Search
(courtesy of Google)

THE PracTEX Journal

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

Archives of *The PracTeX Journal*

Back issues
Author index
Title index
BibTeX bibliography

Next issue
Approx. August 15, 2006

Current Issue

2006, Number 2

[Published 2006-05-17]

Notices

Articles

Columns

Other key people

More key people wanted

How do I interrupt an enumerate environment and then continue it later in the document?
The Editors

Distractions:

Sudoku ABC
Winners of the type quizzes
The Editors

THE PracTEX Journal

From the Editor: In this issue

Lance Carnes

- Comment on this paper
- Send submission idea to editor

In past issues authors have written about making slide presentations with LaTeX packages: **HA-prosper** [Miller], and **beamer** [Mertz, Slough]. In this issue Thomas A. Schmitz shows how to create a presentation with ConTeXt. He provides several presentation examples in both source and PDF that you can try (and admire).

Jan Hlavacek describes the Ipe graphics editor, which he uses to create images for including in LaTeX documents. It's a nicely-designed multi-platform (Windows, Linux, Mac) tool that I'm sure you will want to try — several of the article's reviewers commented that they will be trying it soon.

Many of you reading The PracTeX Journal have found yourself in the same position as Stephen J. Eglen — your colleagues or students have asked you to give an introductory class on the use of LaTeX. Stephen presents his class teaching materials along with an account of his classroom experience. How does his experience match with yours?

The next paper deals with a one-week course on LaTeX, given in India by D. V. L. K. D. P. Venugopal. This course, complete with a quiz, was presented several times to large classes. Mr. Venugopal has made his teaching slides available, and describes some variations in the class format.

Is LaTeX really that good at creating documents, and easy to use? Andy Roberts wrote an article on LaTeX for an online magazine, OSNews, which we are reprinting here with permission of the author and OSNews. He describes LaTeX and some reasons why you might want to use it instead of Word or other formatting systems. When I approached Andy with the idea of reprinting the article in *The PracTeX Journal* he suggested we also include some of the OSNews reader comments, which we have done. After reading the article and comments let us know what you think.

David Walden's Travels in TeX Land column talks about writing books with LaTeX and some productivity tricks he has picked up. (Dave will be guest-editing the next PracTeX Journal issue.) Ask Nelly answers some questions about the pdfTeX program, the engine under the hood of most all current TeX systems, and also helps with some LaTeX problems that may occur when dealing with publishers. And finally, for those who like puzzles and challenges, Distractions offers some Sudoku puzzles with letters instead of numbers. You can also read about the winners of last issue's font contests!

PracTeX Journal readers provided insightful feedback. As you read the articles and columns please use the response links to send comments. If you use a technique from an article or column, be sure to contact the author and report how it worked for you. *The PracTeX Journal* is still evolving and your feedback will help us as we strive to improve it.

Next Issue

Editorial board member Dave Walden will serve as guest editor for the next issue. As indicated by the subject of his column in the current issue, Dave is particularly interested in methods of improving writing and publishing productivity by using TeX and friends (beyond the benefit of TeX and friends for doing beautiful typesetting). He would especially appreciate people submitting papers on this subject. Productivity benefits might include (a) methods of easily and repeatedly changing the format of text, (b) TeX/LaTeX as a standard for communication, or (c) TeX et al. as basis for easy reuse of text. People may also have ideas for how to better organize all their files, maintain an up-to-date distribution, and other techniques that may make use of TeX and friends more productive to use. Of course, Dave will welcome papers for the next issue on any other subject appropriate to TPJ's beginning and intermediate level readers.

If you would like to contribute an article or technical note on productivity or any aspect of LaTeX, TeX, or ConTeXt, please send an article outline to the editors.

Curious how the PracTeX Journal works? Visit the PracTeX Journal wiki. Feel free to log in and suggest an article topic.

Thanks

The Editorial Board and I want to thank the authors, columnists, and Ask Nelly answerers for their excellent pieces which make this journal possible. We also want to thank those who

worked behind the scenes:

Reviewers and copy editors: Jon Breitenbucher, David Elliott, Peter Flom, Jenny Levine, Adam Lindsay, Steve Peter, Tarcisio Praciano-Pereira, Yuri Robbers, Will Robertson, and Dave Walden.

Production editors: Yuri Robbers, Will Robertson.

See also other key people who make this publication possible.

And finally, be sure to attend Practical TeX 2006, a workshop and conference being held this summer at Rutgers University in New Jersey. You'll meet other PracTeX Journal readers there, and learn first-hand about using LaTeX, TeX, and ConTeXt.

Lance Carnes
Editor

THE PracTEX Journal

Feedback

From Readers

- Comment on this paper
- Send submission idea to editor

I'm very surprised that George Williams' `fontforge' editor isn't mentioned in the `Font Development Tools' section [Riggs]. Other tools like Just van Rossum's `TTX' or even Microsoft's `VOLT' are missing too. Adobe's Font Development Kit should also be in this list, and probably a few other important tools also.

Werner Lemberg

TpJ

Intrigued by the article by Tamye Riggs, I followed the link to "Microsoft Typography Foundry List" http://www.microsoft.com/typography/links/links.aspx?type=foundries&part=1 from which I did a search for one of Microsoft's most popular faces, "Verdana". Their search engine returned "Channel Verdana" http://www.microsoft.com/typography/web/fonts/verdana/default.htm as the first hit, from which I followed the link inviting me to "download the Verdana family" href="http://www.microsoft.com/typography/fontpack/default.htm. At which point, Microsoft kindly told me

"We're sorry, but there is no Microsoft.com Web page that matches your entry."

Wonderful. When *will* Microsoft get their internal links in order? :-(((((

(Yes, I know about the copyright issues : a shame Microsoft didn't, when they first offered Verdana/Tahoma/Trebuchet MS/Georgia for free download ...).

Philip Taylor

TPJ

[Whole issue] Thanks, issue is beautiful, interesting and helpful. One nit, why don't my links work?

Very good job,
John Dickinson

[Editor and web site maintainer David Walden has made two "whole issue" PDF versions available, one with links and one without. -Ed.]

TPJ

Excellent!

Hong Feng
Chinese TeX Users Group

TPJ

On your web page it says that the next issue of PracTeX Journal is due out approx May 1, 2006. As it is now May 15, do you have a revised release date? I'm looking forward to reading it, and am chomping at the bit.

Mark Stephenson
New Zealand

[Did we say May 1? Hope you enjoy this issue, Mark. -Ed.]

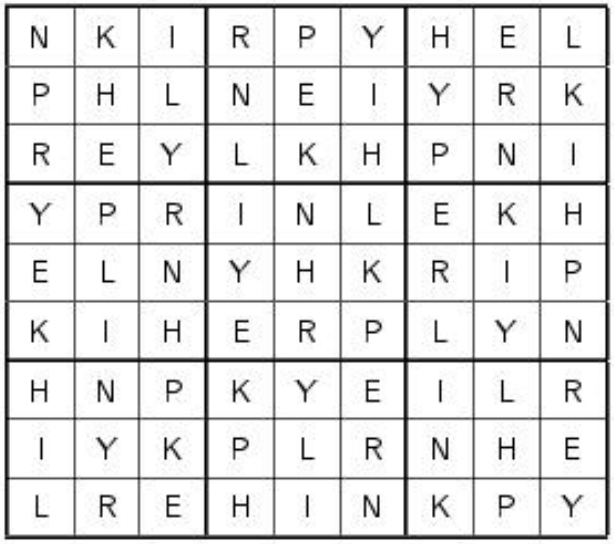
TPJ

Sudoku ABC Answers

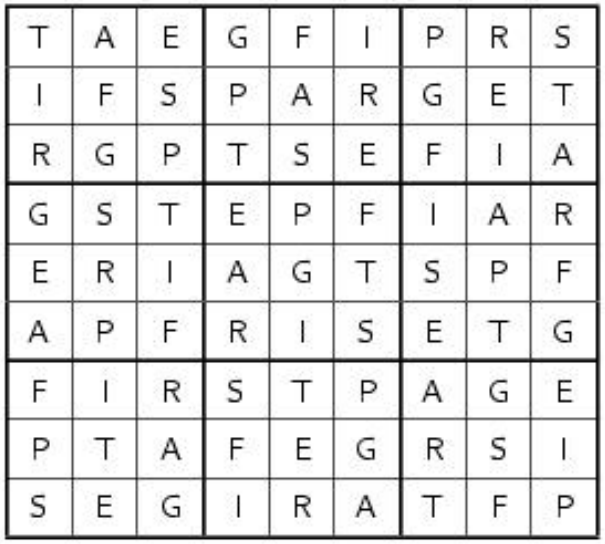
1.

2.

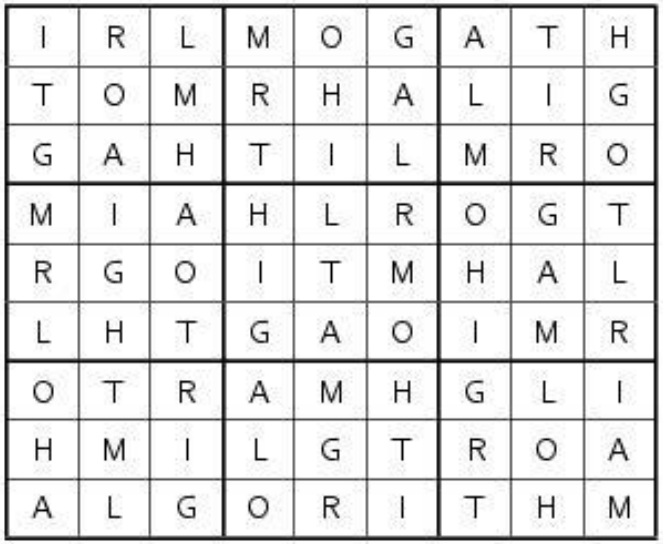
3.



N	K	I	R	P	Y	H	E	L
P	H	L	N	E	I	Y	R	K
R	E	Y	L	K	H	P	N	I
Y	P	R	I	N	L	E	K	H
E	L	N	Y	H	K	R	I	P
K	I	H	E	R	P	L	Y	N
H	N	P	K	Y	E	I	L	R
I	Y	K	P	L	R	N	H	E
L	R	E	H	I	N	K	P	Y



T	A	E	G	F	I	P	R	S
I	F	S	P	A	R	G	E	T
R	G	P	T	S	E	F	I	A
G	S	T	E	P	F	I	A	R
E	R	I	A	G	T	S	P	F
A	P	F	R	I	S	E	T	G
F	I	R	S	T	P	A	G	E
P	T	A	F	E	G	R	S	I
S	E	G	I	R	A	T	F	P



I	R	L	M	O	G	A	T	H
T	O	M	R	H	A	L	I	G
G	A	H	T	I	L	M	R	O
M	I	A	H	L	R	O	G	T
R	G	O	I	T	M	H	A	L
L	H	T	G	A	O	I	M	R
O	T	R	A	M	H	G	L	I
H	M	I	L	G	T	R	O	A
A	L	G	O	R	I	T	H	M

TPJ

THE PracTEX Journal

Invitation to PracTeX'06

Robin Laakso

- Visit the Practical TeX 2006 home page
- Publicity flyer for the conference (please post)
- Email questions about conference
- Email submission idea for conference

Photos of Rutgers

On behalf of the TeX Users Group and Rutgers, The State University of New Jersey, I invite you to attend the Practical TeX 2006 workshop and/or conference. Both the workshop and conference will be held at Rutgers. Building on the success of PracTeX 2004 in San Francisco and PracTeX 2005 in Chapel Hill (NC), this year we're offering a full four-

day LaTeX workshop, followed by three days of talks and presentations. All aspects of the workshop and conference will focus on practical techniques for document production using LaTeX, TeX, ConTeXt, MetaPost, and friends.

Hope to see you there!

Robin Laakso
TeX Users Group

Practical TeX 2006

Workshops and Presentations on LaTeX, TeX, ConTeXt, and more

LaTeX Workshop: July 25-28, 2006
Practical TeX Conference: July 30-August 1, 2006

Bush Campus, Rutgers, the State University
Piscataway, New Jersey, USA

http://tug.org/practicaltex2006
conferences@tug.org

Keynote address: Barbara Beeton, American Mathematical Society and TeX Users Group

Conference web pages:

- **Register!**
- **Call for papers:** abstracts due April 1, 2006.
- Conference program & participants.
- LaTeX workshop.
- Conference sponsorship opportunities.
- Rutgers/New Jersey local information.
- Publicity flyer (please post).

This four-day conference focuses on practical techniques for document production using LaTeX, TeX, ConTeXt, MetaPost, and friends. It includes one day of classes and tutorials, followed by three days of presentations.

Hope to see you there!

Further information

Conference attendees will enjoy an opening night reception and an (optional) banquet one evening. Coffee and lunch will be served each day of the meeting. Located on the Busch Campus of Rutgers University in Piscataway, New Jersey, an easy train ride from New York City.

Conference fee and hotel information is available on the registration page.

Conference flyer and publicity

Mentioning the conference to colleagues and in any other contexts would be very much appreciated.

We'd also be grateful for any posting of this one-page PDF flyer (TeX source).

Sponsorship

If you or your organization would like to help sponsor the conference, numerous options are available, from a straight cash donation (always welcome!) to logos on the conference memorabilia. Please see this separate sponsorship page for details, or email us.

We are very grateful to many contributions.

Contact

Email: conferences@tug.org
Phone, fax, postal mail: see TUG office contact information.

Sponsored by the TeX Users Group. This conference follows 2004's Practical TeX conference in San Francisco and 2005's Practical TeX conference in Chapel Hill, NC.

THE PracTEX Journal

Presentations in ConTeXt

Thomas A. Schmitz

Abstract

TeX is an excellent tool to produce pdf presentations. This paper will show you how to use ConTeXt for writing presentations, and it will teach you to prepare a single source file that can output a presentation, a lecture manuscript, or a handout, if you adapt one single switch. The article is suitable for beginners in ConTeXt, but it should also have interesting things for more advanced users.

Thomas A. Schmitz is Professor of Classics at Bonn University, Germany. After his last book project, he became so dissatisfied with the infamous word processor from a Redmond-based software company that he took the plunge and learned to use ConTeXt, a very nice TeX macro package. And he hasn't looked back ever since. Thomas can be reached at thomas.schmitz@uni-bonn.de

- PDF version of paper
- Example 1 (source)
- Example 1 (output)
- Example 2 (source)
- Example 2 (output)
- Handout output example
- Combined slides/notes output example
- Manuscript output example
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2006, No. 2
Article revision 2006/05/04

Writing Your Own Presentation Style in ConTEXt

Thomas A. Schmitz

1. Introduction

If you use any TEX macro package for your work and have to give presentations, you have probably thought about ways of using TEX for these presentations as well. In principle, this is easy: you can produce a pdf file, and many pdf viewers have a fullscreen mode which allows you to show this pdf page by page, thus presenting a series of "slides."

Both LATEX and ConTEXt have dedicated packages and modules for this job: in LATEX there's the `prosper` or the `beamer` packages which allow to produce such pdf presentations (Michael Wiedmann has written an in-depth **comparison**[1] of different solutions). In ConTEXt, there's a number of predefined modules with very advanced visual effects; there's a **sample document**[2] on the pragma website (it's about 3 MB).

While this is wonderful, I found it extremely instructive to create my own style for presentations. The customizability of ConTEXt is so great that it is quite easy to do, and instead of relying on predefined styles and code you may not understand, you're in control of everything. This article will introduce a way of writing your own presentation style. And there will be one huge benefit: it will teach you how to have the manuscript for your lecture and the material for your slides in the same source file, allowing you to produce different output with just one switch.

The article does not presuppose prior knowledge of advanced features; it is suitable for beginners in ConTEXt.

[1] `http://me.in-berlin.de/~miwie/presentations/presentations.html`
[2] `http://www.pragma-ade.com/show-pre.pdf`

2. The Concept

Before we actually start writing our file, we just think about its structure and the ConTEXt tools that we will be using. What we want: we want one single source file which will output either a presentation that can be run in the full-screen mode of a pdf viewer or a manuscript for our lecture. There is one feature in ConTEXt which will allow us to achieve this aim; it's called "modes" (and it has its own **modes manual**[3]). It allows us, as it were, to have two documents within one source: certain parts of the source file are marked as belonging to one mode and are typeset only if this mode is explicitly turned on (there's a similar feature for LATEX; it's called "conditional text", and there's a section on it in the **UK TEX faq**[4], but as far as I can see, it's less advanced than ConTEXt modes). Modes are sort of a preprocessor command: when ConTEXt reads your source file (a process which is called "parsing") and is passed a mode, it will "see" only the parts that are marked as belonging to this mode. Two switches will be especially interesting for us:

```
\startmode[NAME]
...
\stopmode
```

Everything that goes between these commands is only effective if this mode is enabled. This can be commands and setups, definitions of macros, or simply text.

```
\startnotmode[NAME]
...
\stopnotmode
```

Easy to guess: everything between these commands is effective whenever the mode is *not* enabled.

Enabling a mode is easy. You can either have one line in your sourcefile

[3] http://www.pragma-ade.com/general/manuals/mmodes.pdf

[4] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=conditional

```
\enablemode[NAME]
```

and either comment it out (with a % in front of it) or leave it active (remember that this line has to be placed *before* the definition of your modes!). Or you can pass `texexec`, the command-line tool for ConTEXt, the mode as an option:

```
texexec --mode=NAME myfile.tex
```

Now that we know which toolset we are going to use, let's go to the drawing board and plan how we will be using it to achieve our goal. Here is what I suggest:

1. We will define a mode "manuscript" which will contain the text for our lecture notes. This text will of course not be visible on the slides, so it will be wrapped in pairs of `\startmode[manuscript]` and `\stopmode`. The same is true for any setup and definition which we want to be effective for the manuscript only.

2. For the slides, on the other hand, we will have a particular set of setups which enable, say, a color theme, the right size for screen documents, and other bells and whistles. We could do this with `\startnotmode[manuscript]` and `\stopnotmode` commands, but I suggest defining a mode of its own, `\startmode[presentation]`.

3. I suggest that we include miniature versions of our "slides" in the text of our manuscript so we know what our audience is seeing without having to turn around to the screen. That means the text and graphics for our slides will be typeset for the slides themselves as well as for the manuscript.

3. Setups for the Slides

Let's first think about the slides; they need more special setup than the manuscript. So we start defining: we write `\startmode[presentation]` and use the ConTEXt commands to set things up.

3.1 Paper Size

ConTEXt has predefined "paper" sizes for screen documents (they are in landscape and match the usual ratio of computer screens). These sizes are defined and described in the ConTEXt module `page-lay.tex`; the width and height are roughly 4:3. The actual dimensions (defined in the ConTEXt module `page-lay.tex`) are:

```
[S3] [width=300pt,height=225pt]
[S4] [width=400pt,height=300pt]
[S5] [width=500pt,height=375pt]
[S6] [width=600pt,height=450pt]
[S8] [width=800pt,height=600pt]
[SW] [width=800pt,height=450pt]
[SM] [width=720pt,height=450pt]
```

S6 has approxiately the width of a sheet of A4 paper, so we write:

```
\setuppapersize[S6][S6]
```

3.2 Page Layout

Margins and space for headers/footers are a bit special for a screen document. Here's an example of the layout I use for my own slides:

```
\setuplayout [width=fit,
              rightmargin=1.5cm,
              leftmargin=1.5cm,
              leftmargindistance=0pt,
              rightmargindistance=0pt,
              height=fit,
              header=0pt,
              footer=5pt,
              topspace=2.5cm,
              backspace=1.5cm,
```

```
bottomspace=.8cm,
bottom=12pt,
location=singlesided]
```

I will not go into the gory details of page composition (for which you can consult chapter 3 of the **ConTEXt manual**[5]). Just a few words about the values above: the text area of our slides is calculated automatically; that's why we have `width=fit, height=fit`. This text body is located 1.5cm from the right and left edge (`backspace=1.5cm`), 2.5cm from the top (`topspace=2.5cm`) and 0.8cm from the bottom edge (`bottomspace=.8cm`). ConTEXt offers the possibility to place elements into this `backspace` area; if we want that (and we might!), we have to assign a percentage of this area to the "margin". In our case, we want to use the entire `backspace` left and right for such "marginal" material (`rightmargin=1.5cm,leftmargin=1.5cm`). We will not be using a header, but later, we will be typesetting elements in the bottom part of our slides; that's why we have (`footer=5pt,bottom=12pt`).

These values are merely empirical; I find that my presentations look good with them, but feel free to experiment.

3.3 Page Numbers and Colors

We'll want colors for our presentation slides, and we do not want pagenumbers appearing on them. So we type:

```
\setupcolors[state=start]
\setuppagenumber[state=stop]
```

Now we have to go for a color scheme for our slides. This is the moment to bring your personal style to your slides—but don't go too wild, your audience will be grateful. In classes I taught, I have made completely unrepresentative polls, and most people thought that white text on a dark blue background was most legible. So let's go for this look (again, experiment to see if you like other combinations better):

[5] `http://www.pragma-ade.com/general/manuals/cont-eni.pdf`

```
\setupbackgrounds[page][background=color,backgroundcolor=darkblue]
\startcolor[white]
```

3.4 Defining the Slides

This was the general structure for our documents. We now define the specific look of our slides. We want a "normal" bodyfont of a size of about 24 pt (just how big you want your bodyfont will of course depend on a number of factors: the font you are using, the amount of text you want on your slides, the size of the screen, etc.). In order to have a uniform look for our slides, we will put this into an environment defined by a `\start...\stop` pair. Of course, every slide will be on a "page" of its own; that's what the command `before=\page` does.

```
\definestartstop
   [Slide]
   [commands={\switchtobodyfont[24pt]},
    before=\page]
```

Finally, we want (almost) every slide to start with a centered title in a somewhat larger font and with a blank line after it. So we define a command that will do just that:

```
\define[1]\SlideTitle
  {\midaligned{\tfb #1}
   \blank[line]}
```

We now close the setups for our slides by typing `\stopmode`

4. Setups for the Manuscript

Everything we have done so far will merely modify the look of our slides; for the manuscript, we have left everything at its default settings (which are perfectly alright).

There's only one thing that is absolutely necessary to have: we have defined an environment `\startSlide` and a command `\SlideTitle`, but only for the presentation mode. Since we want the material of the slides to be typeset in the manuscript as well, we must define both for manuscript mode as well, or TEX will complain about "undefined control sequences." In manuscript mode, we don't want `\startSlide` to start a new page and to change to large font sizes. One way to incorporate the material on the slides into your manuscript would be to have it in a little frame in the middle of the page, with an empty line preceding and following it. ConTEXt offers an environment to do this, `\startframedtext`. So all we have to do is set this up and make our own environment `\startSlide` call this command:[6]

```
\startmode[manuscript]
\setupframedtexts[location=middle,
                  before={\blank[line]},
                  after={\blank[line]}]

\let\startSlide\startframedtext
\let\stopSlide\stopframedtext
```

Moreover, we need to redefine the command `\SlideTitle`. In our manuscript, we just want the title to be centered and followed by a small blank space. So we define:

```
\define[1]\SlideTitle{\midaligned{#1}\blank[small]}
```

This is the basic setup. The line `\stopmode` finishes the setup for our manuscript mode.

5. Making it Work

Instead of including a lengthy example in this article, I append a **ConTEXt file** which will give you some ideas. Of course, you can use all the concepts ConTEXt offers on

[6] For the curious: `\let\a\b` "clones" a control sequence `\b` to a new name `\a`; you can find an in-depth explanation in the *TEXbook*, p. 206–7.

these slides: environments, lists, graphics, figures, math, just about anything. Two pdf files are obtained from the same source; one has been compiled in **manuscript mode**, the other in **presentation mode**. If you have a ConTEXt-installation on your computer, you can try it yourself: just compile the attached file. The first line reads `\enablemode[presentation]`; if you change it to `\enablemode[manuscript]`, you will typeset the manuscript for an imaginary lecture, with the slides included as small framed images. Play with the file and test things on the slides.

6. More Bells and Whistles

The slides you get from compiling our example file are very basic; they are just meant as a first encouragement to get you started. If you want your slides to look a bit more refined, here are some ideas:

6.1 Fonts

ConTEXt uses the Latin Modern font by default; this is a replica of Knuth's original Computer Modern typeface (see **Will Robertson's article**[7]). While it is a marvelous font, I find it's not quite suitable for a screen presentation. There's no problem enabling a different font for your slides. One possibility would be to go for a sans serif font like Helvetica. In that case, just put these two lines into the `\startmode[presentation]` section:

```
\usetypescript[helvetica] [ec]
\setupbodyfont[helvetica,ss,24pt]
```

and your slides will be in Helvetica (while your manuscript will be unchanged). A font that I find very legible on such slides is Palatino. This would be enabled with these two lines:

[7] `http://www.tug.org/pracjourn/2006-1/robertson/robertson.pdf`

```
\usetypescript[palatino] [ec]
\setupbodyfont[palatino,24pt] % or maybe 20pt, Palatino is big!
```

You can learn more about the free fonts that are probably already available in your TEX installation at the **ConTEXt wiki**[8].

6.2 Backgrounds and Colors

Our blue background is certainly nice, but with ConTEXt, we can do better. You can define almost anything as a background to your slides: solid colors, a picture, a graphics file. Something I find very impressive is a background with two colors that are interpolated, i.e., fade into each other. This interpolation can be achieved with MetaFun, a macro package for METAPOST that enhances the graphics capabilities of ConTEXt. So instead of just setting the background to `darkblue`, we put some MetaFun code into the background of our slides:

```
\definecolor[lightblue][r=0,g=0,b=1]
\definecolor[darkblue][r=0,g=0,b=0.05]

\startuniqueMPgraphic{LinearShade}
path p ;
p := unitsquare xscaled \overlaywidth yscaled \overlayheight ;
linear_shade(p,6,\MPcolor{darkblue},\MPcolor{lightblue}) ;
\stopuniqueMPgraphic

\defineoverlay[shaded][\useMPgraphic{LinearShade}]

\setupbackgrounds[page][background={shaded}]
```

We have defined two colors; `lightblue` is a very light, `darkblue` a very dark blue. We let MetaFun calculate an interpolation between both colors. If you want to know more about the details of how this works, you can read section 8.1 in the **MetaFun manual**[9],

[8] `http://wiki.contextgarden.net/Psnfss`

or you can just experiment with other values for the colors or for the interpolation (try all numbers from 0 to 9 for the `p,6` variable).[10]

You can have even fancier effects; have a look at the attached file `example2.tex` to see what can be done!

6.3 Pictures

ConTEXt's figures mechanism makes it easy to integrate pictures into your slides. There's a number of points we have to consider:

1. ConTEXt relies on pdfetex, so the supported formats (in addition to embedded metapost code) are `.pdf, .png, .jpg`.[11] So if you have a picture in a different format, you will need to convert it.

2. You can either have all the pictures in the same directory as your source file, and TEX will find them automatically, or you can name a path explicitly via the command `\setupexternalfigures[directory=/PATH/TO/PICTURES]`.

3. Our pictures will be shown on the slides as well as in the manuscript. Since these modes have different setups in terms of textwidth and location of slides, we will have to make two definitions for the dimensions of the picture, in `\startmode ... \stopmode` environments.

How do you want to present your graphics on your slides? This will, of course, depend on a number of factors: the nature of the pictures you want to show, their size, and the relation between graphics and surrounding text. Here are some possibilities:

---

[9] `http://www.pragma-ade.com/general/manuals/metafun-s.pdf`

[10] If you want this metapost code to be executed as you typeset your document, you may need to adjust two configuration files in order for this to work: in `texmf.cnf`, you have to have the line `shell_escape = t`; in your `cont-sys.tex`, uncoment the lines `\runMPgraphicstrue` and `\runMPTEXgraphicstrue`. If you don't you will have to run `texexec` twice to see the results.

[11] A number of documents claims that `.tif` is supported as well, but this is not the case for current versions of pdfetex.

1. If you want your picture to take an entire "paragraph" of your slide, without any additional material next to it, you can simply place it via the command `\externalfigure[NAME][height=.75\textheight]`.

2. It's a bit more complicated to have two pictures or text and pictures side by side. One way to achieve this is via the `\startcombination` and `\stopcombination` environment.[12] It allows you to have a picture and text or another picture on two "columns" on your slide. Instead of a lengthy explanation, see the example in the attached file.

3. Or, the `\setlayer` command allows you to place pictures at arbitrary positions on your slides, independent of surrounding text. Layers are a powerful, yet complex feature of ConTEXt; they are documented in chapter 6 of the **details manual**[13].

It's difficult to give any general receipe; everything depends on your needs and your individual workflow. Have a look at the attached files to get some inspiration of what is possible.

6.4 Progress Meter

One feature I find very nice is a predefined module in ConTEXt. It provides, at the bottom of your "slides," a progress meter that shows how many slides have already been shown and how many more will follow. The feature is called "interactionbar." In order to get it, just put this in the definition of the "slides" mode:

```
\setupinteraction
  [page=yes,
   color=InteractionColor,
   contrastcolor=ContrastColor,
   menu=on,
   state=start]
```

[12] If you want to know more about this environment, consult section 12.3 in the ConTEXt manual.

[13] `http://www.pragma-ade.com/general/manuals/details.pdf`

```
\startinteractionmenu[bottom]
\rightaligned{\interactionbar[alternative=f,width=\makeupwidth,height=1ex]}
\stopinteractionmenu
```

If you want to have such an "interactionbar," you will have to modify the definition of our `\startSlide`: the `\start ... \stop` defines a grouped command and prevents ConTEXt from "seeing" the page breaks. We now define collected setups for the commands `\startSlide` and `\stopSlide`:[14]

```
\def\startSlide{\directsetup{slide:start}}
\def\stopSlide {\directsetup{slide:stop}}

\startsetups slide:start
 \start
 \switchtobodyfont[24pt]
 \setupinteractionbar[state=start]
\stopsetups
\startsetups slide:stop
 \page
 \egroup \stopsetups
```

This will put a progress meter on every slide. I have attached an example with these fancier settings, both in the **source** and the compiled **output**.

6.5 Handout

Now let's suppose that after typesetting your document, you have the idea of giving the material of your slides to your audience as a handout. That's easy to do: you just have to define another mode; let's call it `handout`. What we want: typesetting the text and the graphics on the slides, but without the colored background. Every slide will appear in a small frame, and there will be two columns on every A4 (or letter) page.

[14] Hans Hagen has contributed this code; for this and for his tireless efforts in developing ConTEXt, I am extremely grateful.

Here's the setup:

```
\startmode[handout]
\setuppapersize[A4][A4]
\setuplayout [width=fit,
              rightmargin=0cm,
              leftmargin=0cm,
              backspace=1.4cm]
\setupframedtexts[location=left,
                  width=\textwidth,
                  height=.75\textwidth,
                  before={\blank[line]},
                  after={\blank[line]}]
\let\startSlide\startframedtext
\let\stopSlide\stopframedtext
\define[1]\SlideTitle{\midaligned{#1}\blank[small]}
\usetypescript[palatino] [ec]
\setupbodyfont[palatino,11pt]
\setupcolumns[2,distance=2mm]
\stopmode
```

We can be brief in describing this since you've encountered most of the commands already: we use A4 paper, and we set the margins pretty narrow since every frame carries its own margin. The framed "slides" have a blank line before and after the frame. For our manuscript, we simply let the content of the embedded slides determine the height; for the handout, we give a fixed height which is ¾ of the slide's width (`height=.75\textwidth`). `\startSlide` just triggers the frame. The command `\SlideTitle` is defined to center the title and leave a small blank space after it, and we choose a bodyfont of 11 pt. Finally, we define two columns with a space of only 2 mm between them (again, because every "slide" has its own margins). After setting this up, we have to put this at the beginning of the document, just after the line that says `\starttext`

```
\startmode[handout]
\startcolumns
\stopmode
```

to start setting the handout in two columns. And of course, you will need to end this columnset at the end of the document:

```
\startmode[handout]
\stopcolumns
\stopmode
```

Now typesetting a handout of your slides is as easy as modifying the first line of our examples to `\enablemode[handout]`. Again, you'll find an example attached to this article: it is derived from the same source file, compiled in **handout mode**.

7. Conclusion

Of course, an almost infinite number of optical improvements is possible; you really should have a look at the predefined presentation styles and let this inspire you. But the method described here should allow you to understand the mechanism behind it all and to develop your own personal style. And it should demonstrate how powerful the mechanism behind "modes" in ConTEXt is; it allows for almost unlimited possibilities. You will certainly find more use for it now that you've seen what it can do for you.

THE PracTEX Journal

Ipe — a graphics editor for LaTeX

Jan Hlavacek

Abstract

This article talks about author's experiences with the Ipe graphics editor. This graphics tool, which runs on Windows, Linux, and Mac OS X, is well-suited to preparing graphics for LaTeX documents. The main features of Ipe are described, as well as some advanced usage. A similar editor, VRR, is also described briefly.

Jan Hlavacek is an Instructor of Mathematical Sciences at Saginaw Valley State University, Michigan. He is a long-time LaTeX and Lyx user. You can contact him at jhlavace@svsu.edu

- PDF version of paper
- An Ipe graphic and LaTeX slide presentation
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2006, No. 2
Article revision 2006/05/12

IPE — a graphics editor with LATEX integration

Jan Hlavacek

Email jhlavace@svsu.edu
Website http://www.svsu.edu/~jhlavace/
Address Saginaw Valley State University
7400 Bay Road
University Center, MI 48710
USA

Abstract The article talks about author's experiences with the IPE graphics editor. The main features of IPE are described, as well as some advanced usage. In the section 6, a similar editor, VRR, is mentioned.

1 Introduction

IPE is a graphics editor which runs on Windows, Linux, and Mac OS X and is well-suited to preparing graphics for LATEX documents. In this article I will describe IPE's features and usage, with several examples.

I first started using IPE sometime in the mid-1990's, when both my wife and I were working on our dissertations at Ohio State University. My dissertation

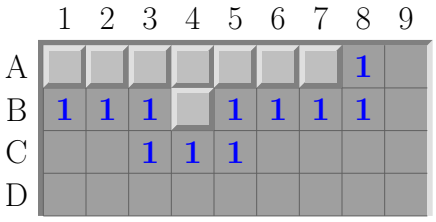


Figure 1: An IPE graphic made for a recent presentation

was in complex analysis and only included three pictures; however my wife's dissertation was in graph theory, and included about 200 illustrations. We looked for a program that would make it easy to create these pictures and to include them in a LaTeX document. In time we stumbled upon Ipe and quickly found out that it completely satisfied our needs.

The name Ipe is an acronym for Integrated Picture Environment, referring to the clever file format the editor used. To solve the problem of integrating pictures with LaTeX text and formulas, Ipe saved all figures in a file that was at the same time a valid LaTeX file and a valid Postscript file. The LaTeX part contained a `picture` environment with all the text and math positioned in proper places. It also contained code for including the file again, this time as a Postscript illustration with the rest of the drawing. You included the file as a regular LaTeX file, and it took care of everything else. Using Ipe including and handling pictures turned out to be very easy.

Another feature of Ipe we appreciated perhaps even more than the ingenious file format was Ipe's snapping modes. In addition to your regular mouse cursor, Ipe provided another cursor[1] which snapped to various objects in the drawing. In addition to the usual snapping to a grid, you could make it snap to lines and curves, vertices of polygons, intersections of curves, and on top of that there were two different angular snapping modes. All this made Ipe very powerful editor for creating mathematical and scientific illustrations.

Several years later I tried to use Ipe on my computer at home which ran the Linux operating system. I was able to compile Ipe but it simply refused to run. It seemed to have some issue with the new version of the X Server, and it kept crashing right after starting. I was unable to find any information about this problem anywhere on the Internet, and for some reason (I don't remember why) I was not able to locate or reach the author of the program. I gave up, converted all my old Ipe drawings to Postscript, and started learning METAPOST.

Come the fall of 2004, I decided to upgrade my Debian Linux system from the rather aged "stable" distribution to the "testing" distribution. When paging through a long list of new packages, my eye caught the name "ipe". Version 6.0pre22, it said. At first I was skeptical: a different program with the same name, I was thinking — happens all the time. Then I looked at the description:

1. This cursor is called *fifi*, after a dog from a popular computer game that runs around your feet in a way similar to the behaviour of this secondary cursor.

> Drawing editor for creating figures in PDF or PS formats. Ipe supports making small figures for inclusion into LaTeX documents as well as making multi-page PDF presentations ...

There was more and it all sounded sort of like the old Ipe, except the part about multi-page presentations. There was also a URL: ipe.compgeom.org.

I installed the package and visited the web page. It turned out that the author, Otfried Cheong, had completely rewritten Ipe, getting rid of old legacy code and using the newly available Qt toolkit. This meant not only that Ipe was easier to use than before, but also that, thanks to the cross platform Qt toolkit, it was now available for Unix and its clones, including Mac OS X, *and* Windows. There was another important change: the old ingenious file format that Ipe was named after was gone, replaced by a new XML-based format. However, in addition to XML, Ipe can also save files in PDF and encapsulated Postscript formats.[2] Ipe, like many modern tools, understands Unicode. There were some other changes, namely a new multiple-page presentation mode, which I will describe later.

2 Main Features of Ipe

Figure 2 shows the initial Ipe window. This screenshot was taken on a Windows system, but the Linux version looks very similar. Most of the window is occupied by a yellow drawing canvas (you can change the color of the canvas to white if you prefer that) with a visible grid. On top of the window is the usual menu bar and several toolbars.

2. In fact, Ipe can be configured so that every time you save a file in either the PDF or encapsulated Postscript format, the other format will also be saved automatically, which makes it easy for a TEX document and its figures to be processed by both TEX and pdfTEX [see the Ask Nelly column in this issue for a discussion of various kinds of image files TEX and pdfTEX can process. — Ed.].

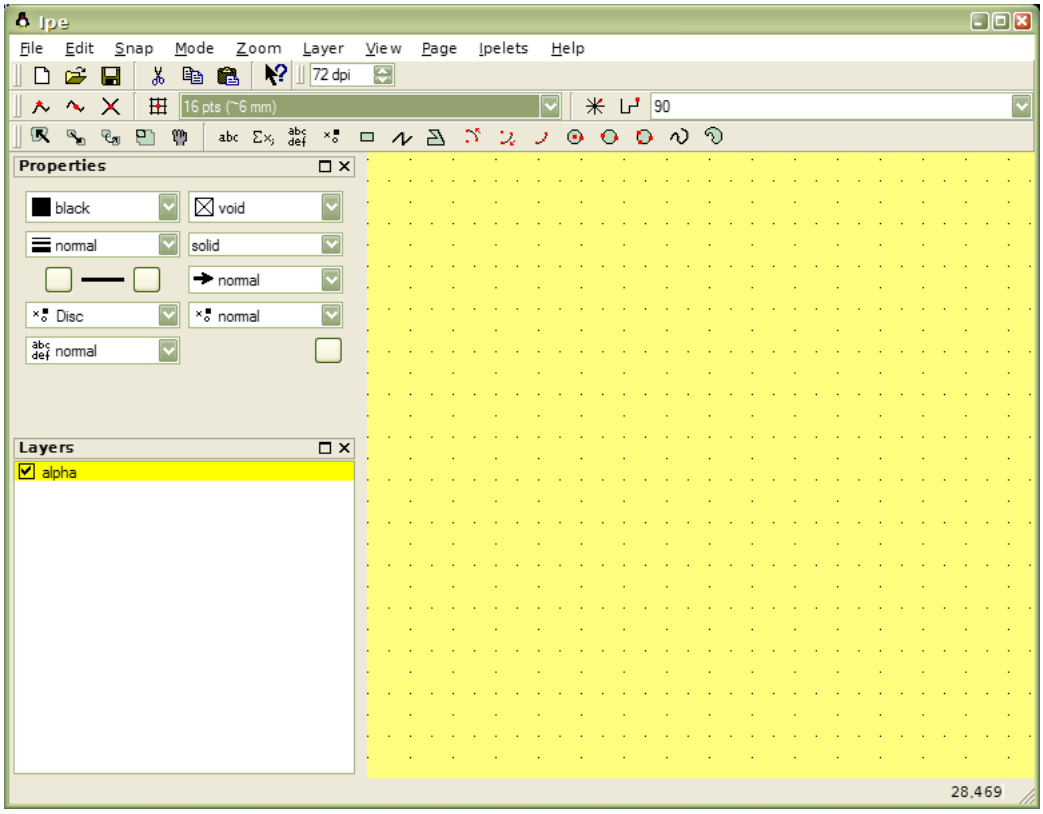
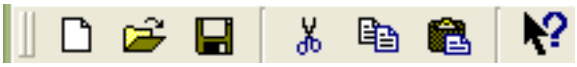
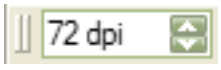


Figure 2: A new IPE window



First is the **file** toolbar, with the usual icons for saving files, opening new files, cutting and pasting, and so forth. What you do not find there is the "print" icon — you cannot print directly from IPE.

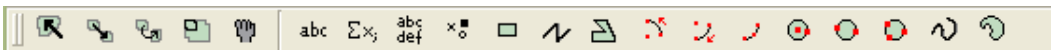


Next is the **resolution** toolbar. You can see and change the current resolution of the canvas (changing the resolution will zoom in or out of your drawing). An easy way to adjust the resolution is with your mouse scroll wheel; there are also several keyboard shortcuts to change the resolution.

Probably the most useful toolbar contains **snapping** tools. Here you can change your snapping mode, change grid size, and change the angle for angular snapping. The available snapping tools are:

- Vertex snapping (shortcut 'F4') — your cursor will snap to the vertices of polygons and polylines, marks, and centers of circles.
- Boundary snapping (shortcut 'F5') — snaps to all lines and curves in your drawing.
- Intersection snapping ('F6') — snaps to intersections of straight lines (currently, IPE does not snap to intersections involving circles, arcs or splines).
- Grid snapping ('F7') — on and off (independently of snapping, you can toggle grid visibility with 'F12').
- Absolute angular snapping ('F8') — this is somewhat harder to explain. According to the manual [1], *When angular snapping is enabled, the mouse position is restricted to lie on a set of lines through the origin of your current axis system. The lines are the lines whose angle with the base direction is an integer multiple of the snap angle. The snap angle can be set in the second box in the Snap toolbar.* The origin of the axis system can be set using the 'F1' key, and the direction using 'F2' key. You can also set the axis system to be parallel to an existing line in you drawing by pointing at the line and pressing 'F3'. The origin of the axis system is also used as a center of scaling and rotation.
- Automatic (relative) angular snapping ('F9') — this one is used only when creating a polygon or a polyline. It works in the same way as absolute angular snapping, except that instead of using the origin of the axis system, it uses the last created vertex of your polygon. That way you can easily create a polygon or polyline in which all angles are multiples of the snap angle.

You can use any number of snapping tools together.



The last visible toolbar is the **objects** toolbar. Use this to select your drawing tool. Available tools are *select, move, rotate, scale. pan, text, formula, paragraph, marks, rectangles, polylines, polygons, three different ways of drawing arcs, three different ways of drawing circles, splines,* and *splinegons*. What most of these tools are for should be fairly obvious to anybody who had previously used any drawing program. However, here are some notes on the use of these tools in this particular drawing program:

- When you select the **pan** tool, you can drag your canvas around to change which part of your drawing you can see. I personally prefer to use the 'x' keyboard shortcut which centers the canvas window around the point your cursor is currently over. My approach is to use the scroll wheel to zoom out until I see the whole drawing, point at the part I want to work on, press 'x' which will center that part, and zoom in using the scroll wheel again.
- There are three text insertion modes.
  The first mode is the simple **Text label** mode. You can use LaTeX commands in your labels, including math mode, color changes, different TeX fonts, etc. You can load packages and set up definitions in the LaTeX preamble, which you can edit by going to the "Edit" menu and choosing "Document properties". When selecting a font, be aware that Ipe can only use Type 1 or TrueType fonts; Ipe cannot handle bitmapped Type 3 fonts. You can include accented characters or characters from non-latin scripts transparently using UniCode. The graphical user environment of Ipe handles UniCode transparently. Look in the Ipe manual ([1]) for instructions how to set up your document preamble for inclusion of UniCode characters into LaTeX.
  The second text insertion mode is called **Mathematical symbols**. This is essentially the same as simple-text-label, except that it is processed by LaTeX in the math mode. It is actually equivalent to a text label in which everything is enclosed by a pair of dollar signs.
  The third text insertion mode is called **Paragraphs**. This will insert a LaTeX minipage environment into your picture. You can therefore use various LaTeX paragraph environments, such as 'center', 'itemize', etc. The 'F10' key will insert a minipage that spans the entire width of your page.

– The text will first show the way you typed it, including LaTeX commands and special symbols. When you press 'Ctrl-L' or when you save your file, Ipe will run LaTeX and display the text the way it will look as a finished picture.
– The 'Ctrl-E' key shortcut will allow you to modify the currently selected object – polyline, polygon, spline, circle, or text.

2.1 The Context Menu

One important thing you cannot really see in the default Ipe window is the context menu. To get to this menu, you press the 'Ctrl' key and right-click on any object drawn on your canvas. In the context menu you can change various properties of the object such as line thickness, color, text alignment, etc., depending on the type of object.

Here you can also find the "scissors" tool, which you can use to cut an object into two separate objects. Currently you can cut polylines, polygons, circular arcs and circles.

3 Layers, Views and Pages

Another nice capability of Ipe is its ability to create multi-page PDF documents. By selecting "Insert page" from the "Page" menu (figure 3), you add another page to your document.

Not only can you have multiple pages, but each page can have several "layers" and "views". It works like this: You arrange your drawing objects on your page into so-called "layers". Everything you draw will belong to the layer which is currently active, and you also can move objects between layers. Then you define several "views", each showing only some of the layers. That way you can build each page interactively, showing and hiding layers as you switch from view to view. The "layers" pane (figure 4) shows you the list of all layers on the current page, clearly marking the visible layers and indicating which layer is active. By right-clicking on a layer name, you can access menu that lets you change its properties.

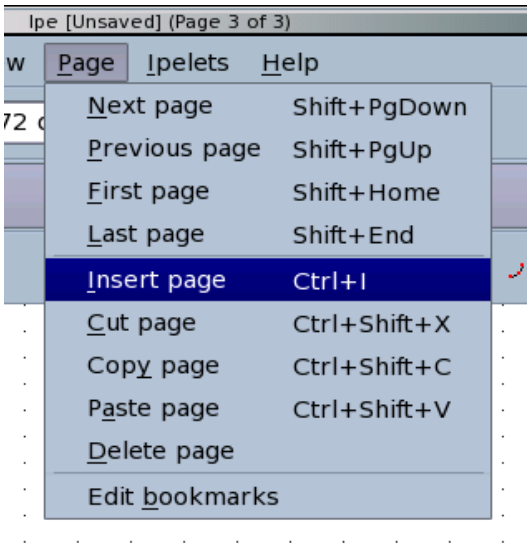


Figure 3: The "Page" menu.

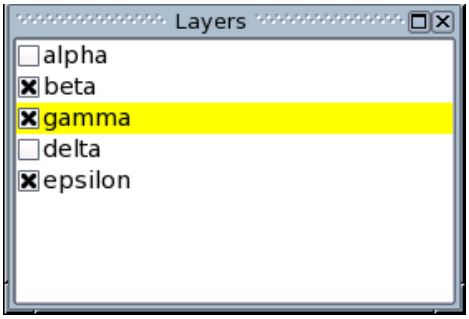


Figure 4: The "Layers" pane with 5 layers, three of them visible in the current view, the "gamma" layer active.

The above capabilities, together with a stylesheet specifying a screen sized page and a large font, allow you to create presentations with Ipe.

4 Advanced Usage

4.1 Other Graphics Formats

By default, Ipe saves all drawings in a special subset of the PDF standard. You can also save them in a subset of encapsulated Postscript. Finally, you can save drawings in a special XML-based format which has the advantage of being easily editable with a text editor or other XML processing tools. at the moment You can insert a raster format images (such as JPEG or PNG files) into your drawing by choosing "Insert Image" from the "File" menu. These images will be shown on your Ipe canvas unless they are too large. In preferences, under "On-screen bitmap resolution", you can set the dimensions of the largest image that will be displayed. Set this to a smaller value if your computer is slow or does not have much memory. Images that are larger than this value will not be displayed by Ipe, but they will show up in your resulting PDF or EPS file.

You cannot insert other Postscript or PDF files into Ipe drawings. Neither can you use Ipe to edit an arbitrary EPS or PDF file. Only files created by Ipe can be edited. However, Ipe comes with a special command line tool, `pdftoipe`. This tool converts an arbitrary PDF file into the Ipe XML format. Unfortunately, the conversion does not handle text very well; it does work quite well with graphics.

Figure 5 shows a graph created using METAPOST with the excellent macros developed by Jean-Michel Sarlat. I converted a METAPOST file to PDF using Hans Hagen's `mptopdf`, and I converted the result of that to the Ipe format using `pdftoipe`. The resulting figure needed very little editing in Ipe. The result of that process is shown in figure 6. Figure 7 shows Ipe (this time running on Linux) editing the file.

4.2 Stylesheets

Many properties of your picture, as well as Ipe's user interface, are affected by stylesheets. By using different stylesheets, you can change such properties as font sizes, line thickness, colors, page size, but also page background and things

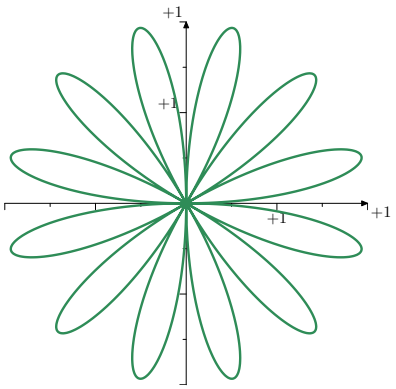


Figure 5: Graph created in METAPOST

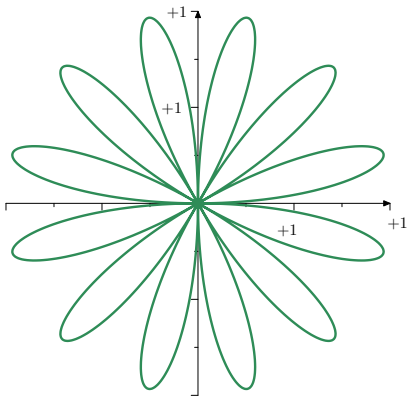


Figure 6: Graph processed by `pdftoipe`

like a LaTeX preamble for your document. You can, for example, have a special stylesheet for presentations in which all fonts are larger, all lines are thicker, the page is the size of the screen, which has a fancy gradient background, and uses Lucida Bright or Arev fonts.

The stylesheets also affect the user interface. In Ipe you can switch between so–called **absolute attributes** and symbolic attributes. When absolute attributes are active, you choose colors using a color selection dialog with RGB values, and things like line thickness, size of arrows, fonts etc. are specified in point units. In symbolic mode, which is the default, you select colors, line thicknesses etc. from a list of available names, such as red, normal, fat, large, etc. These names are defined in the stylesheets. By loading different stylesheets, you can add new options for colors or sizes, as well as change the meaning of the already present options.

4.3 Cutting and Pasting

As with other vector graphics editors, in Ipe you can cut or copy selected objects from your drawing and paste them into another file or another page of the same file. But you can do more than that: Ipe objects are stored in the clipboard using their XML representation. That means you can cut an object from Ipe, paste it into a text editor window, edit it, cut it from the text editor, and paste it back into Ipe. You can also write some of your drawing elements directly in XML in your

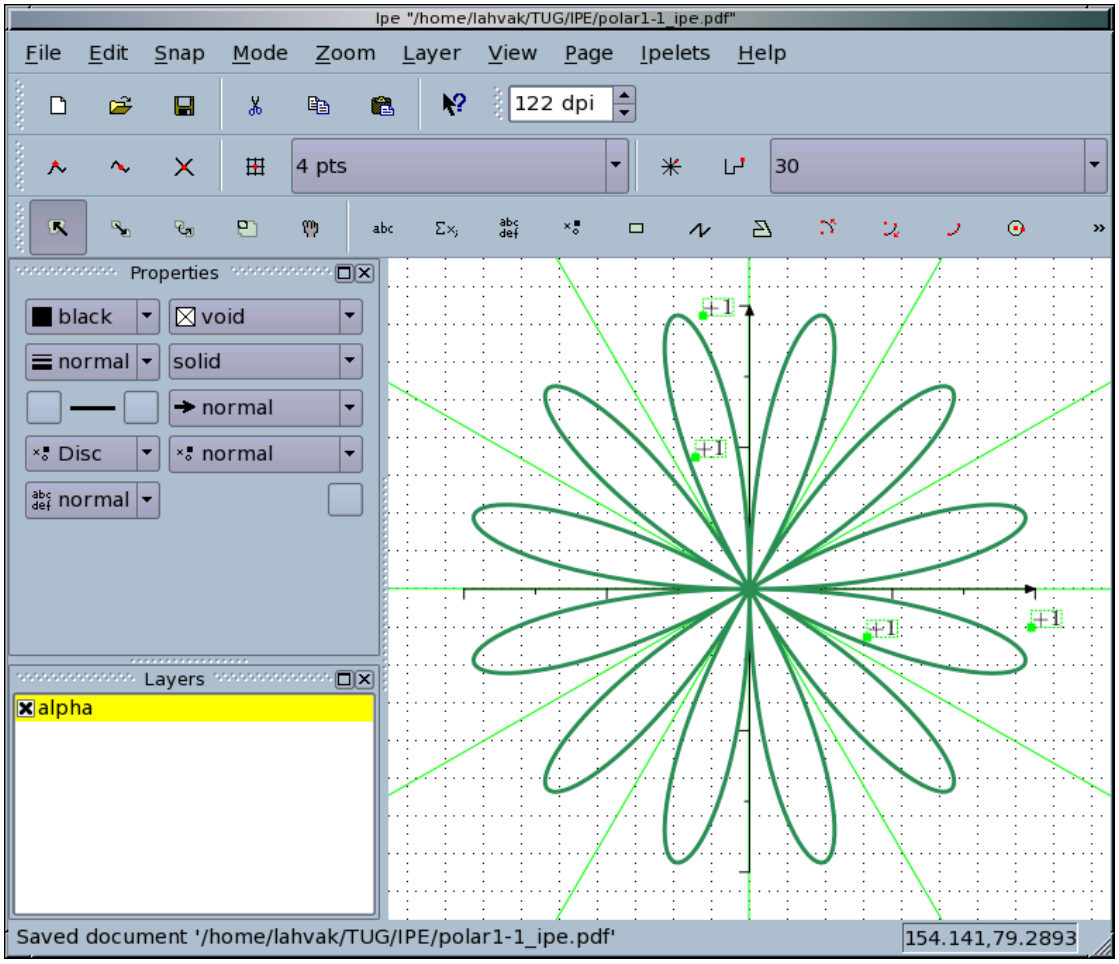


Figure 7: Ipe on Linux, editing a file converted from METAPOST using `mptopdf` and `pdftoipe`.

text editor, and cut-and-paste them into an IPE window. This way you can create objects with precise coordinates, exactly the way you need them, without having to carefully select your grid size and carefully click at exactly the right place on the canvas.

For example, pasting

```
<ipeselection>
<path stroke=".3 .5 1" pen="normal">
100 100 m
300 100 l
300 300 l
100 300 l
h
</path>
<path stroke="black" pen="ultrafat">
100 0 0 100 200 200 e
</path>
<text stroke="black" pos="305 305" type="label"
halign="left" valign="bottom" size="Large">
$\alpha_1$
</text>
</ipeselection>
```

into an IPE window will create figure 8. The exact syntax can be found in the IPE manual [1].

4.4 Ipelets

The functionality of IPE can be extended using so called "ipelets". These are small dynamically loaded pieces of code that interact with IPE and can modify your drawing in some way. The IPE system comes with a number of ipelets pre-installed; for example, there are ipelets for aligning objects in various ways, for precise stretch and rotation, and so forth. You can access them through the "ipelets" menu (figure 9).

The manual explains how to write your own ipelets using C++.

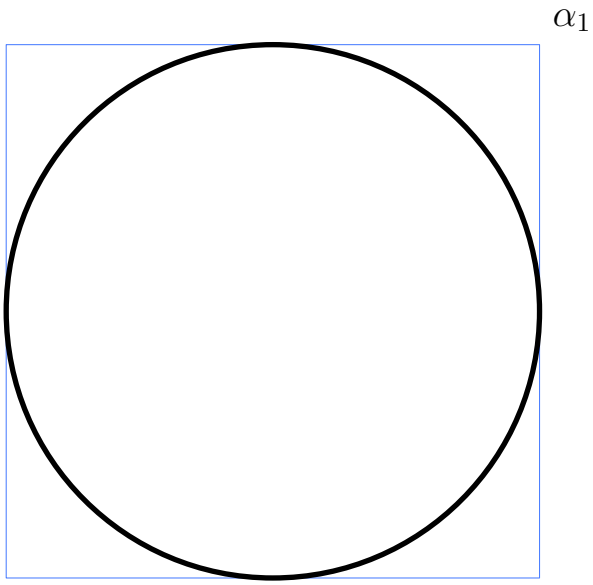


Figure 8: A precise figure created by pasting XML code into Ipe window.

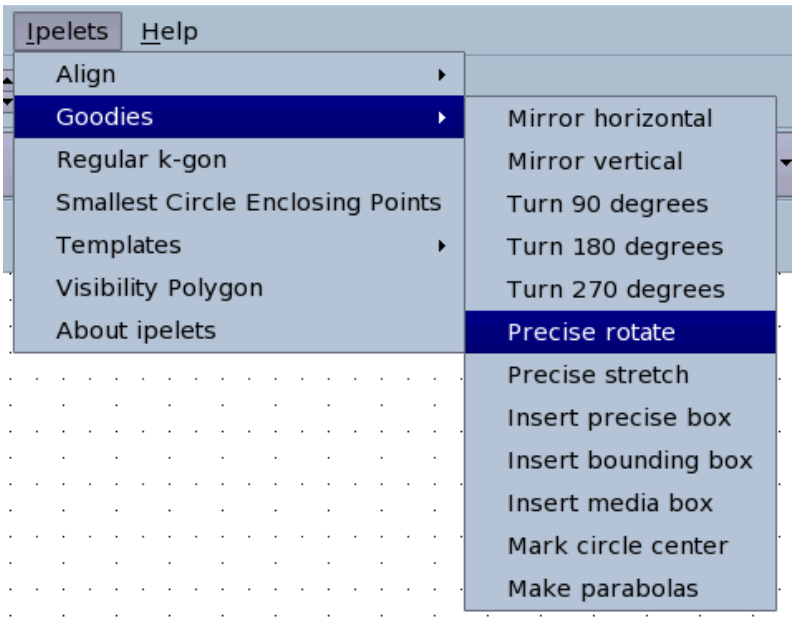


Figure 9: The "ipelets" menu

5 Word of Caution

The current version of IPE described in this article is Ipe 6.0 **preview** 26. It is a great piece of software, and it makes my life much easier every time I need to create illustrations to include in LaTeX documents. Other people I know even use IPE to create drawings to include in Microsoft Word and other documents. It is easy to use and powerful. However, as of this writing, it hasn't been officially released. It does have some bugs, and it has been known to break occasionally under high-stress testing. Usual procedures, such as saving your work frequently, should be followed when working on complex projects.

6 The VRR Editor

While working on this article, I ran across another editor that seems to have a lot in common with IPE in design and basic philosophy. It is the VRR editor, available from `http://atrey.karlin.mff.cuni.cz/projekty/vrr/` [2]. It has snapping capabilities similar to IPE's capabilities, it uses TeX similarly to IPE, it can export drawings in PDF or EPS, and so forth. It has several features not present in IPE, namely, it can snap to intersections of splines and circles, and it can also create geometric dependencies in a way similar to some popular dynamic geometry software, such as Geometer's Sketchpad, KIG, and others. In IPE, if you snap an object to, for example, an intersection of two lines and then you move one of the lines so that the intersection changes, the snapped object is not going to move and consequently will no longer be snapped to the intersection. This is how most vector graphics editors with this kind of snapping work. In VRR, you can specify a geometric dependency so that, for example, if you snap an object to an intersection and the intersecting objects move and the intersection changes, the snapped object also moves and stays snapped to the intersection. The VRR editor can also import and export some additional file formats, for example SVG. For more information on VRR, please refer to the manual [2].

One major shortcoming of the VRR editor compared to IPE is that it is not cross-platform — it seems to run only on Linux. One thing I really like about IPE is that it works exactly the same way on my Linux system at home as it does on WINDOWS on my office laptop.

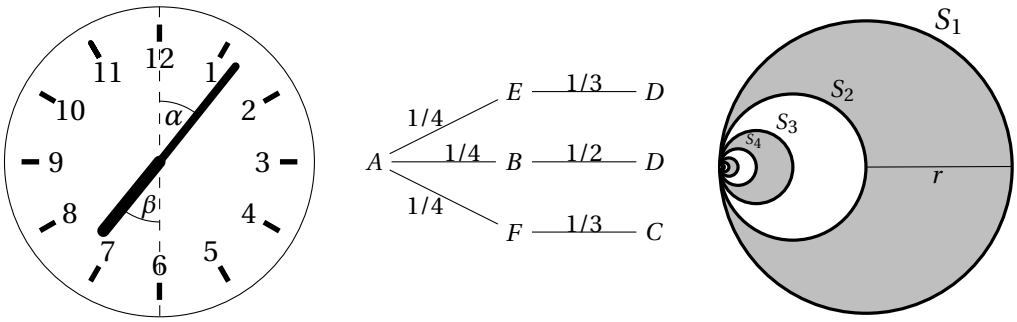


Figure 10: Three simple examples of figures created with Ipe for inclusion into a LaTeX document.

7 Conclusion

This article has presented the cross-platform Ipe drawing editor; its main features are tight integration with LaTeX, sophisticated snapping modes, extensibility via ipelets, easy yet flexible user interface, customizable with stylesheets, and ability to export commonly used PDF and EPS file formats. I find myself choosing Ipe over METAPOST when creating figures in which elements are to be laid out according to a more visual scheme, as opposed to a more programmatic or logical scheme that is easier to achieve in METAPOST. However, thanks to snapping modes and the ability to directly edit the XML code, even very precise highly geometric figures can easily be created with Ipe. (Figure 10 shows few simple examples from one of my recent LaTeX documents.)

Thanks to its system of pages, layers and views, Ipe can also be very helpful when creating presentation. Two very simple examples of such presentations are accompanying this article.

References

[1] Otfried Cheong The Ipe Manual http://ipe.compgeom.org/manual.pdf

[2] The Vrr Team The Vrr User's Manual http://atrey.karlin.mff.cuni.cz/projekty/vrr/doc/man/manual/

THE PracTEX Journal

Introduction to "A short example of how to use LaTeX for scientific reports"

Stephen J. Eglen

Abstract

This short article summarises my reasons for writing a short \LaTeX\ document to act as a template for scientific reports. This document was used as a basis of practical session with Masters students who wished to learn to use \LaTeX \ for their reports. The outcome was quite successful in that many more students are now using \LaTeX for their reports, often using the document as a template.

The introductory document is attached below as intro.zip. Updates to this material will be available from here.

Stephen Eglen is a lecturer at the University of Cambridge in the field of Computational Biology. A regular user of LaTeX since 1990, he can be reached at S.J.Eglen@damtp.cam.ac.uk.

- PDF version of paper
- A short example of how to use LaTeX for scientific reports
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2006, No. 2
Article revision 2006/05/01

Introduction to "A short example of how to use LaTeX for scientific reports"

Stephen J Eglen

Email S.J.Eglen@damtp.cam.ac.uk
Website http://www.damtp.cam.ac.uk/user/eglen
Address Department of Applied Mathematics and Theoretical Physics,
University of Cambridge, Wilberforce Road, Cambridge CB3 0WA U.K.

I lecture on a Master's programme that requires the students to write many scientific reports during their course. A typical report needs figures, tables, and bibliography, all of which LaTeX is well-suited for. Most students have heard of LaTeX from their peers or from their lecturers. However, due to the demands of the course, few students take time out specifically to learn how to use LaTeX, especially when deadlines are close. In response to a few suggestions from students, I therefore decided to put on a short lab session introducing the features of LaTeX to new users.

When preparing the lab, I was keen to avoid writing "yet another introduction to LaTeX" as other people have done that already, to good effect (see Section 7 of the intro.tex article for two guides). Instead, I started looking for a brief overview of how to write reports in LaTeX. One close match was the article by Peter Flom ("LaTeX for academics and researchers who (think they) don't need it", The PracTEX Journal, 2005, number 4). However, that article missed some aspects (BibTeX; graphics inclusion) that I knew students would require. Furthermore, I wanted the students to have all the source files available along with the relevant UNIX commands, so that they could easily modify the document. From these requirements, I then wrote the attached document, "A short example of how to use LaTeX for scientific reports".

The practical session was run in a lab where each student sat at a Linux workstation with access to the source files. They were given handouts of both the

source file (intro.tex) and the output (intro.pdf). I gave a short introduction, encouraging students to read the source file and compare it with the output. The students were left to work through the document, editing it and recompiling to see the effects of their changes. I was available to answer questions and students also exchanged helpful comments during the session. One useful suggestion during the session was that students wanted to type in their own minimal yet complete example. I wrote the following short example on the board for them to type in:

```
\documentclass{article}
\begin{document}
Hello world.
\end{document}
```

Students found this simple example to be useful as a way of starting from scratch. By the end of the 90 minute session, most students then felt comfortable with the core ideas behind LaTeX. More long term, I am happy to say that the number of assignments being submitted in LaTeX has increased, and I've heard many comments about how much they like the output from LaTeX, compared to their word processors. I therefore hope to repeat the course in the next academic year using this material. By making this article available, I hope that it might be useful to similar "getting started" classes.

THE PracTEX Journal

My Experience with Learning and Teaching LaTeX

D.V.L.K.D.P. Venugopal

Abstract

The present article deals with the author's experience in learning \LaTeX independently and disseminating the knowledge acquired through one week structured course to research scholars in a University system.

Mr. Venugopal is a stenographer by profession. He has been using computers since the two floppy PC stage. A former lover of WordPerfect, his first encounter with TeX/LaTeX was in the year 2000. At present exploring various packages, experimenting with Omega, MetaPost and MetaFont and teaching and/or explaining advantages of TeX/LaTeX over wordprocessors are his hobbies. You can contact him at venugopal_duvvuri@rediffmail.com

- PDF version of paper
- Slides for Mr. Venu Gopal's course, in PDF format
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2006, No. 2
Article revision 2006/04/29

My Experience with Learning and Teaching LaTeX

D.V.L.K.D.P. Venugopal

Address Personal Assistant
Vice-Chancellor's Office
Banaras Hindu University
Varanasi - 221 005
India

Abstract The present article deals with the author's experience in learning LaTeX independently and disseminating the knowledge acquired through a one week structured course taught to research scholars in a University system.

1 Learning LaTeX

I learned LaTeX the hard way. Actually it was a challenge to me by my colleague in the year 2001. Exploring various software and experimenting with them is my hobby. Except some academicians nobody in our city (Varanasi) knew LaTeX and the research scholars of the Department of Mathematics used to get the LaTeX typesetting done from far off cities. During the same time a computer magazine in India (PC Quest) published an article about LaTeX and distributed MikTeX on its CD.

I started exploring TeX/LaTeX. I was not aware that a lot of documentation comes with TeX/LaTeX. So I downloaded the "The Not So Short Introduction to LaTeX2e" by Tobias Oetiker from the internet. During the same period I explored a Linux version distributed by the PC Quest magazine. As there was some problem with the MikTeX given by the PC Quest magazine, I used teTeX as available in the Linux box. I typed my experimental text in an editor, saving it with `.tex` extension, running it with `latex test.tex` and then viewing the DVI file, following the method as told in "The Not So Short Introduction to LaTeX2e". I was not aware of the concept of Integrated Development Environments (IDE's) and other editors like Emacs, Vi available in Linux from where one can compile the LaTeX document and view the output without leaving the editor.

Then I came into contact with the Indian TUG (The Indian TEX Users Group) and I joined the Indian TUG mailing list. During that period the Indian TUG was preparing to organize the International Meeting of the TUG in Trivandrum, India. As a curtain raiser they started writing LATEX Tutorials and I used to download the tutorials as and when they were posted on their site. They helped me a lot. With the help of the mailing list I also learned about the Emacs editor and the AUCTEX package.

Learning that I was experimenting with LATEX a research scholar of Mathematics approached me to typeset his article as he was asked by the Editor of the Journal to submit the manuscript typeset in LATEX. It took me nearly one month to typeset this 20 page article! This provided me with the necessary practical experience. From then onwards there was no looking back.

After working with LATEX for 3 years and experimenting with various packages, I got an opportunity to take a class for the research scholars who were attending a short training course on using computers for research. There, for the first time, I spoke about and demonstrated LATEX for about an hour.

2 First LATEX Course

Then in 2005 I got an opportunity to organize a one week course on using LATEX. Banaras Hindu University, with 3 Institute, 15 Faculties, 123 Departments and 3 interdisciplinary Schools, is an ideal place for propagating TEX/LATEX. The University attracts students from all over India and neighboring countries. In this course we restricted ourselves to the research scholars of Faculty of Science and Institute of Technology. We announced the "Short Training Course on Using LATEX" (January, 2005) and feared that we might not receive more than 10 applications. To our astonishment we received more than 80 applications. Finally we admitted 38 candidates in the course. We gave them the TEXLive CD (which I got after the TUG Conference), the LATEX Tutorials (prepared by the Indian TUG and freely downloadable from www.tug.org.in) and Formatting Information by Peter Flynn (TUGboat, Volume 23(2002), No. 2). We adopted the method of one hour lecture and demonstration, followed by one hour hands on practice. One computer was assigned for a group of two or three students.

On the first day a general introduction about TEX, its history, how to install, and the various editors available like Emacs, Vi, WinEdit and Winshell was given.

A plain document with a preamble was demonstrated along with the concepts of classes and style packages. During practice session students were encouraged to prepare a dummy article for publication purpose. The common errors like parenthesis in place of braces, splitting of command names like `documentclass` and how to find them was also explained. All the computers were MS Windows based and we used the WinShell IDE to edit the `tex` documents.

On the second day citing references using the simple `thebibliography` environment and the more advanced BibTEX and creating bibliography databases for BibTEX were demonstrated. After this demonstration the students, who felt that LATEX is a cumbersome software to learn on the first day, showed considerable interest as using BibTEX allows references to be formatted in different ways without much effort.

The third day was given to demonstrate preparation of tables etc. The packages `rotating` (using this package one can turn the table sideways — useful for wider tables), `longtable` (this is useful if the table spans more than one page), `colortab` (this is for shading the cells of tables with various colors – a fancy thing useful in slides etc. for highlighting the data), `booktabs` (on using this package the tables look more professional and beautiful), were also demonstrated.

Typesetting mathematics was demonstrated on fourth day. Though we aimed this course for the research scholars of the Department of Mathematics, unfortunately none from the department joined the course. The attendees were mainly from the School of Biotechnology, Departments of Botany, Chemistry, Geology, Geophysics, Physics, Zoology, Applied Mathematics, Applied Physics, Ceramics and Electrical Engineering.

On the fifth day we demonstrated the `minipage` environment, various boxed matters, inclusion of graphics and how to define macros for individual use. With this the course came to an end and the students were asked to prepare for a short examination on the next day.

During the first day's lecture the features of the `exam` class of Jason Alexander were explained to the students.[1] They were also cautioned that in order to show the power of this package a multiple choice question (MCQ) type examination would be conducted on the last day and the paper would be typeset using this package. So on the sixth day an examination for 25 marks comprising 25 ques-

1. There are two `exam` classes. We are discussing the `exam` class in the `examdesign` folder on CTAN.

tions (Multiple choice – 10; True/False – 5; Fill in the blanks – 5 and Matching – 5) was conducted. The maximum and minimum marks obtained by the students are 20 and 11 respectively. After the examination was over correct answers to the questions were explained. Some beautiful figures drawn using MetaPost and PSTricks (both taken from the internet) were also shown and the features of various packages available to prepare slideshows like `TeXpower` and `pdfslide` were also discussed.

3 Further Courses

Seeing the success of the course and also to teach those research scholars who could not be accommodated in the first training course we announced another course within a short span of two months (March, 2005). Again we received overwhelming response and keeping in view our past experience we admitted only 28 students this time. There was a small change. This time we distributed ProTEXt (a version of MikTEX). The IDE was also different. We used TEXnicCenter instead of WinShell. More or less the same course content was adopted. In order to save time and to deliver more content this time we used the screen version of the LATEX Tutorials. This time we also distributed the BibDB software of Eyal Doron with which Bibliographic databases can be easily created and maintained. As usual a short examination was conducted and the highest mark obtained was 22 and the lowest was 11.

After a gap of 6 months and by request of some research scholars from the Department of Mathematics another course[2] was organised in September, 2005. This time 18 research scholars joined and completed the course. The method followed was more or less same as the second course. In this course the packages `mhchem` (useful for typesetting chemical equations) and `biocon` (useful for typesetting biological names) were also demonstrated.

2. The slides for this course are available in PDF format from http://dw.tug.org/pracjourn/2006-2/venugopal/TeXIntroLect2.pdf

4 Other Endeavours

Introductory lectures about LaTeX were also given in the Orientation Courses conducted for Lecturers in the Academic Staff College of our University. The participants were very much excited by the lecture demonstrations. There is a need to conduct separate course(s) for the teachers.

With these endeavours we could initiate some 84 students into the use of LaTeX. We are not sure whether or not all who attended the course are using LaTeX on a daily basis. If at least 10% of them use it then also our mission is successful. Our experience is that in every batch there were at least four to five students who took LaTeX seriously. We covered only three Faculties to date. The Arts and Social Sciences faculties are not yet touched.

5 Suggestions

Proper guidance and books will certainly help interested persons learn LaTeX quickly. Both the books "LaTeX Tutorials" and "The Not So Short Introduction to LaTeX2e" lack information on IDEs and editors. "Formatting Information" has a good deal of information on subjects like IDEs and Bibliographic data managers and is useful for a starter. But to have a good command of LaTeX help of all three books mentioned above is needed. For Microsoft Windows users MikTeX or ProTeXt are better suited as they have the facility of installing packages on demand.

Acknowledgments

I would like to thank my colleague Mr. P.K. Sinha for his provocation to learn TeX, Prof. Ashok Kumar, School of Biotechnology, BHU for providing me an opportunity to organize the LaTeX courses, and Yuri Robbers and three anonymous reviewers for their guidance in improving this paper.

References

[1] Flynn, Peter. Formatting Information. TUGboat, 23(2), 2002.

[2] Oetiker, Tobias; Hyna, Irene and Schlegl, Elisabeth. The Not So Short Introduction to LaTeX2e. 2004. CTAN:/tex-archive/info/lshort

[3] Doron, Eyal. BibDB for Windows – A BibTeX Database. 1998. http://www.tcisoft.com/tcisoft/bibdb.html

[4] Indian TeX Users Group. LaTeX Tutorials – A Primer. E. Krishnan (Ed.). 2002. http://www.tug.org.in

THE PracTEX Journal

In My Opinion: LaTeX isn't for everyone but it could be for you (with responses)

Andy Roberts

Abstract

This article by Andrew Roberts appeared last summer in OSNews.com. While the merits of LaTeX Andy points out are familiar to most PracTeX Journal readers, the follow-on comments from readers pointed out some of LaTeX's (and TeX's) weaknesses. Weigh in with your comments by clicking the "Comment on this article" link below.

- Comment on this paper
- Send submission idea to editor

Anyone who has used Microsoft Word for a reasonable amount of time will recognise my very own Andy's Laws on Word:

1. Likelihood of a crash is directly proportional to the importance of a document.
2. Likelihood of a crash is inversely proportional to the time left before its deadline.
3. Likelihood of a crash is directly proportional to the duration since you last saved.
4. Likelihood of you throwing your computer out of the window is directly proportional to the number of times Clippy pops up.
5. That's enough laws for now...

In all seriousness, I've written many words in large documents using Microsoft Word.

Nowadays, I can use OpenOffice because it's come a long way and really is a decent product (the current v2 beta is very good). However, ever since my never-ending woes with Word during my degree, when I started my PhD, I decided to go and try out LaTeX. Actually, that's not quite true, I wanted to submit a paper to a journal and it only accepted LaTeX documents. The deadline was the same day as I found out about the call for papers. I jumped straight into the deep-end with both feet. Needless to say, I had a hard time of it and wasn't LaTeX's best fan that day. (Lesson: don't try to learn something new in a rush!)

Undeterred, I stuck with LaTeX and realised that it wasn't so hard after all. There was a learning curve, but for the typical documents that I often wrote, there was very little to learn. I'm very glad I persevered because I wouldn't want to use any thing else for my papers/reports any more. I'm not the only one who's glad to move away from the WYSIWYG world [http://www.ecn.wfu.edu/~cottrell/wp.html]. This article will not be a tutorial for how to use LaTeX, instead an overview of its benefits and why I think it trumps what word processors have to offer.

What is LaTeX?

In 1978, Donald Knuth [http://en.wikipedia.org/wiki/Donald_Knuth] - arguably one of the most famous and well respected computer scientists - embarked on a project to create a typesetting system, called Tex (pronounced 'tech'), after being disappointed with the quality of his acclaimed *The Art of Programming* series. Around 10 years later, he froze the language after originally anticipating spending a single year! Tex gave extremely fine-grained control of document layout. However, the vast flexibility meant it was complex, so by the mid-80s Leslie Lamport created a set of macros that abstracted away many of the complexities. This allowed for a simpler approach for creating documents, where content and style were separate. This extension became LaTeX (pronounced 'lay-tech').

LaTeX is essentially a markup language. Content is written in plain text and can be annotated with various 'commands' that describe how certain elements should be displayed. The LaTeX interpreter reads in a LaTeX marked-up file, renders the content into a document and dumps it a new file. Therefore, it's not an interactive system that is the de-facto method for document creation nowadays.

Separation of content and style

Not the most obvious advantage, possibly because a lot of Word users don't understand why this so beneficial. When producing your LaTeX document, you are concentrating on the content itself. You introduce structure explicitly by telling LaTeX when a new section begins, for example, but you don't then faff around trying to decide how the section headers should *look*. That's done later.

This is opposed to the average Word user, who will immediately highlight a given section header and apply formatting to it: maybe a larger font, maybe underline, etc. The point is that this will then have to be applied to every header manually. LaTeX is better as it uses a *document style*. This defines how different elements within your document should look (like Cascading Style Sheets defining styles in HTML pages). If you fancy a change, you only change the style definitions once, then the presentation of the document will be updated

automatically. This also ensures a consistent looking document (you wouldn't believe how many stylistically inconsistent Word docs I've read!)

Word does in fact have a similar *Styles* feature. However, because it's optional, people don't often know it exists. LaTeX forces you to declare the document semantics (this is a Good Thing!), which is why you can rely on it to produce a consistent looking document.

Portability

LaTeX portability comes in multiple ways:

1. An actual LaTeX file is merely a text file, which is just about the most portable format in computing.
2. The LaTeX system that processes the text file and produces the finished document has been implemented on just about every mainstream platform you care to mention.
3. The default output file format for LaTeX is DVI [http://en.wikipedia.org/wiki/DVI_%28TeX%29] (which stands for *device independent*). This was around well before PDF was dreamed up and the high quality files can be viewed via software viewers or printed out. DVI is an open standard, so once again, readers are extremely portable and exist on most operating systems. Admittedly, DVI is hardly ubiquitous and nowadays it's often bypassed in favour for PDF (or it's very simple to convert to other formats like PS or HTML)

Flexibility

You can get LaTeX to do just about anything you can think of! Over the years, an overwhelming selection of packages to extend its potential and macros that can simplify complex tasks have come into being, most of which are freely available on CTAN. For example, LaTeX's main users are within academia and research institutions and they benefit hugely thanks to the Bibtex package that provides bibliography management - I pity my Word-using colleagues who suffer by actually manually word-processing their bibliographies (unless they've shelled out for a program like Endnote). There are other crazy packages that you can install which allow you to typeset music scores, chessboards and cross-words! CTAN [http://www.ctan.org] is the main repository of these resources. Most are well documented and as you can imagine, with LaTeX being around for so long, the number of extensions is vast. The chances are, if you're struggling to do a task, someone will have undoubtedly written a package to solve it easily!

Control

Even with simple documents, you can quickly become frustrated by Word's rather unintelligent interference. The hours that are wasted trying to position that image which you *know* will fit at the bottom of the page, but Word refuses to put it there! How many can relate to this experience? You have your 30 page document with text, tables and images. You just spent the evening getting it formatted nicely - all your figures in the right place and then you notice that one of your paragraphs isn't clear enough. You add **one** sentence, which then pushes an image on to the next page, leaving a massive gap at the bottom of that page where your image once was. This then daisy-chains down, knocking other tables and images out of

place all the way to the end of your document! It's a real laugh. Fortunately, LaTeX is much more clever in this respect and positions your images and tables with a lot of common sense. So, if you want your image to appear at the bottom of a given page, it'll stay there!

Whilst LaTeX makes decent typesetting decisions for you, if you want to, you can have total control over the presentation of your document.

Quality

It's difficult to disagree that the output from LaTeX is far superior to what Word can produce. This is emphasised greatest when it comes to documents with high mathematical content, which is a major strength for LaTeX. It also has much better kerning, hyphenation and justification algorithms that simply make the output far more professional than what any word processor. Its algorithms for laying out text are more sophisticated and extremely fine-grained. For example, the accuracy is so high because it uses a measurement known as a *scaled point* which translates as 100th of the wavelength of natural light!

LaTeX works with the concept of *niceness* (well, I suppose technically it's *badness* - which it works to minimise). LaTeX has a large set of metrics that it evaluates against when generating your document. It experiments with various permutations of parameters and determines the one which gives the "nicest" output. It can take the time to do this because it isn't interactive. Word processors don't have the computational resources available (yet) to carry out the equivalent calculations and still remain interactive. Also, many people forget that typesetting is actually a professional skill - people train for years to learn how to layout publications. Yet, as soon as you open a word processor, you go about committing typesetting sins [http://www.poynton.com/notes/typesetting/index.html] all the way. Typesetters know for example that its easier to read sentences that are approximately 66 characters wide. Have a look in your books and count the letters! Also, why do newspapers and magazines have narrow columns? But, the default layout of a word processor gives an average of 100 characters per line. I suppose many people don't mind, but you would notice if you read a lot of large documents.

A quick example. I took a document that I had used previously to demonstrate document structure in LaTeX. I used the same text and loaded it into Word and applied the equivalent styles. I've used default settings throughout. Word didn't have a style for abstracts, so I put the title in bold. View the LaTeX output to the Word output. The styles that Word uses aren't great. You could manipulate the default styles in Word to make it look more reasonable, but I've never been bothered because even if I could get it to match LaTeX stylistically, I still have to *use* Word, which I'd rather avoid!

LaTeX has been used regularly typeset entire books. Word processors simply aren't good enough for that job - they are used by the authors to write the content and these files are then imported into professional typesetting software. Ok, that's not strictly true - you *could* typeset a book in Word, just like you *could* drive a car with your feet - it's not a good idea though!

Output

As mentioned, the default output is a DVI file. DVI was a clever little standard but unfortunately didn't take-off. It takes little effort to convert your document into a Postscript or

PDF file (in fact, you can just use the 'pdflatex' command instead of normal 'latex' if you only ever want to create PDFs). There's no need to buy additional software such as Adobe Acrobat like you need to do to convert a Word document into PDF. (At least OpenOffice has its 'Export to PDF' functionality!)

Scalability

In my personal experience, using Word for documents with more than 20 pages has not been a pleasant experience. Obviously, that could be my own bad luck, but that is also the impression I've got from other users too.

With LaTeX, I've never found such problems. Additionally, you are free to split up large documents into smaller chunks and then let LaTeX combine them altogether later (like one chapter per file). It can also create tables of content, indexes and bibliographies easily, even on multi-file projects.

Stability

One of the reasons why perhaps so many people struggle with Word when creating large documents, is because it is prone to crashes. 'Document recovery' is now a high ranking *feature* of Word. I'm sure people would prefer if MS would just make their software more stable! (NB stability issues are not necessarily generalisable, so I'm speaking from personal experience, and of my friends and colleagues - I do not know of a single user who hasn't lost work to Word, but that's not to say that such people don't exist.)

Because LaTeX is so mature - and developed by extremely clever programmers - bugs are negligible [http://en.wikipedia.org/wiki/TeX#Quality]. And even if it were buggy, then there is no risk of you ever losing your original source text. Where as with Word, almost any tool within its integrated environment is capable of corrupting your file if it causes a crash.

Oh, you don't need to worry about macro viruses either!

Cost

Well, this is one area where LaTeX wins hands down, since it is **free**! As with most open source software, the phrase "*you get what you pay for*" doesn't hold true. You get an extremely mature system, that is still years ahead of its competition.

What about spell checking?

It's a good point. This is not a deficiency of LaTeX, because it just processes the words you give it. However, within your text-editor, you do not get fancy lines highlighting your spelling errors or bad grammar as you type, like you get with Word, yet it's a feature users have come to expect when writing documents.

For starters, I do not really care for a grammar checker and anyone who actually relies on it when using Word would be better off buying a book (or looking at writing style guides [http://www.comp.leeds.ac.uk/andyr/misc/index.html]) than taking the useless advice it provides.

Secondly, the 'auto-correct' feature - whilst looking like a good idea - is not beneficial in the long run. Sure, it corrects the common typos that we all make. However, the problem in my opinion is that it means we don't learn from our mistakes, e.g., you will continue to type 'teh' instead of 'the' because Word will sort it out for you. Having said that, if that's your thing, then you can easily configure any decent text editor to perform the same task. (You could, if you really wanted to, use your favourite word processor as your text editor - but then you back to square one on the stability issue.)

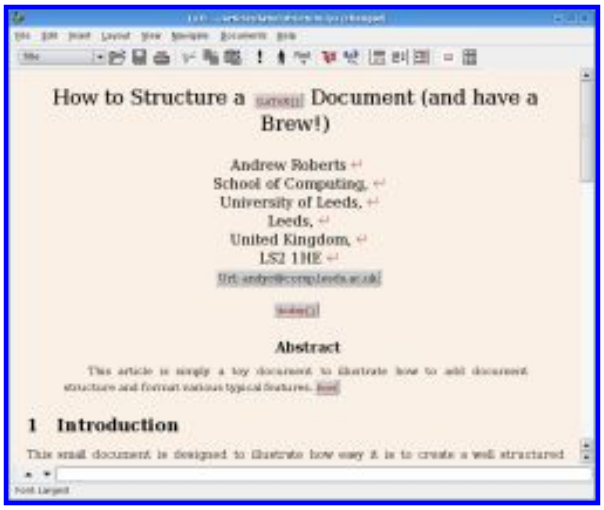
And so on to spelling. The great thing here is that you have a choice! Aspell and Ispell are the most popular spell checkers I know of (both open source). These will check any text file you care to feed it and you can easily configure a decent editor to integrate its functionality from within the editor itself. How to get your text editor to utilise these programs is obviously dependent on your editor of choice. Some, like Kate, interface external spell-checking programs without any effort. I personally use (g)vim which can be configured [http://www.comp.leeds.ac.uk/andyr/misc/vim/spell.html] to use spell-checkers like Ispell.

C'mon, be fair!

Ok, I am obviously biased here. However, I am someone who uses both systems. It's perhaps not really fair to compare LaTeX and Word, because they are different types of system, which are suited to different jobs. However, for as long as people are using Word within academia and research institutions, I feel I should enlighten them and let them know what they are missing out on.

Sure, Word can be extended using its in-built scripting language. It also has document management features to help with large documents. As already mentioned, it has styles that can ensure manageable and consistent presentation. Yet very few people seem to take advantage of them. This is especially worsened by UI *improvements* that mean Word will hide features that you do not use, which makes it more difficult to remember what Word can actually do.

Word may have the advantage of a GUI which is good for beginners. It reduces the cognitive load as it's a case of recognition versus recall. If people really want a GUI, then there are ones that act as a front-end to LaTeX [http://en.wikipedia.org/wiki/LaTeX#Frontends]. It's not a WYSIWYG editor, because what you see on screen is not what you will get when you print it out. Instead, you have What-You-See-Is-What-You-Mean editors that still hold to the ideals of LaTeX by keeping content and style separate. However, they are environments that allow a more visual approach to your content, which is handy for producing complex equations, for example, but will pass your content to LaTeX for producing the final document. Lyx [http://www.lyx.org] is the best example and was originally developed by Matthias Ettrich (yep, that's right, the same guy who founded the KDE project). You can also get LaTeX editors, which are like normal text-editors, in that you see all the raw LaTeX commands, but they come with additional features that help with creating that file, like table wizards, symbol databases, etc.



The learning curve

The reason why everyone isn't using LaTeX is because you can't just load up and go, like you can with a word processor. I consider LaTeX analogous to HTML with CSS. You need to put some markup around your text before your browser knows what to do with it - and the same is true with LaTeX. Of course, nowadays, any one can knock up a webpage thanks to, er, Word, and various other visual HTML editors and as a result, they generally look crap [http://www.affordable-removals.co.uk]. So, you need to invest a bit of time in learning some basic commands, but you'll soon realise that it's very simple afterwards. Here's a LaTeX "Hello World!" as an example:

```
% hello.tex - Hello world LaTeX example

\documentclass{article}

\begin{document}

Hello World!

\end{document}
```

This that generates the following output. It wasn't that difficult, was it? To continue learning the basics, here are the best places to go:

- The Not So Short Introduction to LaTeX [pdf] [http://people.ee.ethz.ch/~oetiker/lshort/lshort.pdf]
- Getting to Grips with LaTeX [http://www.comp.leeds.ac.uk/andyr/misc/latex/]
- Text Processing Using LaTeX [http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/]

So who is LaTeX good for?

Quite simply, anyone who is writing non-trivial documents and is tired of being let down by the performance of the current crop of word processors. If you are in academia, you really ought to be using it! Anybody writing anything maths related will not find a richer and better

quality system. For example, even WikiPedia use LaTeX for rendering [http://en.wikipedia.org/wiki/Texvc] any formulas that appear on their site.

LaTeX isn't for people who are too lazy or dislike change! I personally found the investment paid off because LaTeX allows me to produce my documents at a greater pace. I know that the enterprise will not be interested as Word is so ingrained, even though their business reports would look so much nicer. Their loss! For everyone else, it's time to give it a fair try, just so that you compare and contrast, then decide which does the job best for your needs.

About the author:
Andrew Roberts [http://www.comp.leeds.ac.uk/andyr/] is a computer science graduate from the University of Leeds, UK. He remained at Leeds to study further towards a PhD in Natural Language Processing. He has been using LaTeX for three years and is the author of the Getting to Grips with LaTeX [http://www.comp.leeds.ac.uk/andyr/misc/latex/] series.

Comments from OSNews readers on Andy Roberts' article

[OSNews readers posted over 100 comments on this article, most of them pro-LaTeX. When we contacted Andy Roberts about reprinting his article in *The PracTeX Journal* he suggested we include some of the less LaTeX-friendly comments. What follows are a few of them. Let us know what you think, pro or con -Ed.]

I personally don't like LaTeX that much. It's good for writing simple papers and theses but it's absolutely ugly to customize properly. You have to add a lot of packages which just appear to have their own syntax and incompatibilities.

Word can be quite good if you really know how to use it (styles, rules ...), of course, the equation editor is still ugly. I wish I could just type in a LaTeX formula and it would convert it into an OLE object (like equation magic lite but less buggy...).

I think LaTeX is good if you stay on the already well-defined paths but it's kinda ugly to customize without going through a lot of rules to learn.

I would like a nice language like tbook (xml dtd much simpler than docbook) and to be able to customize it with a simple css and export to pdf with apache's fop (and not going through LaTeX hacks like tbook is currently doing). Like it's done for the web but with printers in mind.

I've written a lot of documents using LaTeX and I'm afraid the author has overlooked one terrible problem; indeed he even acts like it doesn't exist! In particular, he writes: You add one sentence, which then pushes an image on to the next page, leaving a massive gap at the bottom of that page where your image once was. This then daisy-chains down, knocking other tables and images out of place all the way to the end of your document! It's a real laugh.

Fortunately, LaTeX is much more clever in this respect and positions your images and tables with a lot of common sense. So, if you want your image to appear at the bottom of a given page, it'll stay there!

Heh, heh. Not really.

LaTeX is infamous for doing exactly the same thing; in fact it's worse, because its algorithms insist on doing all the thinking for you: so, if you want an image (or a math formula) in a certain spot, good luck getting it there! The choice of vspace is often determined by what goes on two or three pages before (or after) the current page you're on, so changing two pages before can really get you confused. I struggled with this very problem in my dissertation: white space would automagically appear all over the place. It took a long time to get things acceptable (I'm still not happy).

For horizontal spacing issues, however, you have the reverse problem: LaTeX would rather overrun a right margin than leave too much space between words. (The infamous "overfull hbox", whose black slug indicating an error certain styles remove, incidentally, even in draft mode... grrrr) I'm not sure why Knuth thought an overrun was such a better idea than extra whitespace in an hbox, while extra white space was preferable to an overrun in a vbox, but the result in many published papers, and even some books, has been ugly. It certainly does not look professional, but the only way to fix it is to do some really obscure TeXing, or else completely rephrase your wording (the universal fallback I've seen in all LaTeX manuals). This, I think, is the worst problem: your choice of words is necessarily less important than LaTeX's obscure rules. It's the one frustration that unites both beginners and experts: when you're proofing a 260-page document, you have to remember that hyphenated words like "S-polynomial" (for example) won't break across lines, so they need manual hyphenation, and you often won't notice the overfull 2-pt hbox unless you're VERY CAREFUL (or you pay to separate the wheat from the chaff in TeX's output).

Those complaints aside -- for scientific publishing, LaTeX far outclasses anything else. I've never used Word to write a math document; given what I've heard from people (and what I read here) I never will. It was immensely classy of Knuth to make his program freeware.

RE: Has anyone mentionned Prosper yet?

Of course, one can not expect to find all the supported "objects" that PowerPoint supports (like embedded video).

Actually, although you can't do true embedded video, you can make a link that starts an external video app (or any other app for that matter). Not with prosper (which uses the old tex->dvi->ps->pdf route) but with any of the plethora of LaTeX presentation packages that are based on pdflatex (I would recommend beamer).

As to the downsides of LaTeX:

1. It is insanely difficult to get text paragraphs to flow around arbitrarily shaped inserts.

2. As some have already pointed out, LaTeX has its own ideas about what constitutes good typesetting, which may not be the same as your own. With enough experience, it is always possible to get it to do what you want, but many intermediate users get frustrated with this. An excellent resource for solving such problems is

http://www.tex.ac.uk/cgi-bin/texfaq2html?introduction=yes

3. Programming in LaTeX/TeX is almost completely unlike programming in other languages. Another source of frustration for some.

4. For ultra-high-end artistic typesetting (think of Bringhurst's 'Elements of Typographic Style'), LaTeX is probably not the best tool, although with some of the experimental micro-typography additions to pdftex it is 99% of the way there.

Apart from these, I'd say it was pretty perfect :)

Look at the difficulty that [hyphenating] "S-Polynomial" caused. How can anybody possibly claim that LaTeX allows one to concentrate on the meaning and not the layout? So LaTeX fails by its own criteria. Secondly: I agree with [an earlier] quibble over diagram placement. Again, one cannot just concentrate on the substance of one's writing. I used to get a lot of trouble with nested lists. Thirdly: graduates may love or hate LaTeX (you've guessed, I'm a LaTeX-hater and an Emacs-hater); but clerical and secretarial staff always prefer WYSIWYG. Fourthly: where I used to work, we used company macros to deliver a company look and feel for all our documents. These macros evolved in time, with the result that old documents could not be reprocessed. So an ASCII text format does not guarantee lifetime reproducibility.

How can anybody possibly claim that LaTeX allows one to concentrate on the meaning and not the layout?

You know, I kind of half agree with you there. With LaTeX the barriers between "content" and "layout" are a lot more porous than with, say, XML. Although this can sometimes be a source of problems, I think that LaTeX has the balance about right. After all, sometimes the layout is the content, or part of it.

you've guessed, I'm a LaTeX-hater and an Emacs-hater

Yeah, I guessed as much :) Seriously, I have come across a handful of colleagues with similar attitudes - despite being obviously highly intelligent, they refuse to "get" LaTeX and demand that all document preparation be happy-clappy, pointy-clicky. They do seem to be a very tiny minority, though, at least in my line of work. Is this due to some difference in brain structure, some traumatic childhood experience with backslashes, or what?

By the way, how do you feel about GUI front-ends like lyx? Personally, I find these inferior to emacs+AUCTeX+RefTeX as a LaTeX "IDE", but a good fraction of my colleagues and

students prefer them. Different strokes, I guess.

but clerical and secretarial staff always prefer WYSIWYG

I'd have to agree with [an earlier comment] that secretarial staff (like the rest of us) are creatures of habit. Feed them LaTeX from the cradle and they'll never look back!

THE PracTEX Journal

Travels in TeX Land: LaTeX for Productivity in Book Writing

David Walden

Abstract

In this column in each issue I muse on my wanderings around the TeX world. My column in this issue summarizes why I use LaTeX and gives examples of some productivity benefits of using LaTeX to write books.

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. You can contact him at
dave@walden-family.com

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

The PracTEX Journal, 2006, No. 2
Article revision 2006/05/16

Travels in TEX Land: LATEX for Productivity in Book Writing

David Walden

Abstract In this column in each issue I muse on my wanderings around the TEX world. My column in this issue summarizes why I use LATEX and gives examples of some productivity benefits of using LATEX to write books.

1 Why I use LATEX

Two reasons typically given for using LATEX are for its math support and for very nice looking typesetting. Neither of these is particularly important to me: I rarely have any math in my writing (but it is nice to be able to handle it easily in those rare cases where I do have it); I have a pretty undiscerning eye when it comes to typesetting, and what LATEX produces is more than good enough for me.

The things that matter most to me about LATEX are:[1]

1. its programmability and modularity
2. that I get to use a powerful editor with it
3. that the mark-up is clearly visible to me and can be changed directly with a text editor
4. its capabilities for explicitly noting cross-references, maintaining bibliographies, and automatically numbering chapters, sections, figures, tables, footnotes, etc., which permit easy reorganization of text within documents and reuse in other documents
5. its relatively slow pace of change and great concern among the developers for backwards compatibility

1. I'll discuss these more in a future paper; this paper will primarily concentrate on productivity methods I use related more or less specifically to book writing.

In other words, my use of LaTeX is primarily about productivity. (Of course, there are certain limitations on this productivity such as when I finish writing a book using LaTeX and the publisher tells me I must convert the text to Word and the figures to PowerPoint slides for input into the compositor's typesetting system.)

2 Writing books using LaTeX

Much of my work using LaTeX is on book length documents. For these I have compiled a more or less standard set of techniques that I feel help me be more efficient. I don't claim that the techniques I use are the techniques of a master; in fact, I view myself as an intermediate user of LaTeX — I know enough to make LaTeX jump through a few simple hoops, but not enough to know if my approaches are recommended or if they include some bad habits.

In my experience, publishers don't think much about the design of a book until they have the completed manuscript in hand. Since I use LaTeX to develop the original manuscript, I have to make lots of temporary design decisions, and I want to be able to change these decisions with a minimum of work when the publisher does begin to deal with the design. Also, I am currently working on a book that I will be self publishing, and settling on the design for this book is an iterative, experimental process where it is even more important to be able to make changes throughout the book (for instance, to the style of figure captions) with minimal work. My experience, however, should not prevent you from checking if the publisher of your document already has a standard style and perhaps even a LaTeX class file that you can use from the the outset of your writing. In any case, my emphasis here is *not* on the methods of representing preferences for appearance; my emphasis is on methods for easily and repeatedly *changing* the overall document appearance as well as on other methods for working efficiently on large documents.

Some of what I am about to describe for working efficiently on books or other long documents is probably already well known to many readers; perhaps you can make suggestions for how I might do things better.

(At several points in the following, I have included in parentheses discussions of basic TeX and LaTeX issues that reviewers and pre-publication readers have asked me about that are not actually on the subject of book-writing productivity.

Perhaps these parenthetical notes should have been footnotes, but I was too lazy to deal with the need for alternatives to `\verb` in footnotes.)

2.1 Include files

Suppose I am working on a book entitled *Breakthrough Management*, as I have been recently. I created a top level file named `bt.tex` with the following contents:

```
\documentstyle{btbook}
\begin{document}
\include{titlepages}
\include{preface}
\include{surviving}   % a chapter
\include{rapid}       % another chapter
. . .                 % more chapters
\include{acknowledgements}
\include{bibliography}
\include{bio}
\include{index}
\end{document}
```

The text from included files appears to LaTeX as if it was in the file `bt.tex` in place of the `\include` commands. In this way, I contain the text related to each chapter and other parts of the book in its own file. I let LaTeX take care of numbering the chapters and figures (or whatever) within chapters. If I later decided to change the order of chapters, I just change the order of the `\include` commands in the `bt.tex` files, and LaTeX automatically renumbers everything.

To work on one chapter at a time, my file `bt.tex` evolved to include many `\includeonly` commands, e.g.,

```
\documentstyle{btbook}
%\includeonly{preface}
%\includeonly{surviving}
\includeonly{rapid}
%includeonly{surviving,rapid}
. . .
```

```
\begin{document}
\include{titlepages}
\include{preface}
\include{surviving}   % a chapter
\include{rapid}       % another chapter
. . .                 % more chapters
\include{acknowledgements}
\include{bibliography}
\include{bio}
\include{index}
\end{document}
```

In the above example, only the file `rapid.tex` gets compiled when I run LaTeX on the file `bt.tex`. In this 10 chapter book I had a couple of dozen `\includeonly` commands in the bt.tex file that I could comment in and out to work on each chapter individually and with various combinations of related chapters.

(Because the `\include` commands result in text being typeset, they must go after the `\begin{document}` command. The `\includeonly` commands must go in the preamble or else LaTeX complains.)

2.2 Custom class file

I have created a file `btbook.cls` which is my own personal class file for this particular book. This file is processed when LaTeX sees the `\documentstyle{btbook}` command at the beginning of the file `bt.tex`. The first three lines of the file

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesClass{btbook}[2006/01/21 Breakthrough management book class]
\LoadClass{book}
```

define the class for this book to be named btbook and to be an augmentation of the LaTeX book class.

The rest of the lines of the file are read and executed when LaTeX is run as if they were lines of text immediately following the `\documentclass` command in the `bt.tex` file.

(If your publisher already provides a LaTeX class file, you can still collect all of the sorts of things I describe below in their own file and `\input` that file in

the preamble rather than just putting all these things directly in your preamble. I prefer not to have much in my preamble beyond the `\includeonly{...}}` commands that I am constantly commenting in and out.)

2.3 Packages

Next in the class file come the list of packages I use for writing this book.

```
\RequirePackage{mathpazo}  % Palatino is basic roman font
\RequirePackage[scaled=.95]{helvet}  % Helvetica is sans serif font
\RequirePackage{courier}   % Courier is typewriter font
\RequirePackage{graphicx}  % for including images
\RequirePackage{url}       % for formatting URLs
\RequirePackage[figuresright]{rotating} %to be able to rotate figures
\RequirePackage{lettrine}  % for dropped caps
\RequirePackage{paralist}  % for tighter list spacing
       \setlength{\pltopsep}{.05in}
\RequirePackage{comment}   % for comment environment
\RequirePackage{dw-endnotes} % for endnotes with reformatted numbers
\RequirePackage{setspace}  %\doublespacing
```

When I find I need to use another package, I add another `\RequirePackage` line to this list. (As I understand it, `RequirePackage` does the same job as `usepackage` except it doesn't allow the same package to be loaded twice which apparently might cause problems in some cases.)

Notice that the package name in one case includes the characters `dw-`. This is my convention for noting a package that I have modified. In such cases, the file of the modified package is in the same directory with the rest of the files for this book or in the local changes part of my `texmf` data structure. I seldom understand a package I am modifying; I typically use a hit and miss approach to chang stuff until I get the results I want.

Copy editors who edit on hard copy like double spacing, and I can provide that with a one character change — uncommenting the `\doublespacing` command on the last line above that loads the setspace package.[2]

2. The answer to the second question of the Ask Nelly column in this issue describes a different way to cause double spacing.

2.4 Miscellaneous useful macros

The following macros provide a few capabilities I use relatively frequently.

```
\newcommand{\Dash}{\thinspace---\thinspace} % space around em-dashes
\newcommand{\CK}[1]{\textbf{CK #1}}  % mark text that needs checking
\newcommand{\manote}[1]{% marginal note to myself
\marginpar{\scriptsize To do:\\
#1}}
\newcommand{\partitle}[1]{\medskip\noindent\textbf{#1}} % cut-in title

\let \Originalurl = \url  %change function of \url command
\renewcommand{\url}[1]{{\small\Originalurl{#1}}}
```

For some documents I have worked on, I have had many more such miscellaneous useful macros.

Anyone trying to improve productivity using LaTeX who doesn't already define his or her own macros should learn to do so. User-defined macros allow significant improvements in efficiency. For instance, the first macro above defines the command `\Dash{}` to be an abbreviation for the string of characters `\thinspace---\thinspace` which results in an em-dash being typeset with a *little* bit of space on each side of it, as in aaa — bbb. It is less characters and probably more reliable to type `\Dash{}` many times in a book than it is to type `\thinspace---\thinspace{}` many times. In my view, however, the greater benefit of defining the `\Dash` command comes when my publisher tells me that its style is closed-form em-dashes (no space on each side, i.e., aaa—bbb) or a more open form (aaa — bbb). To implement either of these changes *throughout* the book, I merely redefine `\Dash`, e.g.,

```
\newcommand{\Dash}{---} % no spaces around em-dashes
```

or

```
\newcommand{\Dash}{ --- } % full spaces around em-dashes
```

and recompile my document. Containing such style conventions within a few lines of a large document and being able to change the style throughout the document with only a few key strokes is an enormous advantage. (I'll give a more complex example of such containment when I discuss macros for figures and tables below.)

To redefine a command that already exists in LaTeX or has been defined by a package that has already been loaded, for instance to define a variation on `\url` as I do in the last two lines of my group of miscellaneous useful macros, I have to use the `\renewcommand` command. The `\renewcommand` works just like `\newcommand` except that LaTeX doesn't complain with `\renewcommand` if I try to give a definition to a command that already exists — a good thing to be warned about when one uses `\newcommand`. (In the next subsection I give another redefinition example — redefining `\footnote`).

2.5 Footnotes and endnotes

In the case of `\RequirePackage{dw-endnotes}`, I am using the `endnotes` package, modified slightly to change the format of the note numbers.

Typically, I put footnotes on the bottom of text pages where they are referenced, at least while I am drafting chapters and want to be able to see the notes without having to turn a bunch of pages. However, publishers tend not to like having footnotes — it makes a book look too academic to be popular, in their view. Thus, before actually publication, I often find myself converting all my footnotes to endnotes. The next commands in my class file do this.

```
\renewcommand{\footnote}{\endnote} %comment out to not have end notes

\newcommand{\dumpendnotes}
{\medskip
\begingroup
\setlength{\parindent}{0pt}\setlength{\parskip}{1ex}
\renewcommand{\enotesize}{\normalsize}
\theendnotes\endgroup \setcounter{endnote}{0}}
```

First, the `\footnote` command is redefined to be the `\endnote` command; this avoids me having to replace every instance of `\footnote` with `\endnote`. Then the class file defines a command (`\dumpendnotes`) that can go at the end of each chapter to dump the chapter's endnotes, formatted as I want them to me. If `\dumpendnotes` was already defined in LaTeX or some other package, LaTeX would warn me because I didn't do the definition with `\renewcommand`.

2.6 Formatting figures and tables

The next set of commands in the class file have to do with changing the format of figure and table captions without actually modifying a LaTeX or package file. The LaTeX default does not use bold face for captions and uses a period rather than a hyphen between the the chapter number and figure number within a chapter. The following changes patch LaTeX to follow my preference for bold face and hyphens.

```
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{\textbf{#1}. \textbf{#2}}%
  \ifdim \wd\@tempboxa >\hsize
    {\textbf{#1}. \textbf{#2}\par}
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
\renewcommand \thefigure
     {\ifnum \c@chapter>\z@ \mbox{\thechapter-\fi\@arabic\c@figure}}
\renewcommand \thetable
     {\ifnum \c@chapter>\z@ \mbox{\thechapter-\fi\@arabic\c@table}}
%end of commands to change style of figures and table captions
```

(I do not have to bracket these lines, top and bottom, with `\makeatletter` and `makeatother` commands as I would have to if this patch was in the preamble of my document; the at-sign is a letter by default in class files and packages. Some readers may be back a step, at the question of, "What is the about at-signs anyway?": The answer is that the files for basic LaTeX, for class files, and for other packages are full of macros names that include an at-sign (@), e.g., a macro named `\@makecaption` is defined at the beginning of the above example. My understanding is that an at-sign is used in low level programming of LaTeX, class files, and packages to create macro names that can't accidentally conflict with names that may be defined by users not doing such LaTeX "systems programming." An at-sign is normally not a letter and thus cannot be part of a macro name. However, in the above example I want to patch low level LaTeX code that includes at-signs in its macro names; if I was trying to make this patch in my

preamble (as I used to do before I learned to make some patches in a personal class file), I would have to tell LaTeX to temporarily turn at-signs into letters, make the patch, and then turn them back into non letters (other) so the rest of my program could use @ in the normal way where it is not a special character of any kind.)

Perhaps there is a caption package that would allow such changes without patching LaTeX, but I was shown how to make this patch a few years ago and it works, so why bother trying to find and learn a new package?

Next in my class file come a set of definitions for commands I use to include graphics. I seldom insert `\begin{figure}` and `\end{figure}` commands directly into my documents and insert `\begin{table}` and `\end{table}` commands only a little more often. It is inevitable that, before I am done with a big document, I will want to change the formatting relating all figure and tables — perhaps several times. Thus, I use macros for inserting almost all figures (or tables) such that I can make changes to formatting relating to the figures by making changes to only a few lines in the relevant macros.

```
\newcommand{\figfiletype}{pdf}
\graphicspath{{figures/}} %tell LaTeX a directory path for where to find figures

\newcommand{\snfig}[3]{   %scaled numbered figure
   \begin{figure}[htbp]  %drop htb for single page figures and uncomment following
%\vbox to \vsize{%
    \hfil\scalebox{#3}{\includegraphics{#2.\figfiletype}}\hfil
   \caption{\label{fig:#2}#1 \texttt{\small[#2 #3]}}
%\vfil
%}
   \end{figure}
   }

\newcommand{\sntab}[3]{   %scaled numbered tables
   \begin{table}[thbp]
%\vbox to \vsize{%
   \centering
   \caption{\label{tab:#2}#1 \texttt{\small[#2 #3]}} \smallskip
    \scalebox{#3}{\includegraphics{#2.\figfiletype}}
    \label{tab:#2}
%\vfil
```

```
%
   \end{table}
   }

\newcommand{\unfig}[2]{   %scaled unnumbered figure
   \begin{figure}[htbp]
    \hfil\scalebox{#2}{\includegraphics{#1.\figfiletype}}\hfil
   \label{fig:#1}\centerline{\texttt{\small[#1 #2]}}
   \end{figure}
   }

\newcommand{\swsnfig}[3]{   %sideways scaled numbered New figure
   \begin{sidewaysfigure}
    \centering
    \scalebox{#3}{\includegraphics{#2.\figfiletype}}
   \caption{\label{fig:#2}#1 \texttt{\small[#2 #3]}}
   \end{sidewaysfigure}
   }
```

For instance, the macro `\snfig` above takes three arguments. The text for a figure caption, the unique part of the file name for the graphic to be included, and a scale factor for the graphic, e.g.,

```
\snfig{This is the caption}{figure3-31}{.83}
```

The full name of the file to be included is the concatenation of the part of the file name that came from the second argument of the macro call, the directory that is specified by the `\graphicspath` command (an option of the `\graphicx` package) as the place LaTeX searches for figures, and the `\figfiletype` definition as the file name extension. The latter is useful because sometimes all of my figures are `.eps` files and sometimes they are `.pdf` files, and sometimes I switch between these two formats at different times in the production of the book. (When using `.eps` format, I compile using LaTeX and a dvi-to-pdf conversion; when using `.pdf` format, I use pdfTeX to compile. If the graphic format was changing from file to file within the document, I would instead specify the format as another argument to the `\snfig` command.)

While I am drafting and revising a book manuscript, I want to be able to look at a figure in the printed output and know what file I need to modify to

change the figure. Thus, my macros for including figures and tables causes the file name to be included in the printed output in small letters enclosed in small square brackets. When it comes time to create the final manuscript, I delete the instances of `\texttt{\small[#2 #3]}` from the macros and recompile the book's LaTeX files.

The definitions of `\snfig` and `\sntab` also include several lines that are commented out. Professional editors often like to see the manuscript with figures or tables each on its own page rather than in-line with the text. Commenting in these few lines puts the figures and tables of the whole book on their own pages.[3]

The `\snfig`, `\sntab`, and `\swsnfig` macros also define labels for cross referencing the figures with `\ref` or `\pageref` commands. A slight limitation of my implementation is that I cannot reuse the same figure or table file without confusing the labeling. However, it is easy enough to create a duplicate figure or table with a different file name.

I typically create all figures and most tables outside of TeX itself and include them from separate files. If I found myself inserting very many tables directly into my `.tex` files rather than including them from graphics files, I would define a `mytable` environment so that I could still contain and simply change the sort of formatting I have discussed.

2.7 Thought breaks

The next group of commands (mostly commented out) are various options for indicating what I call "thought breaks" — places where formatting indicates a change of topic big enough to highlight but not big enough to have its own section or subsection title. (These commands are defined with `\def` because I know they will pick up the correct arguments this way, and I am not sure enough of the details of how `\newcommand` works. (I understand the details of how TeX defines a macro and then collects its arguments when the macros are called because Knuth explains it pretty completely in *The TeXbook*. In particular, TeX allows macro calls where the arguments of the macro are not all embedded in pairs of braces. However, I have never stumbled across a rigorous explanation of how a macro defined with `\newcommand` collects its arguments and thus in what

3. The answer to the second question of the Ask Nelly column in this issue describes a different way to put each figure on its own page.

situations arguments not in braces will be recognized or to what extent LaTeX defined macros can have both of what Knuth calls delimited and undelimited arguments — and I have not bothered to study the LaTeX code to figure it out. Consequently, out of ignorance, I use \def to define macros which don't have their arguments delimited by braces.)

```
%\def\newthoughtgroup#1{\bgroup \afterassignment\BigFirstLetter \let\next=}
%\def\BigFirstLetter#1{\bigskip\noindex{\Large#1}}

%adapted slightly from Victor Eijkhout on c.t.t
%\def\newthoughtgroup#1{\BigFirstLetter#1$}
%\def\BigFirstLetter#1#2${\bigskip\noindent{\Large #1}#2}

\def\newthoughtgroup#1{%
   \bigskip\noindent {\large #1}}

%\def\newthoughtgroup{%
%  \bigskip\noindent }

%big bold dropped cap letter with rest of word small caps
%\def\newthoughtgroup#1#2 {\bigskip\noindent\lettrine{#1}{#2}\ }

%\def\thoughtbreak{\vskip2pt
%    \centerline{$^{\vrule width2cm height 1pt}$}\vskip2pt\noindent}
```

The version of \mythoughtgroup currently not commented out indicates the new thought by a vertical space and a slightly bigger capital letter at the beginning of a non-indented paragraph.

2.8 Chapter formatting

The final set of commands in my class file has to do with with the beginning and ends of chapters. At the beginning and ending of each chapter I insert some commands that I can change either by changing the commands themselves or changing macros in the class file.[4]

4. When I first started customizing my page headings a few years ago, I used the fancyheadings package; recently I have learned that the package fancyhdr has replaced fancyheadings,

```
\RequirePackage{fancyhdr}
\pagestyle{fancyplain}
\newcommand{\mypartname}{}
\newcommand{\mychaptername}{}
\lhead[\fancyplain{}{\thepage}]{\fancyplain{}{}}
\chead[\fancyplain{}{\mypartname}]{\fancyplain{}{\mychaptername}}
\cfoot[\fancyplain{}{}]{\fancyplain{\thepage}{}}
\rhead[\fancyplain{}{}]{\fancyplain{}{\thepage}}
\newcommand{\EMPTYPAGE}{\clearpage\thispagestyle{empty}\cleardoublepage}

\newcommand{\ENDCHAPTER}{\dumpendnotes}
\newcommand{\fENDCHAPTER}{\vfil\dumpendnotes}
```

In the class file for the book from which I drew these illustrations, there are a couple of alternative macros that I can include at the end of each chapter to dump the endnotes, but the end-of-chapter macros could be defined to cause other actions and outputs. In this book (which has only 10 chapters) I do not combine everything in a single beginning-of-chapter macro (e.g., `\BEGINCHAPTER`), but I have done this with some books (e.g., the 20 chapter book I am also currently working on). The typical beginning-of-chapter commands for the chapter with the file name `rapid.tex` (mentioned earlier) are

```
\EMPTYPAGE
\chapter{Rapid Change in a Global World}
\label{ch:rapid}
\renewcommand{\mychaptername}{Chapter \thechapter: Rapid Change in a Global World}
```

2.9 Using a fully developed class

For some books I have included different or additional capabilities in my custom class file.

Obviously, I could also use a fully developed class such as memoir rather than making lots of modifications of my own to the LaTeX's standard book class. However, I suspect I would still use some of the ideas I have described above.

---

but I have not yet bothered to rewrite all the heading commands to use the new forms that come with the fancyhdr package and don't use the fancyplain device.

It is clear that TEX and its derivatives with their explicit, visible markup provide a strong base for incrementally building a personal library of techniques that are easy to apply from one project to the next.

Acknowledgements

I owe thanks to many sources for what I have learned about using LATEX — books, the `comp.text.tex` list, the `texhax` list, and many individuals. Of course, none of them are responsible for lessons I have mislearned.

I can't remember and acknowledge everyone who directed me to techniques illustrated in this column; however I can remember some of them. Karl Berry reviewed an early version of this column and earlier told me about some of the methods I have described here. I also remember Peter Flynn, Steve Peter, and Steve Schwartz telling me about particular techniques. Peter Flom, Will Robertson, and the anonymous reviewers provided many helpful suggestions about the manuscript.

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. More history is at www.walden-family.com/dave.

THE PracTEX Journal

Ask Nelly:
What is different when I click on the pdfLaTeX rather than the LaTeX icon in WinEdt?
How do I convert my document to the publisher's requirements for double-spacing, line numbers, and figures on their own pages?
How do I interrupt an enumerate environment and then continue it later in the document?

The Editors

Abstract

Ask Nelly is a question and answer column. Nelly is the quiet person who sits at the back corner desk, who knows a lot, and when asked any question is always ready with a patient answer. If Nelly doesn't know the answer, Nelly will know an expert who has the answer. Feel free to Ask Nelly about any aspect of LaTeX, TeX, Context, etc.

- Comment on this paper
- Send submission idea to editor

Q: I have heard that pdfTeX has capabilities beyond those available with TeX alone. What is different when I click on the pdfLaTeX rather than the LaTeX icon in WinEdt?

A: First, it is somewhat misleading to speak of "pdflatex" in this context. Here is a recap of the basic TeX "engines" (actual binaries):

- Knuth's original "tex" (still used in the free distributions for plain TeX)
- "etex" done by Philip Taylor et al. (not currently used in the distributions)
- "pdftex" done by Han The Thanh et al. (not currently used in the distributions)
- "pdfetex" which combines etex and pdftex (used for everything but "tex"), presently maintained by Thanh, Martin Schroder, and others.

So, when you invoke PDFLaTeX, what you are really invoking is the pdfetex binary, with the LaTeX macros loaded as a "format" (.fmt file).

Second, and to get to your real question, you heard right that pdfTeX has capabilities beyond the original TeX. Two main new features in pdftex that come immediately to mind are:

- it can read most image formats (jpg, png, pbm, pdf, metapost output) directly -- with general .eps figures being the primary exception.
- there are "microtypography" capabilities for even higher output quality than the original TeX had; these additional capabilities are made available in LaTeX with \usepackage{microtype} -- for more information, see `http://tug.org/TUGboat/Articles/tb25-1/thanh.pdf`

Also, etex has additional debugging information available from \tracingall. This has proven helpful for tracing obscure problems. It has a number of other additional programming features, but no significant output-related features.

Most other features often associated with PDFs (hyperlinks, colors, etc.) can be achieved with the original TeX. There are related advanced features that are part of the PDF document format, which of course pdftex gives access to, but this is beyond the scope of your question and this answer. See the pdftex manual.

Q (follow-up): Follow-up question: If the pdfLaTeX icon in WinEdt and the LaTeX icon in WinEdt both result in calls to pdfeTeX where does the distinction get implemented such that the former can use all the different image formats and the later only can use .eps. Is this because the former calls pdfeTeX in pdf output mode and the later calls pdfeTeX in dvi mode?

A (follow-up):

That's exactly right.

The distinction is created when the .fmt files mentioned above are made. Invoking a binary named "pdflatex" reads the so-called "format file" pdflatex.fmt, which was created (in teTeX and TeX Live) by reading the file pdflatex.ini. Similarly, invoking a binary named "latex"

reads the format file latex.fmt, which was created by reading the file latex.ini. In both cases, the exact same LaTeX macro files are read.

The difference is that latex.ini sets \pdfoutput=0 (\pdfoutput is a new primitive parameter in pdfTeX-based engines). This, finally, is what makes pdf(e)tex's output be DVI instead of PDF. (pdflatex.ini leaves \pdfoutput=1, meaning to output PDF.) The image-reading code and such are all implemented internally to depend on \pdfoutput.

I should say that the files might not be named "whatever.ini" in other distributions, and the exact names of the .fmt files aren't necessarily identical, but that is all an implementation-dependent issue. The underlying setting of \pdfoutput must be the same; that is the primitive parameter which controls the behavior. (This is true in the freely available TeX systems that I'm aware of. Of course I do not know how the proprietary implementations are implemented!)

Final dangerous bend: there are actually even more steps done behind the scenes, so that the .fmt files can get created on the fly instead of users having to create them explicitly, but that's not germane to your pdf question.

The above question and follow-up question were answered by **Karl Berry**. Karl has been a long-time board member of TUG, became TUG president in 2003, and was elected for another term in 2005. Among other projects, he is co-administrator of the tug.org server, co-editor of TeX Live, and a member of the TUGboat production team. For more about Karl, see his TUG interview.

TPJ

Q: I have written my paper in LaTeX and want to submit it to a peer-reviewed scientific journal. The journal, however, is a tad old-fashioned, and wants me to submit my paper with double linespacing, linenumbers and with each figure and table on a separate sheet rather than in the text. In the text it should only say: "[Figure 1 about here]". How do I do that?

A:

This is, unfortunately, a common problem in several fields of science. On the bright side, this means that several other people came across these problems and solved them. We can just use their expertise. Let's take it in three steps.

1. The linespacing.

   Linespacing in LaTeX can easily be adjusted, using the \baselinestretch command. This is defined by the \begin{document} command, so it should be set in the document itself, rather than in the preamble. To typeset your paper with double linespacing, just give the commands

   ```
   \renewcommand{\baselinestretch}{2}
   ```

```
\normalsize
```

immediately after the \begin{document}. The \normalsize command sets the text to normalsize, which is what it starts at anyway, but when it is left out, your text will not be typeset with double linespacing.

To set linespacing at, for example, one and a half, just replace the "`2`" in the above command by "`1.5`"

Note: Double linespacing usually looks exceedingly ugly, and should, therefore, only be used when absolutely necessary.

2. Linenumbers

Stephan I. Böttcher has developed the lineno.sty just for the purpose of listing linenumbers in your document. If your TeX-distribution does not install it, it can be found on CTAN in the directory

```
/tex-archive/macros/latex/contrib/lineno/
```

The package is now maintained by Uwe Lück, and is currently at version 4.41. It has many options (such as referring to linenumbers, just as you would to pages), for which I refer you to the manual that comes with it. For our purpose only a few of those options are needed. In your document preamble include the package:

```
\usepackage{lineno}
```

And in the beginning of your document, turn on linenumbering with the command:

```
\linenumbers
```

If necessary, you can turn it off again later in your document by using

```
\nolinenumbers
```

3. Figures and tables at the end of the document

In order to get the floats (the technical term for LaTeX material that can be moved about in yoru document; figures and tables are the most common examples of floats) to be placed at the end of your document, just use the endfloats package by James Darrell McCauley and Jeff Goldberg. If you do not have it yet, download it from CTAN in the directory

```
/tex-archive/macros/latex/contrib/endfloat/
```

and simply include it in the preamble:

```
\usepackage{endfloat}
```

This will take care of everything. All floats will be moved to the end of the document, each on a page of their own. The package also automatically generates a list of tables and a list of figures preceding the tables and figures respectively. It also puts the text "[Figure 1 about here.]" on a line of its own in your text at the point where you had your original `\begin{figure} ... \end{figure}` declaration.

Using these three tricks (or only one or two of them, depending on the journal you're writing for) you can butcher your pretty LaTeX layout in no time, just the way the editors want you to!

This question was answered by **Yuri Robbers**. He holds a degree in Animal Behaviour and is a teacher, a researcher and a published author. He's always had a keen interest in typography, possibly because his father is a professional typographer. Ever since he discovered LaTeX in 1995, he's always done his best to avoid using word processors, and he has embarked on a quest to learn as much as possible about TeX and its derivatives, and apply this knowledge whenever possible. Contact him at `yuri.robbers@gmail.com`

TPJ

Q: I have an enumerate environment in my document that I want to interrupt, and continue later on in the document.

A:

This is a common problem. Not only does it occur whenever one wants to explain something in more detail in between "enumerate" entries, but also when creating a presentation using, for example, the Beamer package, and one enumerated list spans more than one slide.

The way to do this is to creat an auxiliary counter in the preamble of your document like this:

```
\newcounter{saveenumi}
```

Then, whenever you want to end a numbered list, save the current value of its counter in your newly defined counter like this:

```
\begin{enumerate}
  \item ...
  \item ...
  \item ...
  \setcounter{saveenumi}{\theenumi}
\end{enumerate}
```

Now you can type additional text, or end your slide and begin a new one, and then, n your

next enumerate environment, just restore the saved counter, and continue where you left off:

```
\begin{enumerate}
  \setcounter{enumi}{\thesaveenumi}
  \item ...
  \item ...
  \item ...
\end{enumerate}
```

Please note the difference between `enumi` and `theenumi`, as well as between `saveenumi` and `thesaveenumi`. In order to read the value of a counter, use its name prefixed with "`the`", and in order to store something in a counter, use its name without any prefix. There is no need to define the version with "`the`": LaTeX takes care of that automatically.

In case you have an enumerate environment within an enumerate environment (within an enumerate environment, etc.) then just create an additional new counters (or two or three...), like this:

```
\newcounter{saveenumi}
\newcounter{saveenumii}
\newcounter{saveenumiii}
```

And at the end of your list, store all relevant counters:

```
\setcounter{saveenumi}{\theenumi}
\setcounter{saveenumii}{\theenumii}
\setcounter{saveenumiii}{\theenumiii}
```

and restore them again when you begin the new list:

```
\setcounter{enumi}{\thesaveenumi}
\setcounter{enumii}{\thesaveenumii}
\setcounter{enumiii}{\thesaveenumiii}
```

That should do the trick.

This question was answered by **Yuri Robbers**. He holds a degree in Animal Behaviour and is a teacher, a researcher and a published author. He's always had a keen interest in typography, possibly because his father is a professional typographer. Ever since he discovered LaTeX in 1995, he's always done his best to avoid using word processors, and he has embarked on a quest to learn as much as possible about TeX and its derivatives, and apply this knowledge whenever possible. Contact him at `yuri.robbers@gmail.com`

THE PracTEX Journal

Distractions:
Sudoku ABC
Winners of the type quizzes

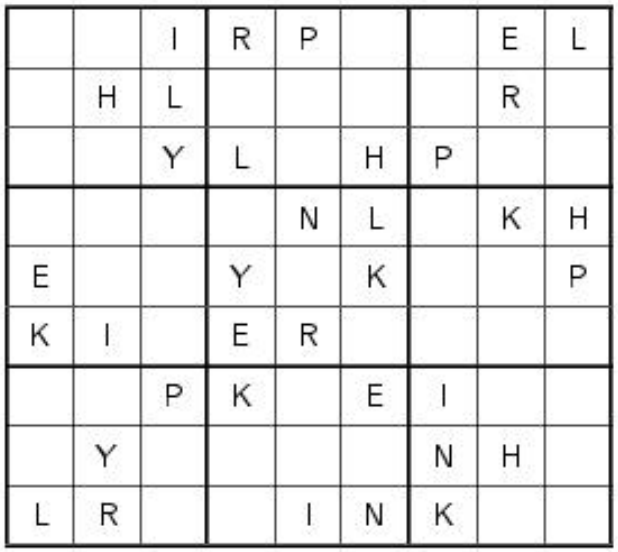
The Editors

- Comment on this paper
- Send submission idea to editor

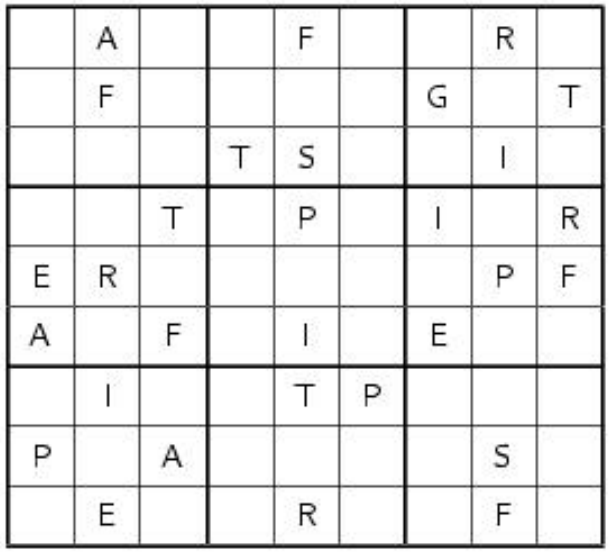
Sudoku ABC

These puzzles are similar to regular sudokus except they use letters instead of numbers. Using the nine letters in the box beneath each puzzle, enter letters so that each appears only once in every 3x3 box, row, and column. When you have completed the puzzle a surprise word will appear in a row or column. (One of the surprise words is not a real word but will be familiar to LaTeX and ConTeXt users.)

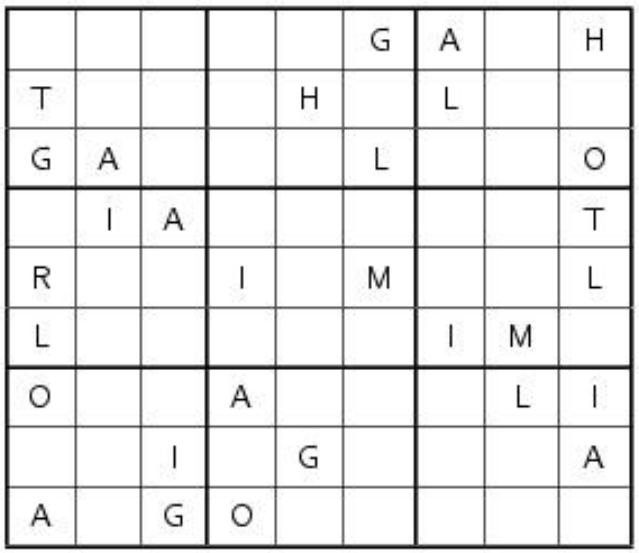
1. 2. 3.



		I	R	P			E	L
	H	L					R	
		Y	L		H	P		
				N	L		K	H
E			Y		K			P
K	I		E	R				
		P	K		E	I		
	Y					N	H	
L	R			I	N	K		



	A			F			R	
	F					G		T
			T	S			I	
		T		P		I		R
E	R						P	F
A		F		I		E		
	I			T	P			
P		A					S	
	E			R			F	



					G	A		H
T				H		L		
G	A				L			O
	I	A						T
R			I		M			L
L						I	M	
O			A				L	I
		I		G				A
A		G	O					

P	H	I	L	Y	K	E	R	N

G	R	A	P	E	F	I	S	T

L	I	G	H	T	R	O	A	M

Answers

Sudoku fans may want to try Peter Wilson's sudokubundle. This LaTeX package provides a coordinated set of packages for typesetting, solving, and creating Sudoku puzzles.

TPJ

Answers to the Typeface Quiz:

1) What famous novel (and the movie or movies made from it) has at least three characters having the last names of two well known type designers?

The Hound of the Baskervilles

character Henry BASKERVILLE — designer John Baskerville
also
character Charles BASKERVILLE
character Hugo BASKERVILLE

character Sherlock HOLMES — designer Kris HOLMES

2) What typeface has had two different authorized names, one meaning bizarre or monstrous and the other signifying a country?

Helvetica. Originally Named Haas Neue Grotesk ("Haas New Grotesque" in English) later changed to Helvetica, for the country of its origin - Switzerland, which was called Helvetia in Latin.

3) What typeface is named after the author of the first book printed in that type, a philosophical dialogue between two men hiking up a mountain?

Bembo, a 1929 revival of a typeface first cut by Francesco Griffo and first used in *De Aetna*, by Pietro Bembo, a dialogue in the Platonic style about Bembo walking up Mt. Aetna with his father, published by Aldus Manutius in Venice in 1495.

The first prize winner is **Jan Hlavacek**, who will receive a Lucida fonts license, donated by Bigelow & Holmes.

Answers to the TypoNovice Contest:

1) Palatino (Book Antiqua, a current clone, is also an acceptable answer)
Designed by Hermann Zapf
Published by Linotype
http://www.linotype.com/1317/palatino-family.html

2) Georgia
Designed by Matthew Carter
Published by Microsoft
http://www.ascendercorp.com/msfonts/georgia_family.html

3) Bookman
Designed by Chauncey H. Griffith, based on a design by Alexander Phemister
Published by Bitstream
http://www.myfonts.com/fonts/bitstream/bookman/

4) FF Dax
Designed by Hans Reichel
Published by FSI FontShop International
http://www.fontshop.com/?fuseaction=catalog.fontpackage&searchby=manufacturer&displayfontid=FF.10269.0.0

5) FF Meta
Designed by Erik Spiekermann
Published by FSI FontShop International

http://www.fontshop.com/?fuseaction=catalog.fontpackage&searchby=manufacturer&displayfontid=FF.11025.0.0

6) Bank Gothic
Designed by Morris Fuller Benton
Published by Bitstream
http://www.myfonts.com/fonts/bitstream/bank-gothic/

The first prize winner is **Jan Steffan**, who will receive the book *Stop Stealing Sheep & Find Out How Type Works*, donated by FontShop.

Answers to the TypoGuru Contest:

1) Tarzana
Designed by Zuzana Licko
Published by Emigre
http://www.emigre.com/EF.php?fid=124

2) Warnock Pro
Designed by Robert Slimbach
Published by Adobe
http://store.adobe.com/type/browser/P/P_1710.html

3) Hydrous
Designed by Anuthin Wongsunkakon
Published by Psy/Ops
http://www.psyops.com/html/spec_hydrous.html

4) Incognito
Designed by Gábor Kóthay
Published by Fountain
http://www.fountain.nu/catalogue/incognito.asp

5) Filosofia
Designed by Zuzana Licko
Published by Emigre
http://www.emigre.com/EF.php?fid=97

6) Relato
Designed by Eduardo Manso

Published by Emtype
http://www.emtype.net/relato_1.html

The first prize winner is **Isaac Sijaranamual**, who will receive the *Indie fonts* books and CDs, donated by SOTA (Society of Typographic Aficionados).

There were no entries for the Math font quiz. But there is still a chance to test your math font knowledge and win a prize: Three entrants who answer three or more questions correctly by June 30, 2006 will receive a free copy of Michael Spivak's new *MathTime Professional 2 fonts* "Lite" version ($49 value).
