Advanced LaTeX

Joel Berger

November 12, 2009

Contents

1 Introduction

In this talk I intend to give a brief overview of the tool, techniques and packages that I use to make the most LaTeX. Some in the audience may know some or all of these things or may have other methods to accomplish these tasks. I hope you will share those in our discussion!

Further this outline is made for both of a series of talks I am giving on this topic, on in the CS department and one in the Physics department. In these talks I may focus on slightly different material; in the CS talk I may focus more on the Beamer class than the toolchain as this audience may have its toolchain decided upon; in Physics, as I know some of you are new to LaTeX as well as programming I may spend more time on the toolchain and its relationship to using BibTeX. Both groups should familiarize themselves, at some point, with UICThesi.

2 Tool Chain

Your toolchain includes your LaTeX distribution (i.e. MikTeX), your editor and any other software that you use to produce your documents. You may choose the software in your toolchain to suit any purpose that you have. There are some LaTeX programs that I use in order to produce my documents quickly and with lots of simplifications built in to ease the process.

2.1 Eclipse / TEXlipse

Eclipse is a free software development program. Remember that your LATEX source is actually a software program defining the document you want to produce. TEXlipse is a free plugin for Eclipse that provdes LATEX integration.

TEXworks is a simplistic, stipped-down LATEX editor that has two advantages: A clean interface that isn't scary to newcomers and inverse search in PDF documents. It is just fine to stay with the clean interface if you like that, but other editors have added features that TEXworks gives away in the name of simplicity. The one exception is the inverse search. Almost all editors can inverse search in DVI mode, but that document format has been supplanted by PDF. TEXworks is the first (and almost only) to have inverse search in PDF mode; if you love this feature you may want to stick with this as your editor. Otherwise trying out other editors may help you streamline your document production.

2.1.1 Installation

Get Eclipse (Classic version is most suitable) from `http://www.eclipse.org/downloads/`. Be sure to get the build for your platform (Windows/Mac/Linux). You can get the bleeding edge from `http://download.eclipse.org/eclipse/downloads/`; I recommend a stable build of an upcoming version to get new features with the stability that you need.

Unpack anywhere and run "eclipse." Click "Help"→"Install New Software." In the box "Work with:" enter `http://texlipse.sourceforge.net` for the release build or `http://texlipse.sf.net/daily` for the bleeding edge (I know it has some good features). You will have to give it some name; anything will do. Then click the check-box for Texlipse and follow the install prompts. After restarting you will have TeXlipse fully installed.

2.1.2 Setup

In "Windows"→"Preferences" under "TeXlipse"→"Builder settings" set the bin directory to your LaTeX folder. Similarly find your viewer, this might be acrobat reader, though I like using a program called Xpdf because it can refresh the viewed document for when you recompile.

2.1.3 Benefits

Some of the reasons I like this editor are

- Code folding
- Colorization
- Error/Warning sorting
- Tasks (ToDo menu)
- Table entry
- Outline view (even with included documents)
- Autocompletion
  - Environment auto-matched insertion (enter begin, also get end)
  - Delimiter auto-matched insertion
  - Command completion
  - Converts `"` to ‘‘ or ’’ automatically
  - Converts `...` to `\ldots` automatically
- Similar to delimiter insertion is delimiter pairing. A cursor after a delimiter highlights its pair.
- Cite and ref lists (remembers your labels so you don't have to)
- Cite and ref pairing (cursor in ref highlights the label, etc.)
- Tab exiting
- Multiline commenting
- Build management (Helps run builders to cross-reference)

2.1.4 Use

To use TEXlipse you first create a project. Once this is going all you need to do to compile is save the source document. Thats right, it automatically compiles after every save. Sometimes if you change section numbers etc. you might want to save and then do "Project"→"Clean."

For further functionality:

- Many of the features are configured in the preferences menu seen above.
- To view the complied file hit Ctrl+4 to view with the viewer set in preferences or double click on the file in the "Project Explorer" to use the system default.
- Be sure to initially add extra views "Window"→"Show view"→"Other and under Texlipse you can get the Table and Full Outline.
- Autocompletion and lists are triggered by pressing Ctrl+Space.
- Tab exiting is indicated by a green cursor after the delimited block you are in. Hitting tab will bring your cursor to that point (i.e., outside parenthesis or braces).
- Delimiter matching should be automatic, but be careful when doing complicated nesting as sometimes this gets it confused.
- Tasks are triggered by adding the comment `%TODO message` in your source and your message will appear in the Tasks tab.
- Multiline comment by selecting what you want to (un)comment and hit Ctrl+5 (think Ctrl+%) and it will put(remove) a comment symbol (%) in front of each highlighted line.

2.2 Zotero

Zotero is a free firefox extension installed (inside firefox!) through "Tools"→"Add-ons." It saves, formats and exports citations from inside firefox!

To use, find some article's page online, say `http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=5888920` then in the url bar click on the new "Save to Zotero" button (looks like a page) next to the usual "bookmark" star. This saves the information of the article to Zotero.

Open Zotero by clicking the icon in the lower right corner in firefox. You can export saved entries by right-clicking "My Library"→"Export Library." Pick "Bibtex" and the save in the same directory as your .tex files; it should have a .bib extension.

2.3 BIBTEX

BIBTEX is an extension to LATEX that aids typesetting your bibliography.

2.3.1 General Use

To use BIBTEX you create a .bib file with bibliographic information in a specific format. This format is complicated and specific; since there are so many reference management programs that can format it for you, I recommend these. The documentation can be found at `http://www.bibtex.org/`.

The most important piece of information that must be in the entries is the name key. This is your method of referencing that particular reference item. While you are typing your document now all you need to do to create a citation is enter `\cite{key}` and BIBTEX automatically adds a number and will add this reference to the bibliography section.

Finally we need to include the bibliography at the end. Just before the `\end{document}` command we need to do two things we need to give a bibliography style which acts — much like a document class acts for the paper — in defining how the bibliography should look. For a basic bibliography we can do `\bibliographystyle{plain}`. Then we just need to tell it what our .bib file is named with `\bibliography{mybiblio}` if your file is called mybiblio.bib.

2.3.2 As Part of Toolchain

My biggest complaint was that since Zotero is so "hands-off" that you don't know what the key for each item is! It is (usually!): <Author's Last Name>_<Title's First Word>_<Publication Year>. This is why the toolchain concept is important; since TEXlipse has cite listing built in, the problem of knowing the cite key doesn't really matter anymore, simply find it in the drop-down menu with "Crtl+Space."

2.4 Creating Graphics

I use several programs for creating figures, some of them are included here.

2.4.1 SVG Images / Inkscape

SVG is a type of vector image format. This means that rather than storing pixel information the image is comprised of vector that form the image. Vector images are preferred when making a publication or presentation for several reasons.

- The images scale without loss of resolution
- They can be easily modified to make corrections or to reuse for future projects
- With a little work one can make a series of images used for an incremental presentation

However, their biggest problem is that they cannot be used in LaTeX directly. They must eventually be converted to some other format for inclusion.

Inkscape (`http://www.inkscape.org/`) is a nice program for creating and editing SVG images. It is available for all platforms for free and can export to .png and .pdf (which is also a vector format) for use in your documents. It does have some rendering capabilities (simple functions etc.) but for making plots of data I often generate an initial SVG from another program and then edit it in Inksape.

2.4.2 QtiPlot

QtiPlot (`http://soft.proindependent.com/qtiplot.html`) is a semi-free scientific plotting program similar to Origin (free for Linux users and free source code for other platforms but you have to compile yourself). There are several other programs like this one, however this seems to be the most advanced.

Once the plot is created it can be exported as an SVG (or other formats). It is not foolproof, sometimes I have to clean it up in Inkscape after the export.

It also allows export to TikZ (see section 2.4.5) by exporting to .tex. However I have found this to be even less relible than SVG export, though it would give a starting point for using TikZ.

2.4.3 Mathematica Export

You can also export as SVG or other image formats from Mathematica. For example, the following will export a 3^{rd} order Super-Gaussian example figure as both an svg and png in the same directory as the notebook.

```
F[n_, x_]:=Exp[-x^(2 n)];

(*For exporting as svg*)

Export[
  StringJoin[
    NotebookDirectory[],
    "SG3.svg"
  ],
  Plot[
    F[3, x], {x, -3, 3},
    PlotStyle -> {Black, Thick},
    Axes -> False,
    Filling -> Bottom,
    FillingStyle -> Cyan
  ]
]

(*Or for png*)

Export[
  StringJoin[
    NotebookDirectory[],
    "SG3.png"
```

```
  ],
  Plot[
    F[3, x], {x, -3, 3},
    PlotStyle -> {Black, Thick},
    Axes -> False,
    Filling -> Bottom,
    FillingStyle -> Cyan
  ],
  ImageSize -> 700,
  ImageResolution -> 350
]
```

Again, I then go back to Inkscape to do the finer points.

2.4.4 GIMP

GIMP (`http://www.gimp.org/`) is a free image manipulation program comparable to Photoshop. I use it sparingly as it is very complicated. I do find it very useful for the final image sizing and resizing as needed, as well as cropping etc.

2.4.5 TikZ Package

I haven't tried using TikZ directly yet, however it is this package that gives Beamer its good looks. Basically it is a method of drawing using LaTeX commands directly inside your source. This native method saves the hassle of having to get your figure to meld correctly with your paper, however it is not a graphical drawing program so you will have to work on your figure's source as you do for your text.

Find out more at `http://sourceforge.net/projects/pgf/`. A good tutorial is available from the PracTeX journal at `http://www.tug.org/pracjourn/2007-1/mertz/` and examples at `http://www.texample.net/tikz/`.

3 Beamer

3.1 Basics

Beamer is highly documented, however the manual is very long, and it can be hard to find what you want. Therefore I will walk you through a simple presentation.

3.1.1 Themes

First you pick your theme. I like the Madrid theme with with `secheader` option which shows the current section in the header. I also like the color theme `ostrich` but it is not included with the typical LaTeX-Beamer distribution so I will leave it commented. It was actually developed by some folks at UIC, so it is easy for us to use. Get it at `http://alcazar.sisl.rites.uic.edu/wiki/view/Main/SISLBeamerColorThemes`. It was a coincidence, I really like

```
\documentclass{beamer}
\usetheme[secheader]{Madrid}
%\usecolortheme{ostrich}
\setbeamercovered{transparent=50}
```

Themes are a nice feature as you can change it quickly and easily if you change your mind. I threw in an additional command, we will see it's use later.

3.1.2 Title Material

Similar to the title material in an article Beamer has macros to make a title slide.

```
\title[Short Title]
  {Much Longer Title for Front Page}
\author{Joel Berger}
\institute{Venue or Employer}
\date{\today}

\begin{document}

\begin{frame}
  \titlepage
\end{frame}
```

3.1.3 Frames

Beamer makes slides called frames.

```
\begin{frame}{Title of Frame}
  Content of frame
\end{frame}
```

Since it creates a PDF as its output it cannot have incremental slides as Powerpoint does. What it does instead is create a series of PDF pages that when viewed sequentially accomplishes the same task. We do the incrementing by adding markers such as `<2->` meaning show from increment 2 and on.

```
\begin{frame}{I like lists}
  I like lists because
  \begin{itemize}
    \item They keep me organized
    \item<2-> It's fun to check
      off completed items
    \item<3-> I'm a geek
  \end{itemize}
\end{frame}
```

Since we set the transparency above, items that are not yet uncovered are present but 50% transparent. There are other methods to accomplish incrementing this but that is the robust one.

For items that cannot take a simple marker or to do specific things, there are commands like: `\uncover<2->{}` (much like the markers above), `\visible<2->{}` (hidden even if transparency is set) and `\only<2->{}` which is not typeset at all until increment 2. Give your text or commands as arguements to these commands and see what happens.

3.1.4 Blocks

Beamer contains a special mechanism to create boxes (for emphasis or organization, etc.) called blocks. These blocks may have a title and are automatically sized to fit their contents. In fact the whole thing can be brought in later for even more emphasis. For example

```
\begin{frame}
  {My Experiment and Results}
  My experiment and how it works.
  More info, blah blah blah.
  \only<2->{
    \begin{block}{My Results}
      This is the stuff that
      I want you to see!
    \end{block}
  }
\end{frame}
```

3.1.5 Columns and Graphics

Maybe you want to have a picture on one side and a list on the other. Beamer also has columning mechanisms that are more powerful than the simple twocolumn article. First you need a `columns` environment and then you need `column` environments that have their width as options. I recommend using relative sizing and not having the columns overlap. For graphics use the same commands you are used to, you dont need a `figure` environment but sometimes it helps. Also the linewidth is redefined inside the column. In this example look at the width I define for the graphic.

```
\begin{frame}
  \begin{columns}
    \begin{column}{0.49\linewidth}
      \begin{itemize}
        \item One
        \item<2-> Two
        \item<3-> Three
      \end{itemize}
    \end{column}
    \begin{column}{0.49\linewidth}
      \begin{figure}
        \centering
        \includegraphics
        [width=0.95\linewidth]
        {
          image.png
        }
      \end{figure}
    \end{column}
  \end{columns}
\end{frame}
```

3.1.6 Sectioning and Outline Pages

`section` and `subsection` commands are given outside of the frames. Once this is done you can use a page (perhaps right after the title page):

```
\section{Introduction}
\begin{frame}
  \tableofcontents
\end{frame}
\section{On with the talk}
...
```

Maybe you want to remind your audience where you are in the talk. Certain themes have outlines built in; remember I added `secheader` to my theme. Additionally you can tell Beamer to revisit the outline at each new (sub)section and even highlight were you are! This is done outside of a frame and sometime just before when you want to begin this behavior (I always do it after the introduction section command and initial outline page; say just before the second section)

```
\AtBeginSection{
  \begin{frame}{Outline}
    \tableofcontents
    [currentsection]
  \end{frame}
}
%and maybe you also want
\AtBeginSubsection{
  \begin{frame}{Outline}
    \tableofcontents[
      currentsection,
      currentsubsection
    ]
  \end{frame}
}
```

3.2 Additional Documents

Beamer also allows you to create supporting documents. Sometimes it is useful to have handouts of your talk to give to the audience, or perhaps you need lecture notes for your students. Beamer can produce these all from the same (though possibly modified) source code.

Modes

Beamer now operates in several "modes" which are: "beamer" for the main presentation "handout" for the presentation but without incrementing slides etc., and "article" for lecture notes. There are a few other modes that are less common but I won't talk about them. There are also the umbrella "modes:" (mode-like commands that include more than one mode) "presentation" which is any mode except "article" and "all."

Using modes allows us to specify which commands one wants in different forms of the document. I will give specific examples later. Many commands can take mode options but the most useful is `\only<‘‘mode’’>` such as

```
\section<presentation>{Introduction}

\begin{frame}

Some text shown in all modes.

\only<article>{
  Text not shown on presentations.
}
\end{frame}
```

One final note is that I find that sometimes when switching between modes I have to recompile from scratch (remove aux files etc.) and run the compilers the full number of times or else a strange output may be obtained.

3.2.1 Beamer

We have already seen how to invoke beamer, however we may want to include the option `ignorenonframetext` to the documentclass in case we want to add additional text between slides in the article.

3.2.2 Handout

Handouts are created by passing the `handout` option (possibly still while ignoring the non-frame text) to the documentclass. Sometimes a few tweaks can be useful. I like adding in the preamble

```
\mode<handout>{
  \usepackage{pgfpages}
  \pgfpagesuselayout{4 on 1}
```

```
    [border shrink=3mm]
  \setbeamercolor{background canvas}
    {bg=black!5}
}
```

to cause there to be 4 slides per page in handout mode. One may also want to specify a different theme or color scheme as well.

3.2.3 Article (Lecture Notes)

Article mode is different than other modes. The documentclass is not `beamer` but `article`, and then load the package `beamerarticle`. Once this is done the beamerness of the document is ignored. I find that this mode needs the most tweaking from the original beamer source. One can write additional text between frames (see above) for the flow to be right and only do certain commands in this mode (i.e. I only use the caption in figures in article mode).

3.3 Impress!ve

Beamer creates incremental frames by having sequential pdf pages. This works just fine but doesn't have the fancy presentation style that a powerpoint might have. To reclaim some of this glory I use the presentation program Impress!ve (`http://impressive.sourceforge.net/`), rather than acrobat or other PDF viewer.

Impress!ve has several nice features:

- Transition effects
- Overview mode
- Spotlight or "spot-box" effect
- Cross-platform

The transitions (and other properties) can be controlled on a per-page basis by creating a ".info" file next to the ".pdf" file. I have created a perl script called "makebeamerinfo" (`http://code.google.com/p/makebeamerinfo/`) that creates this file for showing a clean but still interesting presentation.

Overview mode displays all the slides at one time on a grid, then one can click on a frame to go to that point in the presentation. This is especially useful during Q&A after the talk for going to the frame in question while still looking professional and remaining in presentation mode (as opposed to exiting to the user interface in powerpoint, or worse "clicking back").

As I said above, Impress!ve is cross-platform (written in python). In Linux (with python installed) the script itself is all that's needed. Mac will also work, but python may have to be installed or upgraded manually (as per the documentation page on the site). For Windows, where python is probably not installed (particularly if we are using a shared conference presentation laptop), the downloaded ".zip" file contains a stand-alone, working python as well as the script! This way Impress!ve can be run from the desktop or even your USB stick on any windows computer.

3.4 Posters with Beamer

Beamer has presented a window into the colorful and shapely side of LaTeX that is in large part due to the `pgf` package that underlies it as well as `TikZ`. If you imagine hard enough you might envision a large Beamer slide that is used as a poster (not unlike how most people make posters - powerpoint). This concept has been cotified into the (aptly named) package `beamerposter` (`http://www-i6.informatik.rwth-aachen.de/~dreuw/latexbeamerposter.php`).

A poster is made with some header material (see the examples as well as the website). Themes are the same though a few alterations or new styles might be useful (again see the website). The majority of the poster is then inside one `frame` environment, with the content then in `column` and if desired `block` environments.

4 UICThesi

Did you know that UIC has a document class for your thesis? The Math Department hosts it

(and presumably maintains it). Files and explaination are available at `http://www.math.uic.edu/graduate/current/uicthesi`. My guess is that the name is what it is due to the old DOS/Windows 8-character filename limit.

The commands are simple enough. Based on the article style, the only real additions (besides the formatting) are some extra commands for sectioning (i.e., `\dedication`). Reading throught the pdf of the UICThesi manual (uictman.pdf) will not only help you with UICThesi but also with the formatting and contents of a thesis for a Ph.D. here at UIC. It also contains some basic tutorial information for using LaTeX most of which is review for people at this level. Notice also that the uictman.pdf is itself formatted using UICThesi as a thesis (containing all the necessary elements).