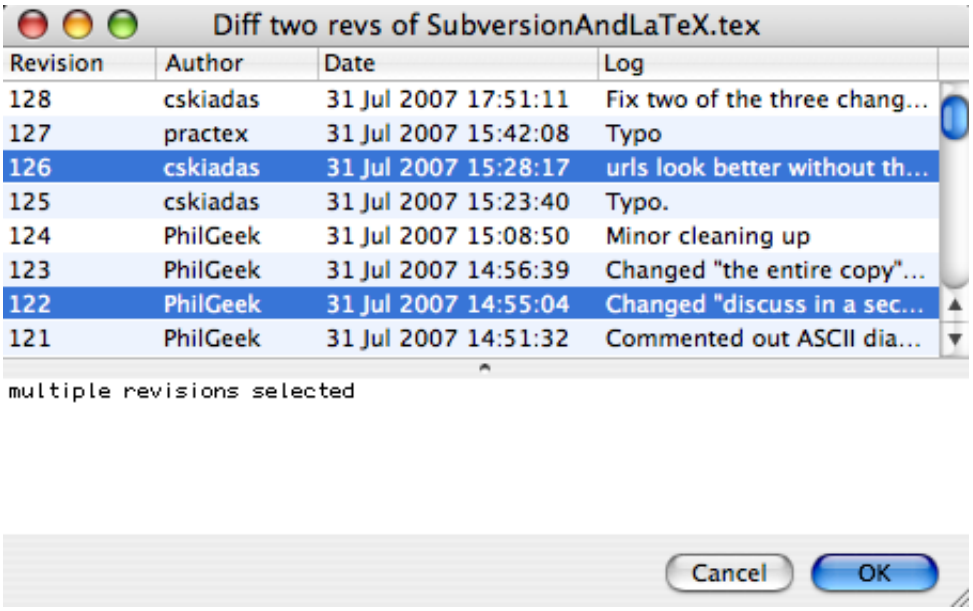


Figure 11: The Diff Revisions command allows you to select the two revisions whose differences you want to look at.

The next command is the "Diff with Working Copy" command. This shows you the differences between your version of the file and the version in the latest

15. What the commands do is essentially the same regardless of what diff mechanism is used.

revision that you have in your working copy (which might not be the latest revision in the repository if you haven't updated recently). This is particularly useful if you have to often work offline, since it doesn't need access to the repository. "Diff with Previous Revision" is helpful when you have just used `svn up` to update your working copy, and want to see the latest changes made to the file. It will show you a diff file between the two newest revisions. Finally, the "Diff with Revision..." command shows you the diff file between your current version of the file and the version in one particular revision.

The standard diff mechanism is useful, but it has its limitations. As we mentioned, one big limitation for LaTeX users is that it only displays *line* differences. The problem is that paragraphs are long lines and so the standard diff mechanism won't discriminate multiple differences within a line. What would be more useful is a diff mechanism that displayed *word* differences. Another limitation of the standard diff mechanism is that it represents differences with textual output in diff format. It would be useful for these to be visually displayed. As we have seen, TextMate addresses this by helpfully providing syntax coloring for the diff output. Another alternative is to use a diff mechanism with a GUI interface.[16] One such alternative that addresses both these limitations is available on Mac OS X. If you install the developer tools that comes with Mac OS X[17], one useful tool that you will have installed is FileMerge. FileMerge is a GUI diff program that visually displays differences between files. And while FileMerge displays line differences, it also highlights word differences. To use FileMerge as your diff mechanism from within TextMate, you need to download and install the script, `fmdiff`.[18] Then, in TextMate's preferences, under Advanced, Shell Variables assign `fmdiff` as the value of the variable `TM_SVN_DIFF_CMD`. Wow. That's a lot of work. But here's the payoff. When you call any of the diff commands discussed in this section, word differences will be visually displayed in a nice GUI. (See Figure 12.)

16. See [2] for further discussion of these limitations and the alternatives.
17. These are on the CD that installs Mac OS X and can be downloaded by members of the Apple Developer Connection (membership is free) at: `http://developer.apple.com/tools/download`.
18. `http://ssel.vub.ac.be/ssel/internal:fmdiff`

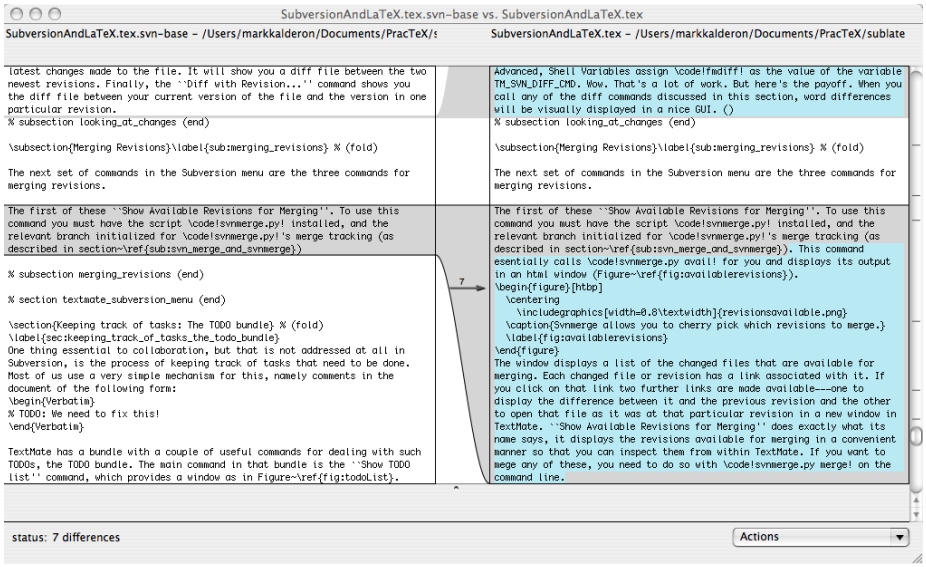


Figure 12: FileMerge graphically represents word and line differences.

3.4 Merging Revisions

The next set of commands in the Subversion menu are the three commands for merging revisions.

The first of these is "Show Available Revisions for Merging". To use this command you must have the script `svnmerge.py` installed, and the relevant branch initialized for `svnmerge.py`'s merge tracking (as described in section 2.4). This command essentially calls `svnmerge.py avail` for you and displays its output in an html window (Figure 13). The window displays a list of the changed files that are available for merging. Each changed file or revision has a link associated with it. If you click on that link two further links are made available—one to display the differences between it and the previous revision and the other to open that file as it was at that particular revision in a new window in TextMate. Note that "Show Available Revisions for Merging" does exactly what its name says, it displays the revisions available for merging in a convenient manner so that you can inspect them from within TextMate. If you want to merge any of these, you need to do so with `svnmerge.py merge` on the command line.

The next command is "Resolve Conflicts with FileMerge". To use this com-

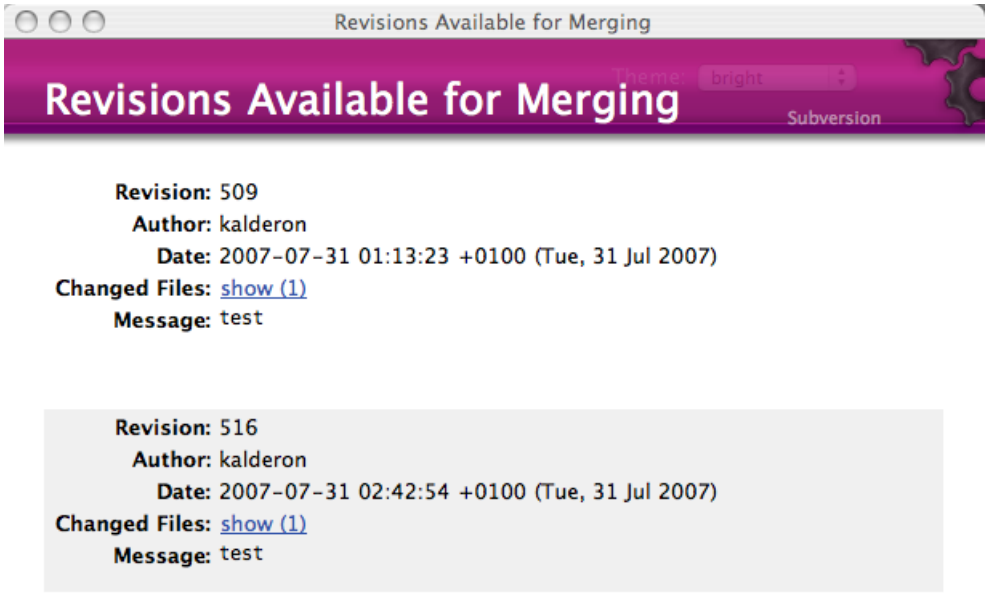


Figure 13: Svnmerge allows you to cherry pick which revisions to merge.

mand you must install Apple's developer's tools and the script `fmdiff` (as described at the end of section 3.3). As we noted in section 2.4, merging changes between a file in a repository that conflicts with the version of that file in your working copy must be done by hand. It cannot be done automatically since it requires the application of human judgement and, in a collaborative setting, consensus among coauthors.[19] In manually resolving conflicts, it would be useful for the conflicting changes to be explicitly displayed. "Resolve Conflicts with FileMerge" visually displays these conflicting changes in FileMerge's GUI. (See Figure 12.)

The next command is "Resolved". Manually resolving conflicts involves making the relevant changes to the file in your working copy. Suppose you have made these changes and that you and your coauthors are satisfied with them. Before you can commit these changes, you must tell Subversion that the conflict has been resolved. On the command line, this is done with the command:

```
$ svn resolved mymainfile.tex
```

19. See the subversion manual [1] for detailed instructions about resolving conflicts by hand

where `mymainfile.tex` is the file in your working copy that is in a conflict state. This command tells Subversion that the conflict has been resolved and that it is OK to commit these changes to the repository. "Resolved" is essentially a wrapper for `svn resolved`. Having made the relevant changes to `mymainfile.tex` within TextMate, there's no need to go to the command line—you can tell Subversion that the conflict has been resolved within TextMate with this command.

4 Keeping track of tasks: The TODO bundle

One thing essential to collaboration, but that is not addressed at all in Subversion, is the process of keeping track of tasks that need to be done. Most of us use a very simple mechanism for this, namely comments in the document of the following form:

```
% TODO: We need to fix this!
```

TextMate has a bundle with a couple of useful commands for dealing with such TODOs, the TODO bundle. The main command in that bundle is the "Show TODO list" command, which provides a window as in Figure 14. A list of all the TODOs shows up, along with links to the location in the document where they appear. The command will actually look for TODOs in the entire project, not just the current file. The command actually recognizes a variety of tags like FIXME and CHANGED, and you can add your own tags through the bundle preferences. In Figure 14, we have added a new tag for each of the authors, and those tags look for items like the following:

```
% TODO(cskiadas): This is assigned to Charilaos
% TODO(kjosmoen): This is assigned to Thomas
% TODO(markkalderon): This is assigned to Mark
```

Used in conjunction with Subversion, the TODO bundle, in effect, constitutes an efficient ticket system.

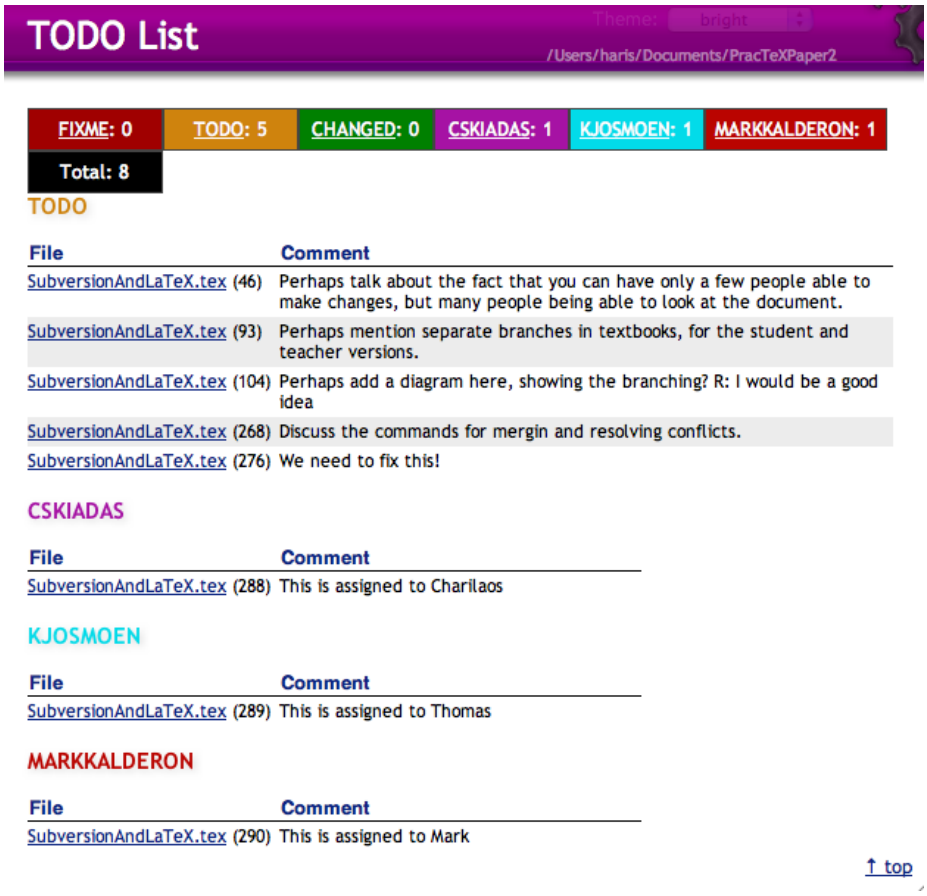


Figure 14: The TODO list window: A simple and efficient way to keep track of tasks.

5 Concluding Remarks

LaTeX is great for the construction of complex documents. Some form of version control is really indispensable for managing the development of your complex LaTeX document. This need will be especially apparent when that document is the collaborative effort of coauthors. Subversion, a free and open source version control system, can help meet this need. Keeping your LaTeX documents in version control with Subversion is made easier if your LaTeX-aware editor has Subversion integration. We have discussed the Subversion integration of one popular LaTeX-aware editor on the Mac OS X platform, TextMate. Other LaTeX-aware editors on Mac OS X with subversion integration include BBEdit[20], Aquamacs[21] and Carbon Emacs[22], to name a few. For more information about Subversion, consult the book, *Version Control with Subversion* [1]. This book is available free online in both pdf and html format.[23] Besides being free, it is a well written and user-friendly introduction to Subversion. Consult it, and consider using Subversion for your next collaborative LaTeX project. And for a really enjoyable collaborative LaTeX experience, try using TextMate.

References

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

[2] Mark Eli Kalderon. LaTeX and Subversion. *The PracTeX Journal*, (3), 2007.

[3] Charilaos Skiadas and Thomas Kjosmoen. LaTeXing with TextMate. *The PracTeX Journal*, (3), 2007.

20. http://www.barebones.com/products/bbedit
21. http://aquamacs.org
22. http://homepage.mac.com/zenitani/emacs-e.html
23. http://svnbook.red-bean.com

THE PracTEX Journal

LaTeX Document Management with Subversion

Uwe Ziegenhagen

Abstract

In this article I will describe the Subversion setup on Windows and Linux systems, the elementary steps of document management and various LaTeX packages working hand in hand with Subversion.

I am research assistant in the stats department at Humboldt-University Berlin. For about eight years I have been working with LaTeX and its friends.

You can reach Uwe Ziegenhagen at ziegenhagen@wiwi.hu-berlin.de

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/17

LaTeX Document Management with Subversion

Uwe Ziegenhagen

Email latex@ziegenhagen.info
Website http://www.uweziegenhagen.de
Address Humboldt-Universitt zu Berlin, Germany
Center for Applied Statistics and Economics

Abstract From the single-author composition of a Bachelor thesis to the creation of a book by a team there are many occasions, where version management of a document may be helpful. With the aim of overcoming the shortcomings of CVS (Concurrent Version System) the Subversion version control system was implemented.

In this article I will describe the Subversion setup on Windows and Linux systems, the elementary steps of document management and various LaTeX packages working hand in hand with Subversion.

1 CVS versus Subversion

Contrary to CVS the versioning scheme of Subversion does not refer to single files anymore but to a whole tree of files. Each revision number n refers to the state of the repository after the n-th commit. When we speak about a file in revision 4 we mean the file in the state of revision 4.

The revision numbers of a single file may even have gaps if it had not been changed on every commit to the repository. Table 1 illustrates an example: Up to revision 4 all files have been changed before each commit, so the revision of the repository and the revision numbers of the files are equal. Before the commit to revision 5 only chapter1.tex is modified however the whole repository receives the revision number 5, before the commit to version 6 all files were modified again and therefore have the revision number 6.

On each checkout from a Subversion repository the highest revision number of each file will be checked out which is smaller or equal to the desired revision

Revision 4	Revision 5	Revision 6
thesis.tex:4	thesis.tex:4	thesis.tex:6
preamble.tex:4	preamble.tex4	preamble.tex:6
chapter1.tex:4	chapter1.tex:5	chapter1.tex:6

Table 1: Gaps in Subversion revisions

number. Subversion stores a second copy of each file in a special directory (`.svn`) on each checkout, update and commit.

Although this doubles the required space on the hard-disk it has certain advantages, especially when dealing with remote repositories: Viewing local changes can be made without access to the network, when committing a file, Subversion has only to send the changed parts whereas, CVS calculates the changes on the server and has to send the whole file on each commit. Commits are atomic, which means a change to a file is either completely stored or not stored at all. Thus network issues or concurrent commits cannot lead to inconsistent status.

2 Installation

There are different options for the installation of Subversion. One can either use svnserve [3] or install Subversion as an Apache 2 module, which uses WebDAV[1].

In this article I will use the latter option by installing Subversion as an Apache2 module, since the integration into Apache 2 provides a few interesting features such as the possibility of surfing through repositories in a web browser and the use of Apaches authentification mechanisms.

1. *Web-based Distributed Authoring and Versioning*, a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers.

2.1 Windows XP

2.1.1 Apache Setup

Binary versions of Apache 2 are available from [1], however I usually prefer to use a WAMP[2]-solution provided by apachefriends.org. We extract the xampp.zip[3] to e.g. `C:/xampp` and start the Apache server using `xampp-control.exe`. When we open `http://localhost` in a web browser the XAMPP start page should be displayed as shown in Figure 1.

Figure 1: Screenshot of xampp starting page

As we assume that only the local computer should be allowed to access the webserver it is strongly recommended securing against access from outside. For details please see the respective chapter in the Apache documentation [7].

2. Windows-Apache-MySQL-PHP
3. current version at printtime: 1.6.2

2.1.2 Subversion

We download Subversion[4] from [2] and extract all files from the zip archive to e.g. `C:/Program Files/Subversion`. After adding the path to the `C:/Program Files/Subversion/bin` directory to the PATH environment variable from Windows, we can call `svn help` from the commandline to check if our installation is working. In the next step we copy `mod_authz_svn.so` and `mod_dav_svn.so` from the `subversion/bin` directory to the Apache modules directory and overwrite older versions of this file if necessary.

In the final step we enable WebDAV and the Subversion module by adding

- `LoadModule dav_svn_module modules/mod_dav_svn.so` and
- `LoadModule authz_svn_module modules/mod_authz_svn.so`

to the `httpd.conf` in the Apache `/conf` directory. Before we restart Apache to load the modules we need to make further adjustments to this file. We create a root directory for all our repositories (e.g. `c:/allMyRepositories`) and add the code from Listing 1 to `httpd.conf`:

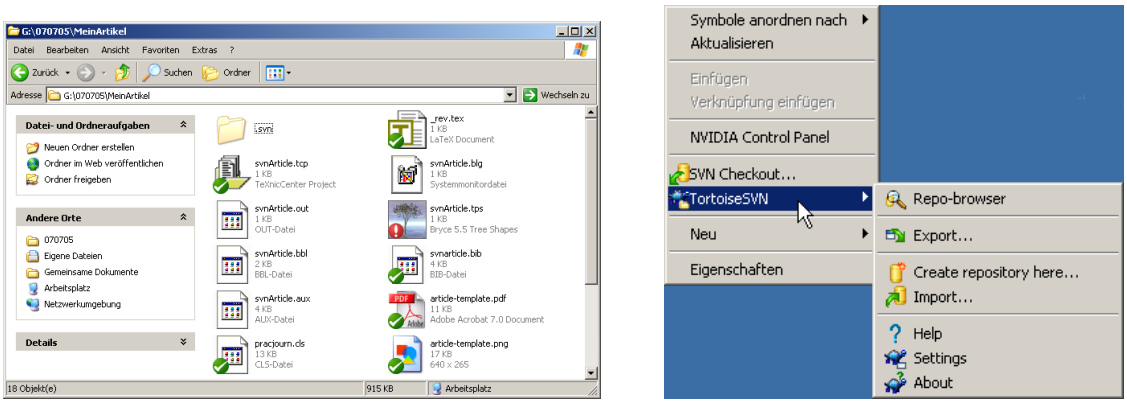
```
1 <Location /svn>
2 DAV svn
3
4 SVNParentPath c:/allMyRepositories
5 </Location>
```

Listing 1: Setup code for the Windows repository root

Using the command line we go to `c:/allMyRepositories` and create our first repository by executing `svnadmin create firstSample`.

If we now open `http://localhost/svn/firstSample/` in a browser we should see an empty directory listing with the headline `Revision 0: /`. The basic installation of Subversion is now done, however we can achieve a much more convenient way of handling repositories by installing TortoiseSVN.

4. current version at printtime: 1.4.4



(a) (b)

Figure 2: Screenshot of a Working Directory with Subversion installed and context menu of Subversion 1.4.4.

2.1.3 TortoiseSVN

TortoiseSVN[5] [4] is a free Subversion client, implemented as a Windows shell extension. It features a multilingual interface with Windows Explorer integration, its icon overlays show immediately which files/folders have been changed and need to be committed to the repository. The installation is straightforward, after rebooting the computer we find various entries in the context menu to manage our repositories. Besides there are more clients available, for example RapidSVN (Windows, Unix/Linux) and SVNcommander (Linux).

2.2 Linux (Ubuntu 7.04)

The installation on a Linux system is much easier than the installation on Windows. Using `apt-get install` or the Synaptic package manager we install the following packages:

- apache2.2-common and apache2-utils
- libapache2-svn
- subversion

5. current version at printtime: 1.4.4.9706

Additional packages are selected automatically by the package management tool. After the installation of these packages the remaining step is to make the necessary adjustments to `/etc/apache2/sites-available/default` and to set the access rights for this directory via `chmod -R 770 /home/uwe/repositoryRoot`:

```
<Location /svn>
DAV svn

SVNParentPath /home/uwe/repositoryRoot
</Location>
```

Listing 2: Setup code for the Linux repository root

3 First steps

To fill the repository we created during the installation we create a empty directory (all files in this directory will be imported to the repository) which we populate with a small LaTeX document (article-template.tex):

```
\documentclass{article}
\begin{document}

Hello World!

\end{document}
```

Listing 3: A simple LaTeX file

Using the the command line (`svn import http://localhost/svn/firstSample/ -m "import"`) or the TortoiseSVN context menu or we can now import the file using our URL for the repository `http://localhost/svn/firstSample/` and use "import" as comment. Apache will list now `Revision 1: /` when we browse the repository (see Figure 3. To work with the file again we need to make a check-out into a `working directory`. The files in the working directory are the files we edit, all future revisions will be committed from this directory.

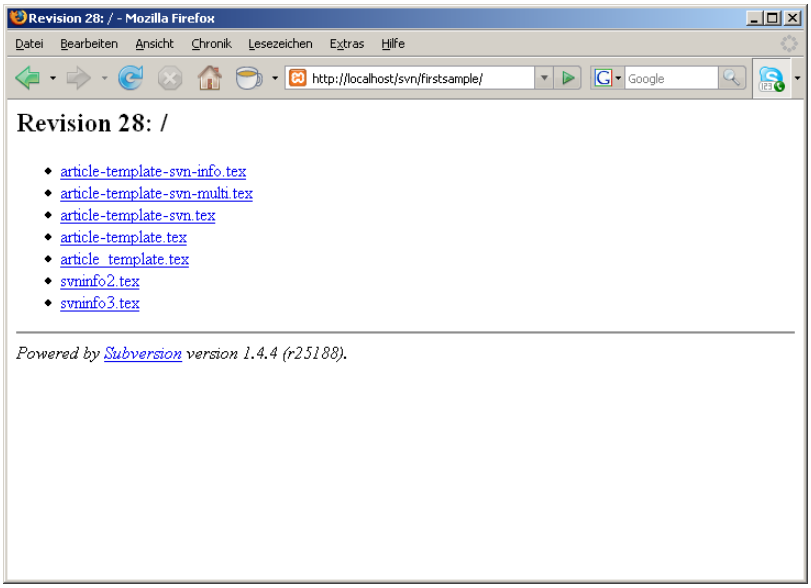


Figure 3: Repository browsing with Apache and Subversion Module

4 Integration with LaTeX

To integrate the Subversion metadata in our LaTeX files we need to tell Subversion to include them. The following list contains the available keywords and their description:

Date (LastChangedAt) date and time of last check-in
Revision: (LastChangedRevision) the number of the revision
Author: (LastChangedBy) name of the submitting author
HeadURL: the URL of this file
Id: a summary of the above keywords

After using `svn propset svn:keywords "Date HeadURL Revision Id" article_template.tex` from the commandline Subversion will expand those keywords (enclosed in $) in our file when we include them in the LaTeX code. Subversion will expand the keywords as following:

July 15, 2007
2007-07-15 17:33:30 +0200 (So, 15 Jul 2007)
17:33:30
article-template.tex 12 2007-07-15 15:33:30Z
http://localhost/svn/firstSample/article-template.tex

Figure 4: Output of article-template.tex with svn package

```
% $Revision$
% $HeadURL$
% $Date$
% $Author$
% $Id$

\documentclass{article}
\begin{document}
 Hello World!
\end{document}
```

Listing 4: A sample file with expanded Subversion keywords

All LaTeX packages introduced below are based on the evaluation of these keywords.

4.1 svn

The svn package allows access the metainformation by evaluating the Subversion information using a \SVN $Keyword: <metadata>$ syntax. If the keywords are correctly expanded, then the svn package defines:

- \SVNDate for the date of the checkin, \SVNTime as the check-in time and \SVNRawDate as raw date and time if Keyword was $Date$.
- \SVNKeyword otherwise (Examples: \SVNId, \SVNHeadURL)

```
\documentclass{article}
\usepackage{svn}

\SVN $Id$
\SVN $Date$
\SVN $Id$
\SVN $HeadURL$

\begin{document}

\SVNDate \\
\SVNRawDate \\
\SVNTime \\
\SVNId \\
\SVNHeadURL
\end{document}
```

Listing 5: A sample file using the `svn` package

4.2 svninfo

The `svninfo` package needs information from the Id keyword only which need to follow the `\svnInfo` command: \svnInfo *Id* : *article* − *template* − *svn* − *info.tex*182007 − 07 − 1516 : 11 : 21*Z*

To use the meta information the package defines the following commands:

- `\svnInfoFile` the name of the file
- `\svnInfoRevision` the revision number
- `\svnInfoDate` the date of the last check-in
- `\svnInfoTime` the time of the last check-in
- `\svnInfoYear` the year of \svnInfoDate
- `\svnInfoMonth` the month of \svnInfoDate
- `\svnInfoDay` the day of \svnInfoDate
- `\svnInfoOwner` the owner of the file (if specified at check-in)
- `\svnToday` date of last check-in in the `\today` format
- `\svnInfoMinRevision` the minimum revision of the document
- `\svnInfoMaxRevision` the maximum revision of the document

\svnInfoMinRevision and \svnInfoMaxRevision are useful for multi-file documents. Furthermore the packages allows a few optional parameters such as fancyhdr,eso-foot, scrpage2 to typeset Subversion information in the margin or the footer of the document. For details please see the manual.

4.3 svn-multi

The svnmulti package provides two commands, \svnid and \svnidlong, to capture the input from Subversion. To use the variables, the package provides the following commands:

- \svnrev the revision
- \svndate the date of the last check-in
- \svnauthor the author
- \svnfilerev the revision of the current file if it contains a \svnid or \svnidlong or the values of the last file if it does not contain one of these commands
- \svnmainurl and \svnmainfilename typeset the URL respectively name of the main file, as it was defined by the internal command \svnmainfile at the end of the preamble

Furthermore svn-multi uses \svn{*keyword*} and \svnkw{*keyword*} to print Subversion keywords directly. To access date information the package provides some more commands, explanations can be seen directly from each respective name: \svnfileyear, \svnfilemonth, \svnfileday, \svnfilehour, \svnfileminute, \svnfilesecond, \svnfiletimezone, \svnyear, \svnmonth, \svnday, \svnhour, \svnminute, \svnsecond and \svntimezone.

5 Conclusion

This article described the basic usage of Subversion from LaTeX and the most common features of the three LaTeX packages svn, svninfo and svn-multi. More information about integration with LaTeX or Subversion itself can be found in the documentation of the packages or in books on Subversion ([6, 8]. Feedback on this article is welcome, if you find any mistakes or have comments please send me an email. Updates of the article can later be found online at http://www.uweziegenhagen.de/latex/

References

[1] Apache 2 web server. URL http://httpd.apache.org.

[2] Subversion. URL http://subversion.tigris.org/.

[3] Svnserve based server. URL http://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-serversetup-svnserve.html.

[4] TortoiseSVN. URL http://tortoisesvn.tigris.org.

[5] apachefriends.org. Xampp. URL http://www.apachefriends.org/en/.

[6] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. M. Pilato. *Version Control with Subversion. Next Generation Open Source Version Control*. O'Reilly, 2004.

[7] Apache Foundation. Apache HTTP server version 2.2 documentation. URL http://httpd.apache.org/docs/2.0/en/.

[8] Mike Mason. *Pragmatic Version Control Using Subversion*. Pragmatic Programmers LLC., 2006.

THE PracTEX Journal

Version Control of LaTeX Documents with svn-multi

Martin Scharrer

Abstract

This paper describes how to use the software Subversion to version control your LATEX files while also placing the current revision information in your document using the package svn-multi (v1.3 or later). It covers all steps needed to setup and use Subversion, and to manage multi-file documents. Usage examples are provided to show the basic and advanced features to allow the reader to get the most out of the package.

Martin Scharrer started to use LaTeX for his final year project for his electronic engineering degree in 2003. After introducing himself to the basic usage of LaTeX and his standard packages he got interested in how it actually works and read books about macro programming, package creation and TeX, i.e. the TeXbook. In 2006 he got confronted with the need of a version control software and decided to go for Subversion. This led very quickly to the idea to write some Subversion LaTeX macros to store and typeset the provided version control keywords. After the basic version was finished the requirement to handle multi-file documents was spotted and so the 'svn-multi' (aka svnkw) package was born.

He is currently studying a PhD in electronic engineering on the University of Limerick, Ireland.

You can reach Mark at martin@scharrer-online.de and know about his LaTeX researches at http://www.scharrer-online.de/latex/.

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

The PracTEX Journal, 2007, No. 3
Article revision 2007/08/07

Version Control of LaTeX Documents with `svn-multi`

Martin Scharrer

Email martin@scharrer-online.de
Website http://www.scharrer-online.de/latex/

Abstract This paper describes how to use the software Subversion to version control your LaTeX files while also placing the current revision information in your document using the package `svn-multi` (v1.3 or later). It covers all steps needed to setup and use Subversion, and to manage multi-file documents. Usage examples are provided to show the basic and advanced features to allow the reader to get the most out of the package.

1 Introduction to Version Control

Most people working with source code, whether it is a software programming language, markup languages like HTML/XML or in our case LaTeX, face the problem on how to organise their code when it spans over an increasing number of files. Backups should be created in order to roll back changes and to rescue lost data. Nontrivial changes should be logged and important revisions should be marked so they can be found at a later stage. Also when multiple people work on the same source code simultaneously their changes must be synchronized to ensure that their changes are not overwritten unintentionally.

This is the point where a *Version Control* software should be used. *Version Control* (VC), which is also called *Revision* or *Source Control*, manages your files, directories, and the changes made to them. For this all created revisions are stored in a repository which can be envisaged as a special kind of database. Every time the source code is revised it should be committed to the repository where it is saved in a compact differential form together with a log message describing the changes. The revision number is then incremented to identify the new revision.

Subversion [1] is a free modern general purpose VC system which was developed as a replacement to the widely used *Concurrent Versions System* (CVS).

Subversion can be easily used to manage LaTeX code of any size and any related files, e.g. image files, configuration files and the outputted PS or PDF files.

The author's LaTeX package `svn-multi` [3] can be used to place automatic revision keywords, like for example, the last changed revision number, author and date, into your LaTeX files and typeset this information anywhere in your document. If the document contains multiple files which are included (using `\include` or `\input`) in a master file then all files can and should contain these keywords. In addition to the keyword values of the current file, the most recent changed file is recognized and its values are also provided as macros using an auxiliary file (.svn extention).

2 Preparing Subversion

After Subversion has been installed, a repository must be created and existing source files must be placed in this repository for version control. To enable keyword expansion in Subversion a so-called "property" must be attached to these files.

2.1 Installation

The installation of subversion is simple, under MS Windows just download the installer (svn-⟨*version*⟩-setup.exe) from `http://subversion.tigris.org/project_packages.html#windows`, run this executable file and follow the installation instructions. Under Linux the normal software management tool of your distribution can be used to install the Subversion package. For example the author installed it on his Gentoo Linux OS using "`emerge -av subversion`".

It is also recommended to install the config file as outlined in Appendix A.

2.2 GUI Front-ends for Subversion

Subversion provides a command line interface which is good but can be difficult to use for some users who are primarily used to working with a graphical user interface under MS Windows.

Such users can use *TortoiseSVN*, which is a graphics front-end for MS Windows and integrates nicely with Windows Explorer, showing the current status of your

files as overlay icons. It can be freely downloaded from http://tortoisesvn.net/downloads and can be installed simply by running the installer executable file.

Under Linux, *KDESVN* (http://www.alwins-world.de/wiki/programs/kdesvn) can be used as a Konqueror plug-in or a stand-alone application for non-KDE users. It can be installed using the same software management tool used for the installation of Subversion.

Also other GUI front-ends exist, some of them can be used on both operating systems, e.g. *RapidSVN*, *SmartSVN*, *Subcommander* and more.

2.3 Creation of a local repository

A local Subversion repository is created in the command line using

`svnadmin create` ⟨*path of new repository*⟩

which works under Linux and Windows (as all Subversion command line tools do). Using TortoiseSVN just open the Windows Explorer, create an empty folder, right-click on it and select 'TortoiseSVN' → 'Create repository here'. Ensure a "FSFS" type repository is created as well.

2.4 Setting up a local working directory

While all revision of your code is saved in the repository the current version under development lies in a "working directory" which contains VC metadata in a hidden subdirectory.

The simplest way to get a working directory is to check-out your fresh repository using

`svn checkout file://`⟨*path of your repository*⟩ ⟨*working dir name*⟩
or using TortoiseSVN:
Right click on an empty directory → 'SVN Checkout ...'

and then copy all your existing document files to the working dir. To enable version control on these files, perfom the following inside the working directory

`svn add *`
or
Mark all files (CTRL-A), right-click them → 'TortoiseSVN' → 'Add' then click the 'Ok' bottom

The Subversion team suggests to create three directories 'trunk', 'tags' and 'branches' off the repository root. All files should then put in this 'trunk' while the other directories are used to tag revisions and to create development branches. For LaTeX sources this doesn't have to be done as long you don't need tags and branches (in an extra directory).

The working directory must to be commited using

`svn commit`
or
Right-click on working directory → 'SVN Commit'

The log message can be "Initial commit" or similar.

2.5 Basic Workcycle

When using Subversion for version control a basic workcycle should be followed to reduce the risk of problems, e.g. with concurrent changes made by other authors. The cycle starts when you need to modify your LaTeX sources. Here it is assumed that a working directory is already present (as described in the last section).

Tables 1 and 2 list all the main Subversion subcommands and their most important options.

2.5.1 Update your working directory

To ensure that you are working with the latest revision of the repository, the working directory should be updated before every change is performed. If this step is omitted collisions between changes from different authors can easily occur. Even single author sources should be updated, especially before manipulating version controlled directories which must be at the latest revision in order to be changed.

The update process is straightforward:

`svn update [⟨file or dir or nothing (current dir)⟩]`
or
Right-click on working directory (or file or subdir) → 'SVN Update'

When updating the root of the working directory all contained files and subdirectories are also updated. Please note that the displayed brackets `[ ]` symbolise a optional argument which, when used, should be written without the brackets.

2.5.2 Make changes, add new or delete files

The files under version control can be changed as normal with our prefered editor or any other software. Please note that the files and directory are linked to the working directories by name, so if you wish to rename or move files you have to do this using the Subversion client tools and not using methods provided by the operating system. One important concept is that all changes described here only get taken over with the next commit and can be reverted to the previous revision at any time.

New created files can be added, i.e. put under version control, using

`svn add ⟨file(s) and/or dir(s)⟩`
or
Right-click on the new file or dir → 'TortoiseSVN' → 'Add' then click the 'Ok' bottom

To remove files from version control, which also deletes the working copy in the working directory, use

`svn del ⟨file(s) and/or dir(s)⟩`
or
Right-click on the file or dir → 'TortoiseSVN' → 'Delete'

The deleted files are no longer part of future revisions but of course still exist in older revisions.

Subversion, unlike CVS, also supports the copying and moving/renaming of files and whole directories under version control. The copies are created very efficiently in the repository by internally linking them to the revision of the original file or dir. This action attaches the revision history of the original file or dir to the new one and allows the later tracking of the relationship between them. Such

copies are very cheap in terms of repository space and are also used to create branches and tags. Copying is done using

> `svn copy` ⟨*existing file/dir*⟩ ⟨*new file/dir*⟩
> or
> Right-click on and hold existing file or dir, drag-n-drop it to the new location, release the right-click and select 'SVN Copy versioned files here' in the appearing menue.

The moving of version controlled items in Subversion is implemented by copying the original file or dir to the new location and then deleting it afterwards. This can performed using

> `svn move` ⟨*existing file/dir*⟩ ⟨*new file/dir*⟩
> or
> Right-click on and hold existing file or dir, drag-n-drop it to the new location, release the right-click and select 'SVN Move versioned files here' in the appearing menue.

2.5.3 Commit our changes to the repository

Finally all local changes made to the working directory are committed to the repository to create a new revision. A suitable log message should be provided to be able to understand and track changes. The commit, also called check-in, is done using

> `svn ci [`⟨*file or dir or nothing (current dir)*⟩`]`
> or
> Right-click on working directory (or file or other dir) → 'SVN Commit'

If any change is undesired, it can be reverted back to the previous revision prior to the commit using

> `svn revert [-R]` ⟨*file or dir or nothing (current dir)*⟩
> or
> Right-click on working directory (or file or other dir) → 'TortoiseSVN' → 'Revert' then click the 'Ok' button

The optional `-R` option used with directories ensures the recursive descent through the subdirectories, reversing changes to all included files.

2.6 Install `svn-multi`

The installation of the `svn-multi` package can be done using the normal LaTeX package installer provided by your distribution (e.g. MikTeX), using the enclosed Makefile or manually as per the following procedure

1. Download the zipped package from CTAN
   http://theory.uwinnipeg.ca/scripts/CTAN/macros/latex/contrib/svn-multi.zip
2. Unpack ZIP file using your standard unzipper.
3. Run file `svn-multi.ins` through LaTeX, i.e.:
   `[pdf]latex svn-multi.ins`
4. The documentation can be produced with:
   ```
   [pdf]latex svn-multi.dtx
   [pdf]latex svn-multi.dtx
   makeindex -s gind.io svn-multi.ind svn-multi.idx
   makeindex -s gglo.-o svn-multi.gls svn-multi.glo
   [pdf]latex svn-mudtx
   ```
5. Copy the files `svn-multi.sty` and `svnkw.sty` to your TEXMF tree, e.g.:
   `cp svn-multi.sty svnkw.sty /usr/share/texmf/tex/latex/svn-multi/`
   or, for current user only
   `cp svn-multi.sty svnkw.sty ~/texmf/tex/latex/svn-multi/`
   or, for Windows
   `copy svn-multi.sty svnkw.sty C:\texmf\tex\latex\svn-multi\`
6. Optionally the documentation can be copied into the directory `/doc/latex/svn-multi/` in your TEXMF tree.

2.7 Subversion properties

Subversion utilises a concept called "properties" which allows you to attach arbitrary data to files using a key:value pair. The property name is the key and the value can be ASCII or binary data. Subversion itself uses properties to store file configurations. These properties always start with 'svn:', e.g. 'svn:keywords' or 'svn:mime-type'.

Properties can be set, edited, listed, returned and deleted using

`svn propset` ⟨*prop name*⟩ ⟨*prop value*⟩ ⟨*file(s) or dir(s)*⟩
`svn propedit` ⟨*prop name*⟩ ⟨*file(s) or dir(s)*⟩
`svn proplist` ⟨*file(s) or dir(s)*⟩
`svn propget` ⟨*prop name*⟩ ⟨*file(s) or dir(s)*⟩
`svn propdel` ⟨*prop name*⟩ ⟨*file(s) or dir(s)*⟩
or
Right-click on working directory (or file or other dir) → 'TortoiseSVN' → 'Properties' and use the graphical dialog

Table 1: Subversion Basic Subcommands with important options

Command	Alt.	Options	Arguments	Description
add		-N	⟨*files*⟩	Places ⟨*files*⟩ under VC
update	up	-N -r⟨*arg*⟩	[⟨*files*⟩]	Updates local copy
commit	ci	-N -r⟨*arg*⟩	[⟨*files*⟩]	Commits changes to repository
checkout	co	-N -r⟨*arg*⟩	⟨*URL*⟩ [⟨*dir*⟩]	Checks out ⟨*URL*⟩ to ⟨*dir*⟩ or to last dir element in ⟨*URL*⟩
log		-r⟨*arg*⟩	[⟨*path*⟩]	Show log messages of ⟨*path*⟩ or current dir
revert		-R	⟨*path*⟩	Reverts ⟨*path*⟩ to last committed state
propset	ps	-R -F⟨*file*⟩	⟨*name*⟩ ⟨*value*⟩ ⟨*path*⟩	Sets property ⟨*name*⟩ of ⟨*path*⟩ to ⟨*value*⟩
proplist	pl	-R -v	⟨*path*⟩	List properties of ⟨*path*⟩

Table 2: Subversion Command Options

Name	Option	Argument	Description
Non-recursive	-N		Operate on single directory only
Recursive	-R		Descend recursively
Revision	-r	⟨*revision*⟩	Specify revision (by number, date or name)
From File	-F	⟨*file*⟩	Read value from ⟨*file*⟩
Verbose	-v		Print extra information

For all commands and options see [2, Chapter 9] or type "svn help [⟨*subcommand*⟩]".

3 Version Control of a multi-file LaTeX Document

After Subversion has been installed with a repository and working directory set up the version control of your LaTeX document can begin.

3.1 Enable Subversion keyword substitution in your files

In order to have keyword substitution in your LaTeX files a special Subversion property `svn:keywords` must be assigned to them. The property value is a list of the wanted keywords. Normally it is good practise to enable all keywords even if only a subset are actually used. The property is set using

```
svn propset svn:keywords 'Id Author Date Rev URL' *.tex
```

or

Right-click on LaTeX file(s) → 'TortoiseSVN' → 'Properties', press 'Add' and select 'svn:keywords' as name and 'Id Author Date Rev URL' as value, 'Ok'

3.2 Placing Subversion keywords in your files

All LaTeX files of the document must contain one or more Subversion keywords at the very start to allow the calculation of the last change. The shortest way is to use the compact `Id` keyword but the longer `Date`, `Rev`, `Author` and `Revision` keywords can also be used. The package `svn-multi` provides the macros `\svnid` for the `Id` keyword and `\svnidlong` for the other four keywords. Deciding which of the two macros is used depends on the document author. The first macro is shorter but only shows the file name not the whole URL. The second one is clearer arranged and easier to read in source code, while also providing the full file URL. It is also possible to use both together.

The recommendation is to put the following lines before any other code in every source file.

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

Please note that \svnidlong awaits four arguments which can be placed in a single line. No comments should be placed after this arguments because they are processed verbatim.

Subversion expands the keywords at the next commit, so the lines look like this afterwards:

```
\svnidlong
{$HeadURL: file://somewhere/file.tex $}
{$LastChangedDate: 2006-05-26 15:47:47 +0100 (Fri, 26 May 2006) $}
{$LastChangedRevision: 15 $}
{$LastChangedBy: maryd $}
\svnid{$Id: file.tex 15 2006-05-26 15:47:47Z maryd $}
```

3.3 Typesetting keyword values in your document

When each source file has all neccessary keyword macros, the information provided can be typeset anywhere in your document.

There are two groups of values: document-global and file-local ones, i.e. the VC information of the complete document and that of the current file. This allows to typeset the last changed revision number with date and author of the document, e.g. at the title page and also the same information of each single chapter. Please note that the global values are saved in an auxiliary file and have a one LaTeX run delay like the Table Of Contents.

The typeset macros are displayed in Table 3. The date is provided as a full string but also splitted in the single components. The macro \svnkw can be used to typeset arbitary keywords which is useful for non-Subversion keywords which are explained in Section 3.4.6.

3.4 Usage Examples

With the concepts outlined, some procatical examples are now given.

Table 3: `svn-multi` typeset macros

Keyword	Global	Local (current file)
Revision	`\svnrevision`	`\svnfilerevision`
Author	`\svnauthor`	`\svnfileauthor`
Date	`\svndate`	`\svnfiledate`
Year	`\svnyear`	`\svnfileyear`
Month	`\svnmonth`	`\svnfilemonth`
Day	`\svnday`	`\svnfileday`
Hour	`\svnhour`	`\svnfilehour`
Minute	`\svnminute`	`\svnfileminute`
Second	`\svnsecond`	`\svnfilesecond`
Timezone	`\svntimezone`	`\svnfiletimezone`
URL	`\svnmainurl`	`\svnkw{URL}`
Filename	`\svnmainfilename`	`\svnkw{Filename}`
Full Id		`\svnkw{Id}`
Any ⟨*keyword*⟩		`\svnkw{`⟨*keyword*⟩`}`

3.4.1 Putting 'Last Changed' values in header/footer

The header and/or footer line is an ideal location to place some of the 'Last Changed' values of the document. The current revision, last change author and date are normally the main information most people would need. The global/local scheme of `svn-multi` can be used to display the revision number of the document together with the revision number of the current chapter:

```
\usepackage{fancyhdr}

\pagestyle{fancy}

\fancyhead{}
\fancyhead[ol]{\slshape\leftmark}
\fancyfoot[ol]{Rev: \svnrev\ (\svnfilerev)}
\fancyfoot[or]{\svnyear-\svnmonth-\svnday\
\svnhour:\svnminute} %Date
```

```
% If the information should be also placed
% on the chapter page use:
\fancypagestyle{plain}{%
\fancyhead{}
\fancyhead[ol]{\slshape\leftmark}
\fancyfoot[ol]{Rev: \svnrev\ (\svnfilerev)}
\fancyfoot[or]{\svnyear-\svnmonth-\svnday\
\svnhour:\svnminute} %Date
}
```

This gives the following footer line:

Rev: 185 (154) 1 2006-11-10 09:54

where 185 is the last change document revision and 154 the last change chapter revision. The shortened date applies to the whole document.

3.4.2 Placing VC Information on title and chapter page

A good application of `svn-multi` is a longer document, e.g. a technical report, which is written by multiple authors responsible fro writing different chapters. These chapters are sometimes unrelated and therefore change independently. To track this changes, the VC information for the whole document can be typeset on the titlepage while the local values can be typeset for each chapter. Every chapter should lie in a own LaTeX-file which is `\included` in a main file.

The following code showns one way to achieve this. Remember to put a `\svnid` and/or `\svnidlong` on top of every chapter file and also in the main file somewhere after `\usepackage{svn-multi}`.

Titlepage

```
\begin{titlepage}
 \vspace*{\stretch{1}}
 {\Huge Title}
 \vspace*{\stretch{2}}

 \noindent
```

```
 Version control information:\\
 Last changed by: \svnFullAuthor{\svnauthor}\\
 Last changed date: \svndate\\
 Last changed revision: \svnrev\\
 Document URL: \svnmainurl\\
 %Document filename \svnmainfilename\\
\end{titlepage}
```

Chapter page

```
\svnidlong{...}{...}{...}{...}
\svnid{$Id: $}
\ chapter{Introduction}
 Version control information:\\
 Last changed by: \svnFullAuthor{\svnfileauthor}\\
 Last changed date: \svnfiledate\\
 Last changed revision: \svnfilerev\\
 Chapter URL: \svnkw{URL}\\
 %Chapter filename \svnkw{Filename}\\
```

3.4.3 Using full Author and Revision names

The `\svnauthor` and `\svnfileauthor` macros return the username of the author. However in most cases the full author name would be prefered, more legible, and therefore more useful. For this purpose `svn-multi` provides the following macros:

`\svnRegisterAuthor{`⟨*username*⟩`}{`⟨*full name*⟩`}`
`\svnFullAuthor`⟨*⟩`{`⟨*username*⟩`}`

The first macro should be used in the preamble to register all authors of the document. This is done by simply stating the full name of every username. The second macro can then be used to typeset the full name of a username which is normaly given by `\svnauthor` or `\svnfileauthor`. The star version also prints the username in parentheses.
For example:

```
\svnRegisterAuthor{martin}{Martin Scharrer}
\svnRegisterAuthor{other}{Some O. Author}
...
Last changed by: \svnFullAuthor*{\svnauthor}
```

gives

> Last changed by: Martin Scharrer (martin)

assumed that the author "martin" made the last change.

If the macro is used on a non-registered username the (empty) full name is suppressed and only the username is printed.

The same can be repeated for the revision number to assign a name. This has only limited use because the next revision number has to be guessed (it's normally the current revision plus one, but this is not always guaranteed). The macros for this are `\svnRegisterRevision` and `\svnFullRevision`.

3.4.4 Include the VC Values as PDF Metadata

The VC values can be used to generate PDF metadata like the author and the creation date. A trick can be used to avoid problems with the fragile `\svnFullAuthor` macro, the internally called macros `\svnFullAuthor@star` or `\svnFullAuthor@normal` are used directly.

The creation date can be set using `\pdfinfo` and `\svnpdfdate`, which returns the commit date in PDF format, e.g. `20060526154747+00'00'`.

```
% In preamble
\usepackage{svn-multi,hyperref}
\svnid{$Id: file.tex 15 2006-05-26 15:47:47Z maryd $}
\pdfinfo{ /CreationDate (D:\svnpdfdate) }
\makeatletter
\hypersetup{%
  pdfauthor={\svnFullAuthor@star*{\svnauthor}},
  %pdfauthor={\svnFullAuthor@normal{\svnauthor}},
  pdfkeywords={Revision \svnrev}
}
\makeatother
```

3.4.5 Keywords in moving arguments

All `svn-multi` keywords shown in Table 3 are robust and so can be used in moving arguments like chapter or section commands. But the `\svnFullAuthor` and `\svnFullRevision` macros are fragile and therefore must be preceded by `\protect` when used in such places.

3.4.6 Handling Non-Subversion Keywords

Non-Subversion keywords, e.g. user-defined[1] keywords or keywords from other version control software, can be placed and typeset with the `\svnkwsave` and `\svnkw` macros, respectively.

One example, taken from the Usenet group de.comp.text.tex, is:

```
\documentclass{article}

% Place, i.e. save, the keyword
\svnkwsave{$RepoUrl: file:///Server/repos/trunk $}

\ begin{document}
% Typeset keyword
Repository URL is ``\svnkw{RepoUrl}''.
\ end{document}
```

4 Future Features

While `svn-multi` is already "stable", i.e. non-beta, it is still under active development to enrich it with more useful features. User requests and suggestions are always appreciated.

One main feature in planning is the support of namespaces, i.e. to be able to group the keyword values of selected files together. This would expand the global/local scheme by providing values which are "global" only in the current namespace/group. The best example of this is a book or large report which is

1. User-defined keywords are not yet supported by the current Subversion version, i.e. 1.4.4, but might come in a future version.

separated in \parts. By creating a namespace for every part, the latest revision of each part will be available along with the file-local and document-global values. These can then be used to typeset a VC cover on the part page as shown for the titlepage in Section 3.4.2.

References

[1] Subversion project website http://subversion.tigris.org/.

[2] Collins-Sussmann, B. and Fitzpatrick, B.W. and Pilato, C.M., *Version Control with Subversion*, 2004, O'Reilly, ISBN: 0596004486. Available online under http://svnbook.red-bean.com/.

[3] Scharrer M., *svn-multi*, LaTeX package, current version 1.3a, 11 July 2007, CTAN: Package information: http://tug.ctan.org/pkg/svn-multi/, User manual: http://tug.ctan.org/tex-archive/macros/latex/contrib/svn-multi/svn-multi.pdf.

A Suggested Subversion Config File

The following listing shows the recommended Subversion user config file, which add the required properties to all newly added .tex files. The line endings are set to "native" so that the files always have the right format for the used operation system, even when multiple people working on the same document under different systems.

The location of this file is:

Linux/Unix: ~/.subversion/config

Windows: %APPDATA%\Subversion\config
where %APPDATA% is normaly
C:\Documents and Settings\⟨*username*⟩\Application Data

Subversion User Config File

```
[miscellany]
### Ignore LaTeX auxilary files (can also be set with svn:ignore to single
### direcories if not wanted global)
global-ignores = *.aux *.glo *.idx *.ind *.log *.out *.svn *.toc *.ilg *.gls

# Non-englisch people could require the next line
# log-encoding = latin1

# Automatically set the below properties to newly added files
enable-auto-props = yes

### Section for configuring automatic properties.
[auto-props]
*.tex = svn:mime-type=text/x-tex;svn:eol-style=native;svn:keywords=Id Date Author ↙
    Rev URL
*.ltx = svn:mime-type=text/x-tex;svn:eol-style=native
*.sty = svn:mime-type=text/x-tex;svn:eol-style=native
*.cls = svn:mime-type=text/x-tex;svn:eol-style=native
*.cfg = svn:mime-type=text/x-tex;svn:eol-style=native
Makefile = svn:mime-type=text/x-makefile;svn:eol-style=native
*.ps = svn:mime-type=application/postscript
*.eps = svn:mime-type=application/postscript
*.jpeg = svn:mime-type=image/jpeg
*.jpg = svn:mime-type=image/jpeg
*.png = svn:mime-type=image/png
*.dvi = svn:mime-type=application/x-dvi
*.pdf = svn:mime-type=application/pdf
```

THE PracTEX Journal

Travels in TeX Land: Fonts, self-publishing and another reason I like TeX

David Walden

Abstract

In this column in each issue I muse on my wanderings around the TEX world. This column describes some work I did organizing my experiences of using different fonts within TEX, also describes use of an external processor in combination with TEX, and gives an update on my self-publishing efforts using TEX.

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

- PDF version of paper
- font-text-cm-ps.tex
- font-text-cm-ps.pdf
- some-packages.tex
- yfonts.tex
- m4 definitions to produce HTML
- Example interview source file
- m4 definitions to produce LaTeX
- btbook class file
- btbookLSI class file
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2007, No. 3
Article revision 2007/07/29

Travels in TEX Land: Fonts, self-publishing and another reason I like TEX

David Walden

Abstract In this column in each issue I muse on my wanderings around the TEX world. Section 1 of this column describes some work I did organizing my experiences of using different fonts within TEX. Section 2 describes use of an external processor in combination with TEX. Section 3 gives an update on my self-publishing efforts using TEX.

1 Organizing my experience with fonts

It occurred to me since the last column that I have used a number of fonts (i.e., font families) in different documents since I began using TEX seriously half a dozen years ago. However, each instance of a font use was pretty much a one-shot effort with me not having much of an overall picture about the spectrum of fonts available for my use. It seems like it is time to try to organize my experiences with fonts a bit.

In the following, I am merely trying to see which fonts are *easily* accessible—not trying to use the best LATEX (or whatever) conventions for specifying the fonts. In particular, I didn't study the compilation logs to see if there were various issues that needed to be resolved; I would do this if I used one of the fonts for a real project.

When I refer to *The LATEX Companion* below, I mean the second edition of this essential reference book by Frank Mittelbach et al.

1.1 My actual experience using different fonts

I decided to list all of the fonts I had used to date along with how I had accessed the font. Most of the fonts were accessed from LATEX; I have noted the couple of instances when another macro package was being used. Also, with a couple of

exceptions I note below, I did nothing special to have access to these fonts; most these are (apparently) fonts that come with my edition ProTEXt.

For simplicity of constructing this column, I have pointed to examples of the fonts in print rather than providing in-line examples of the fonts listed below.

Palatino. My first big project was using LaTeX to draft the book *Four Practical Revolutions in Management* (http://www.walden-family.com/4prim). For this book project I used Palatino by giving the following command.

```
\usepackage{palatino}  %obsolete now according to The LaTeX Companion;
        % \usepackage{mathpazo} is what is now recommended, I believe
```

(This project was described in a paper in *TUGboat* (http://tug.org/TUGboat/Articles/tb24-2/tb77walden.pdf).

Because it was what I already knew, I also the applied same approach to Palatino for my first column in this journal, TPJ 2005-1. (I'll return again to the Palation font in the next section of this column, where there is a pointer to an example.)

Latin Modern. By the second issue of this journal (TPJ 2005-2), a class file had been written and I used this.

```
\documentclass{pracjourn}
 %which in turn uses \RequirePackage{lmodern} and \usepackage[T1]{fontenc}
```

I also used the journal's default style for my column in TPJ 2006-4 and for *this column*.

For more on the Latin Modern fonts, see Will Robertson's exploration in issue 2006-1 http://www.tug.org/pracjourn/2006-1/robertson. (Will did the bulk of the work to develop the LaTeX class file used for this journal.)

Computer Modern. For the pieces I have written for *TUGboat* (see http://www.tug.org/TUGboat/Contents/listauthor.html), I used the `ltugboat` class which in turn uses Computer Modern (the TEX default). I also used this class (slightly modified but not with regard to font use) for my column for TPJ 2006-3; see the PDF at http://www.tug.org/pracjourn/2006-3/walden.

```
\documentclass[final]{ltugboat}
```

Palatino in eplain TEX. My column in TPJ 2005-4 was an experiment using eplain TEX. I set up the fonts to be used with the following code which I put near the top of my eplain file (unlike LATEX, eplain does not have an explicit preamble). The following code is discussed in more detail at http://www.tug.org/pracjourn/2005-4/walden, where the PDF is also an example of the fonts.

```
\input eplain
\def\fmtname{plain} %added per Oleg to make PDFTeX work on my file

\font\smallrm = pplr7t
\font\bigbold = pplb7t at 14pt
\font\bigbig = pplr7t at 18pt
\font\smalltt = pcrr7t
\font\smallit = pplri7t
\font\tensc = pplrc7t at 12pt

\font\sevenrm = pplr7t at 9pt  % or 8pt or whatever looks right
\scriptfont0 = \sevenrm

%The \ten... command names are used in plain TeX, so by redefining the
%fonts this way, the regular commands \rm, \it, \bf, etc. continue to work.
%The final \rm switches to the new roman font by default.
\font\tenrm = pplr7t at 12pt
\font\tenit = pplri7t at 12pt
\font\tensl= pplro7t at 12pt
\font\tenbf = pplb7t at 12pt
\font\tentt = pcrr7t at 12pt  % Courier; maybe smaller would look better
\rm \baselineskip = 16pt

\everyfootnote{\smallrm \baselineskip = 8pt}
```

Lucida. My column in TPJ 2006-1 described purchasing the Lucida fonts from TUG and trying to use them.

```
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\usepackage{lucidabr}
```

Thus, you can see the font in use in the PDF at http://www.tug.org/pracjourn/2006-1/walden.

Minion Pro. For my book *Breakthrough Management* (http://www.walden-family.com/breakthrough), I used Minion Pro (upon the recommendation of Steve Peter). I already had the Minion fonts on my computer because they came with Illustrator and Photoshop; thus, I felt I could use the font legally. Steve Peter then provided me with a small set of files which made the Minion fonts available to LaTeX and the following commands for accessing a limited subset of the font family. I put this code in my class file first discussed at http://www.tug.org/pracjourn/2006-2/walden and elaborated upon at http://www.tug.org/pracjourn/2006-3/walden.

```
\RequirePackage{minion}
\RequirePackage{microtype}
\linespread{1.0325}

\renewcommand\normalsize{%
   %\@setfontsize\normalsize{11.8pt}{15.3pt}
   \abovedisplayskip 11\p@ \@plus2\p@ \@minus6\p@
   \abovedisplayshortskip \z@ \@plus3\p@
   \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
   \belowdisplayskip \abovedisplayskip
   \let\@listi\@listI}

\renewcommand\small{%
   \@setfontsize\small{10.7pt}{13.05pt}
   \abovedisplayskip 9.5\p@ \@plus2.5\p@ \@minus5\p@
   \abovedisplayshortskip \z@ \@plus\p@
   \belowdisplayshortskip 5.25\p@ \@plus2.75\p@ \@minus2.5\p@
   \belowdisplayskip \abovedisplayskip}
```

This was my first experience using old style numerals.

Because I was in the habit of using Minion Pro for my book and was describing that experience in my columns for TPJ 2006-2 and TPJ 2006-3, I also used that font for the 2006-2 and 2006-3 columns:

```
\usepackage{textcomp}
\usepackage{minion}
\usepackage{microtype}
```

You can see examples of the font in the PDFs at the above URLs.

Antykwa Toruńska. Quite arbitrarily I used the Antykwa Toruńska font for my column in TPJ 2007-1:

```
\usepackage{anttor}
```

It an interesting font; see the PDF at http://www.tug.org/pracjourn/2007-1/walden. However, I'm not sure when I would use it in a real project (rather than just experimenting in my column); nonetheless, it's nice to know the font is available and maybe I can find a use for it in a future document.

Latin Modern (again). In my column in TPJ 2007-2, I described my first experience using ConTEXt. I didn't make an explicit font specification, so I got ConTEXt's default—Latin Modern I believe (see the PDF at http://www.tug.org/pracjourn/2007-2/walden).

Latin Modern Proportional Typewriter. I have begun work on an oral history book (titled *Recollections*) that was originally typed on a typewriter and printed using a copier a Kinko's. I am going to republish that book and probably have it printed by Lightning Source Inc., and thus it will be available via Amazon. I want to maintain the feel of the original font but I'd like left and right justified pages, so I am planning to use a proportional typewriter font. I found such a font in Latin Modern via Will Robertson's paper in TPJ 2006-1 (http://www.tug.org/pracjourn/2006-1/robertson) which I have enabled as follows:

```
\RequirePackage{lmodern}
\RequirePackage[T1]{fontenc}
\renewcommand{\rmdefault}{lmvtt}
\renewcommand{\sfdefault}{lmvtt}
\renewcommand{\ttdefault}{lmvtt}
```

It's odd, I know, to specify typewriter for the roman and sans fonts, but this was the easiest way I found to tell LATEX to use typewriter for everything. (After I wrote the above, Yuri Robbers directed me to the alltt package.)

Times. For several short, miscellaneous documents I have written, I used Times

```
\usepackage{times}
```

although I understand this is also an obsolete way to specific this font and

```
\usepackage{mathptmx}
```

is a preferred approach.

1.2 The fonts chapter in *The LaTeX Companion*

Having listed all the fonts I had used myself to date, I decided to keep investigating which fonts I could easily access without downloading anything new that did not come with my ProTeXt distribution. I started by thumbing through chapter 7 of *The LaTeX Companion*, on fonts and encodings.

On pages 354 and 372 I found summary charts for Computer Modern and the Postscript base set of 35 fonts. I accessed these using the `\usefont{T1}{XXX}{m}{n}` command where `XXX` specified one of the families in those charts. To simplify my testing, I only used the `m` (medium) series and `n` (normal) shape for each font. I also didn't access the Postscript fonts using the packages listed in the chart on page 371. For instance, I used `\usefont{T1}{cmr}{m}{n}` to specify the Computer Modern Roman font.

My test file for the Computer Modern and Postscript base 35 fonts can be accessed from the link `font-text-cm-ps.tex` on the HTML page for this paper, and the output files has the link `font-text-cm-ps.pdf`.

I also looked at some of the packages described in the book, e.g., `eco` and for the Concrete fonts. My test files for these also have links on the HTML page for this column—with the file names including the text `some-packages`.

Finally, I checked out the yfonts. See the links to the files with `yfonts` in their names.

1.3 Fonts I could use from ConTeXt

In my column in the last issue (TPJ 2007-2, I described a simple approach to using fonts from ConTeXt that comes from Bill McClain.

```
\font\myfirstfont=bchb8r      %CharterBT bold
\myfirstfont
```

Since then, I have tried the following variations on the above commands to get other fonts, e.g., .

```
\font\myfirstfont=pplb8r     %Palatino bold
\myfirstfont

\font\myfirstfont=hlhr8a     %didn't work
\myfirstfont

\font\myfirstfont=rpzcmi     %Chancery italic
\myfirstfont

\font\myfirstfont=rpcrr      %Courier
\myfirstfont
```

As noted, this simple approach didn't work for one of the fonts but worked for the others.

Next I tried the fonts given in the examples in the document "Fonts in ConTEXt—Examples of Using Typescripts" (www.pragma-ade.com/general/manuals/showfont.pdf). The first two lines below were needed to make the rest of the pairs of commands work for the various fonts.

```
\switchtobodyfont[ec-lbr]
\usetypescript[berry][ec]

\definetypeface [times] [rm] [serif] [times] [default] [encoding=ec]
\switchtotypeface [times] [12pt,rm]

\definetypeface [charter] [rm] [serif] [charter] [default] [encoding=ec]
\switchtotypeface [charter] [12pt,rm]

\definetypeface [palatino] [rm] [serif] [palatino] [default] [encoding=ec]
\switchtotypeface [palatino] [12pt,rm]
```

```
\definetypeface [zapf] [cg] [calligraphy] [chancery]
\switchtotypeface [zapf] [12pt,cg]

\definetypeface [helvetica] [ss] [sans] [helvetica] [default] [encoding=ec]
\switchtotypeface [helvetica] [12pt,ss]

\definetypeface [utopia] [rm] [serif] [utopia] [default] [encoding=ec]
\switchtotypeface [utopia] [12pt,rm]

\definetypeface [informal] [rm] [casual] [informal] [default] [encoding=ec]
\switchtotypeface [informal] [12pt,rm]
```

The following combinations of commands didn't work, for reasons I don't yet understand.

```
\definetypeface [antykwa] [rm] [serif] [antykwa] [default] [encoding=ec]
\switchtotypeface [antykwa] [12pt,rm]

\definetypeface [bookman] [rm] [serif] [bookman] [default] [encoding=ec]
\switchtotypeface [bookman] [12pt,rm]

\definetypeface [postscript] [rm] [serif] [times] [default]
\definetypeface [postscript] [ss] [sans] [helvetica] [default] [rscale=.9]
\definetypeface [postscript] [tt] [mono] [courier] [default] [rescale=1.1]
\switchtotypeface [postscript] [11pt]
```

1.4 Summary

All in all, what I described above gives me a lots of font flexibility without buying any more fonts or learning about a lot of other fonts or ways of accessing them, especially since there are *many* variations within some of the font families.

Doing this exercise of summarizing the availability of all these fonts is something I would recommend to others who are also trying to mentally consolidate the font capabilities easily available to them. Maybe I'll expand the experiment some time in the future by investigating which fonts that are not in the tables in *The LaTeX Companion* I can find also come with my TEX distribution.

Still, I am looking forward to the day when I have XeTEX (http://scripts.sil.org) installed and working which I hope will allow me to use more fonts, also without much effort.

2 Another benefit of using TEX

As mentioned in my 2006-4 column (http://www.tug.org/pracjourn/2006-4/walden, one reason for using TEX is that one can choose the text editor best suited to one's needs. Another reason for doing typesetting with a system such as TEX where the editor is separate from the typesetting engine (unlike Word or InDesign) is that the markup is explicit (unlike typical use of Word or InDesign). I keep discovering additional reasons why it is useful to work with explicit markup as TEX uses. A capability I used recently was inserting another processor between my editor and the TEX compiler in my work process for one project on which I am working.

2.1 TUG interview series

For the past two and a half years I have been doing interviews for the TUG Interview series (http://tug.org/interviews). The interviews have been carried out as a sequence of plain text emails with me and the interviewee alternating sending questions and answers. Once an interview has been close to complete, I have edited it a little and then converted it manually to HTML for final review by the interviewee and eventual posting to the interview website.

Recently, however, Karl Berry (TUG president) and I have been contemplating creating a book of interviews with any sales proceeds going to benefit TUG. For this we will have to convert the interviews from HTML to TEX or LATEX.

2.2 Switching between HTML and LATEX using m4

To save an extra conversion step with the remaining interviews before we have enough interviews for a book of interviews, I decided to start doing interviews using m4 macros. For more about m4, see:

http://www.gnu.org/software/m4/

http://en.wikipedia.org/wiki/M4_(computer_language)

http://www.gnu.org/software/m4/manual/

Suffice it to say that among macro processors, m4 is powerful and not embedded in some other system (as Knuth's macro processor is embedded in the TEX engine). (See also the Endnote at the end of this column.)

The m4 macros can be defined to produce HTML for the website or to produce TEX/LATEX for the eventual book. This provides the desired capability of being able to insert another processing system in between the interview editing phase and the interview typesetting phase.

I have continued editing the interview source file using my usual editor, WinEdt, but now use m4 macro calls of macros defined to produce HTML for the interviews rather than inserting HTML markup explicitly; see the "m4 definitions to produce HTML" link and the "Example interview source file" link on the HTML page for this column, and the resulting HTML file at http://tug.org/interviews/interview-files/dick-koch.html. (The .m4 files shown by those links temporarily have .txt extensions so they open in a plain text editor when you click on the link.)

I create the .m4 file in WinEdt. Then I compile it under cygwin (http://en.wikipedia.org/wiki/Cygwin) on Windows XP with the command

```
c:/a-files/m4/bin/m4.exe htmldefs.m4 name.m4 >name.html
```

To generate LATEX, I have a different set of definitions at the beginning of the .m4 file as shown in "m4 definitions to produce LaTeX" on the HTML page for this paper, and I compile it with the command

```
c:/a-files/m4/bin/m4.exe texdefs.m4 name.m4 >name.html
```

and, in turn, I compile the name.tex file with PDFlatex.

The example in the LATEX source file is very crude. We will have to significantly refine and expand these definition to convert the actual collection of interviews for the book into LATEX.

2.3 Simplifying command processing

I actually execute those two different commands by executing two different little files under cygwin; see the files named `m4e.txt` and `m4et.txt` in the links list on the HTML page for this column. There may be a better way to execute this under cygwin, but the way I do it is to give the command

```
. m4e.txt
```

where the period says to execute the following file like the contents of the file had been given on the command line. The file can be reexecuted using the command

```
!!
```

as long as no other command has been given in between.

I suppose it is possible to cascade such commands under cygwin to expand the m4 macros and then compile the resulting LaTeX in one command, but I'm not familiar enough with the command shell within cygwin to do that.

2.4 The m4 approach versus other alternatives

Obviously, if one was using very complicated HTML or LaTeX, then it might take a lot of work to develop the two sets of m4 macros and it might well be easier to use some other capability to be able to generate either HTML or LaTeX for a common source file; an example might be TtH (http://hutchinson.belmont.ma.us/tth/), LaTeX2HTML (http://www.latex2html.org/), or TeX4ht (http://www.cse.ohio-state.edu/~gurari/TeX4ht/mn.html). However, with the small set of definitions we need for the interview series, using m4 to target either HTML or LaTeX seems to be a sensible way to go.

3 More about self-publishing my book

In my columns in issues 2006-2 (http://www.tug.org/pracjourn/2006-2/walden), 2006-3 (http://www.tug.org/pracjourn/2006-3/walden), and 2007-1 (http://www.tug.org/pracjourn/2007-1/walden), I described various TeX techniques I used to self-publish a book. I thought my 2007-2 column included my final notes on this, however I was wrong. I have more to report in the following subsections.

More generally, we often talk about TeX's place in the world of typesetting and publishing, in particular why more publishers don't use it (and why many publishers discourage its use). It occurs to me that the world of self-publishing is a place where we should be trying to teach people about the value of TeX. Self-publishers are often people who are already doing a lot of the work of bringing

their writing into publication, and perhaps doing their own typesetting already, and are perhaps not so subject to the constraints of traditional publishers.

I'm not sure how we should best do such promotion of TEX in the self-publishing world. TEX user John Culleton (http://www.tug.org/interviews/interview-files/john-culleton.html) does a good bit of it in the Yahoo Self-Publishing discussion group (http://groups.yahoo.com/group/Self-Publishing), where I continually see how-to questions about things people are having trouble with that are simple to do in TEX. I do believe that the self-publishing world is getting larger as information technology allows disintermediation in publishing as it is doing in so many other fields.

My own intention to keep reporting on what I have learned about self-publishing to readers of this journal about TEX use, and I am trying to get a local Sandwich, Massachusetts, writers group to let me give them a presentation on self-publishing (and TEX). I also am maintaining a separate document that summarizes what I have learned about self-publishing — http://www.walden-family.com/public/notes-on-self-publishing.pdf).

3.1 Converting the book for printing by Lightning Source

As described in my previous columns, the first two printings of my book, *Breakthrough Management*, were done by sending the PDF file of the interior pages of the book to the printer who took the 6 by 9 inch text blocks from the middle of 8.5 by 11 inch pages in the PDF file and placed them on 6 by 9 inch pages for printing. The printers also made slight adjustments I asked for regarding the margins on the printed pages by sliding my entire 6 by 9 inch text block up or down and in or out on the printed pages. The third (corrected) printing of the book (done recently) was done in the same way.

However, in July I decided also to make the book available via Lightning Source Inc. (LSI). LSI is a very large print-on-demand company that is owned by the Ingram Book Group which says it is the "world's largest wholesale distributor" of books. Consequently, anything printed by LSI is supposed to go into the Ingram catalog which in turn means that Amazon and other on-line or traditional book stores can order copies of books printed by LSI.

LSI's specifications said that I had to send them a PDF file with the pages already set up for printing a 6 by 9 inch book. To do this I changed the parameters

for my calls to the geometry package (http://www.tug.org/teTeX/tetex-texmfdist/doc/latex/geometry/geometry.pdf) as described below.

The original PDF files with 6 x 9 inch text blocks on 8.5 by 11 inch pages were produced with the following commands (I'm not sure why I used metric dimensions):

```
\RequirePackage{geometry}
\geometry{paper=letterpaper,lmargin=4.75cm,
   rmargin=4.75cm,tmargin=4.3cm,bmargin=5.57cm}
```

I thought I should be able to just calculate the new parameters for the geometry package, and started doing that calculation as follows:

```
8.5 - old left margin - old right margin => X
6 - X => Y
Y/2 => new left and right margins
```

That worked perfectly. I then turned on the two-sided geometry option which makes left and right be outer and inner and kept sum of the horizontal margins the same while changing to the minimum outer size, thus leaving more space for at the inner margin for binding.

Next I did the following calculation to obtain new top and bottom margins scaled to put the text block at the same place on the smaller pages:

```
old top margin + old bottom margin => Z
11 - Z => W
11 - W => V
old top margin/Z => a
old bottom margin/Z => b
V*a => new top margin
V*b => new bottom margin
```

Unfortunately, that didn't work exactly right, and I had to adjust this by trial and error to get the pages to be the right length. (There was something else I was not compensating for in the above calculation, e.g., headers may not be part of margin calculation.)

In any case, my trial and error adjustments on the results of the above calculation got TEX to perfectly reproduce the pages of text from the previous printings

of the book on the new page size in the PDF file. I added crop marks, which were hard to see because they are at the corners of the 6 x 9 page. I also did a final adjustment, keeping the sum of the top and bottom margins the same while slightly increasing top and decreasing bottom.

The following are the package options I used for what I ended up sending to LSI (the silliness of using metric dimensions for English system page sizes now becomes obvious):

```
\geometry{twoside,paperwidth=6in,paperheight=9in,lmargin=1.88cm,
   rmargin=1.27cm,tmargin=2.289606716cm,bmargin=2.487143284cm}
\RequirePackage[cross,height=9truein,width=6truein,center]{crop}
\RequirePackage{layouts}
```

I have long been using my own class file which is an augmentation and modification of the normal book class which I call using

```
\documentclass[final]{btbook}
```

For the version of the book going to LSI, I created a different class file with the changed described above and called with with

```
\documentclass[final]{btbookLSI}
```

I'm not sure this is the best way to parameterize which page size is being generated, but it's seems like an OK approach.

I don't recommend these class files as examples of good class file writing, but anyone who wants to look at them can access copies of them via the links on the HTML page of this paper. (The actual class files have `.cls` extensions rather than the `.txt` extensions used with this column to facilitate opening the files by clicking the link.)

3.2 Signing up to use LSI

Signing up to do printing with LSI is done via their website (http://www.lightningsource.com), but you also can talk by email and phone with a customer service person assigned to your account. She was very helpful to me as I navigating their web-based forms.

Once I had the interior pages and cover adjusted to meet LSI's specs for PDF files being submitted, I uploaded cover and interior via their web-based process. The status of my account was constantly noted in my LSI account, e.g., "waiting for uploads," "problem with cover (contact Customer Service Representative)," "proof sent and awaiting approval," and "proof approved."

The problem with my cover was a mistake on my part; I had placed the cover art horizontally on a vertical page, and the sides of the cover were cropped off. So, I had to resubmit the cover.

Costs were $50 for cover, 5 cents a page for the 280 pages of the book, and $40 for any revision, including resubmission of my cover. It also cost $30 to have a proof sent to me but that included overnight express shipping.

The cover art looked fine, except it was a glossy coating rather than the matte coating I prefer (there is no option for a matte coating from LSI). The interior printing was definitely inferior to what was done by Ames On Demand (http://www.amesondemand.com) which did the second printing of my book: the text was not as sharp and the B&W images were darker and muddier. This printing was with LSI's purportedly new presses which are supposed to do better graphics; hopefully the improved software which is still being installed with further improve the image and print quality. Nonetheless, it is good to have the book available via LSI, and I approved the proof.

I also asked Lightning Source Inc. to make my book available through their UK branch so that I can order books printed and shipped from there if that will reduce the cost to get books to European customers. This worked smoothly without me having to pay anything more or upload my files again. Also, the printing from Lightning Source in the UK was slightly better than from LSI in the US. Within a day, the book was listed with http://www.amazon.co.uk (and Amazon in Austria, Germany, France, and Japan), and a few days later the book was also shown on the Amazon UK website as being for sale from two other book stores which were undercutting the Amazon UK pricing for the book. I had told LSI to list my book for US$30, 17 GBP, and 24 Euros, with a 55 percent wholesale discount in all cases, but Amazon UK soon dropped its price to close to the low price from the competing UK book store. No doubt Amazon has a program constantly manipulating its books prices to try to maximize profits. (At this point I am still waiting for the book to be listed at Amazon.com in the US.)

In parallel with setting up to work with LSI, I sent the new cover and interior files to Ames On Demand where I will continue to order books a carton (e.g., 25

or 30 books) at a time and mostly use these for filling orders that come in via my website.

3.3 Another option with self-publishing

In the summer of 2007, I began working with with a group in Hungary to provide them with translation rights to the book, for which I still plan to act as publisher in name but with them paying for the translation, retypesetting, and printing, and doing local selling. One of the advantages of self-publishing is that I completely control the rights to the book and can do anything I want without having to get permission from a publisher or convince a publisher to do something.

3.4 More about receiving payments

Originally I had set thing up so I could receive payments for the books I sold from my book website (`www.walden-family.com/breakthrough`) only via PayPal — in particular only via PayPal for credit cards which can be done without becoming a PayPal member.

However, my experience with people trying to use a credit card via PayPal from some foreign countries, e.g., Thailand, is that it can be *hard* to do. They have to do extra work to get their credit card verified by PayPal which takes talking to their bank, etc. In two cases, we have gone around and around with me trying to tell them how to use PayPal within only a credit card, and not a PayPal account, and eventually I gave up, especially since I couldn't see what was happening with PayPal at their end and thus couldn't give good instructions. I opened a bank account that has no significant money in it for these cases and allow wire transfers into it (which apparently is easy for lots of non-US countries). As soon as I see the money in the bank account, I transfer it to my regular bank account (which I don't tell people in foreign countries how to access), and mail the book to them.

Acknowledgements

William Adams reviewed the accuracy of my statements about Word and InDesign. Elizabeth Dearborn coached me in getting my layout ready for LSI to print.

TPJ issue editor Francisco Reinaldo made many useful comments for improving the clarity of this column. Yuri Robbers spotted many typos and made other suggestion for improvement. Karl Berry and Aditya Mahajan helped me access some of the fonts in ConTEXt.

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.

Endnote

The history of m4 makes me feel sentimental about using it. It is derived significantly from Christopher Strachey's GPM ("A General Purpose Macro generator", *Computer Journal* 8,3 (1965), pp. 225–241) which I reimplemented for the PDP-1d at Bolt Beranek and Newman in 1967. M4 itself was also used in the implementation of the Ratfor programming language which I and a small team of programmers used in the late 1970s to implement InfoMail, the first multi-platform email system. We used Ratfor because it produced Fortran code that could be compiled on any operating system and thus provide the computer and operating system independence we needed. For more about Ratfor, see one of the following:

http://sepwww.stanford.edu/software/ratfor.html
http://en.wikipedia.org/wiki/Ratfor
http://wolfram.schneider.org/bsd/7thEdManVol2/ratfor/ratfor.html

THE PracTEX Journal

Ask Nelly:
How do I adjust the height of the square root sign?
How do I create inline numbered lists the LaTeX way?

The Editors

- Comment on this paper
- Send submission idea to editor

TpJ

Q: Dear Nelly: I am writing a Mathematics reader for my students, and in one chapter I have an addition of several terms that all contain square roots. Some of these terms have parentheses under the square root sign, and others do not. This results in square root signs of two different heights, which I find ratehr ugly. Can you tell me how to remedy this?

A: This is a problem with one of those answers that can be hard to discover, but is really easy when you knwo how to do it. LaTeX defines the command

```
\mathstrut
```

which is defined as

```
\vphantom(
```

for such purposes: this is an emtpy box of width 0 but width the height and depth of a parenthesis. Just put such a strut under all square root signs that do not contain parentheses, and they will all end up being the same height as the square root signs that do contain parentheses.

If need be you can also define other struts of different heights.

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

Q: Dear Nelly: I have written a paper in which I use a number of lists. But since those lists have to be inline rather than typeset as an actuallist, I have done them by hand, like this: 1) one; 2) two and 3) three, etc. As I should have expected, I pay the price now that I have to rewrite parts of this paper, and I need to change some lists as well, having to renumber them by hand. Also, I reference list items from the text, so all of that needs to be fixed by hand as well. This does not seem the true LaTeX way to me. Do you know how I can avoid this problem in the future?

A: This can be solved easily using the *paralist* package, which can be found at http://www.ctan.org/tex-archive/help/Catalogue/entries/paralist.html.

This package has several alternative ways of typesetting list, including one that does precisely what you want:

```
\usepackage{paralist}
There are four things to consider, which I will number
within this paragraph:
\begin{inparaenum}[1)]
  \item one;
  \item two;
  \item three;
  \item etc.
\end{inparaenum}
```

This will number your items exactly the way you showed in your letter, and as you can see it allows choosing the way the numbering works when starting the environment. In this case it is arabic numbers followed by a parenthesis, beginning at 1, but you could just as easily have chosen "a." for letters followed by a period.

The *paralist* package has far more options and ways of typesetting lsits, so do read the documentation to see what else it allows you to do!

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

Distractions — Alea iacta est!

The Editors

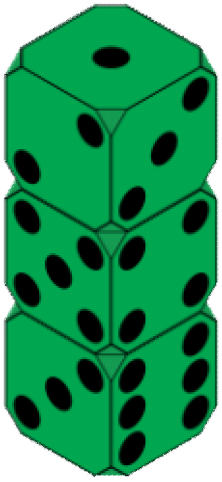
- Comment on this paper
- Send submission idea to editor

Alea iacta est!

In this issue we present a small puzzle that has been typeset using a not too well known extension to *PSTricks*, namely *pst-ob3d*. The package is available from CTAN and can be found at the address http://www.ctan.org/tex-archive/help/Catalogue/entries/pst-ob3d.html

The package can be used to construct images involving three-dimensional shapes. One of the pre-defined shapes is the die. Any number of dice in various orientations, sizes, etc. can be drawn easily. These dice can have rounded corners or regular pointy corners, and the colour and size of the dice as well as their attributes (dots, corners, lines, etc.) can easily be changed with optional parameters

We have created a simple puzzle using a stack of three randomly oriented dice, using the `Randomfaces=true` option of the `PstDie` command as follows:



The question is: what is the sum of the dots on the faces that are both *horizontal* and *hidden*?

The LaTeX source code is available here.

The solution is available here.

Enjoy!

Alea iacta est! (The die is cast!)

There are five faces that are horizontal and hidden: the bottom face of the top die, and the top and bottom faces of the other dice. Since on normal dice the number of dots on two opposite faces adds up to seven, there are two times seven, or fourteen dots hidden on the horizontal faces of the two bottom dice. The top die has seven dots in total on its two horizontal faces as well, minus the one dot that is visible, equals six. So in total there are fourteen plus six, or *twenty* dots hidden on horizontal faces.