LaTeX, it's not just for Academia - Part 2

Kevin O'Malley

CONTENTS

Note - This is an early draft of the article. See the O'Reilly MacDevCenter site for the final version.

I. INTRODUCTION

In the first article of this two-part series, you learned about LaTeX's most popular implementations for Mac OS X, and that it's not a word processor, but rather a document preparation and typesetting system. Under Mac OS X, there are many high-quality writing environments that simplify the process of composing LaTeX documents.

Before going any further, let me address some corrections/omissions from the first article. As a few readers pointed out, TeXShop and OzTeX can be setup to use Fink's UNIX TeX package. Saying "X11-based LaTeX implementations" may have mislead readers. TeX and X11 are not coupled in any way. You use X11-based DVI or PDF previewers to view LaTeX output. The file II2.dmg is not from the TeXShop folks, but rather is linked from their page. The program is available from the i-Installer Home Page. Finally, many readers have asked for more complete LaTeX sources. I have included these in the Resources section.

Now, let's see how you can use LaTeX to accomplish some common writing tasks. As you can imagine, LaTeX has many commands and features, and can be somewhat complicated to learn. Today, I'll keep things simple and concentrate on the basics.

This article introduces you to the LaTeX language, the basic structure of a LaTeX document, and shows some common writing tasks and how to accomplish them using LaTeX. Finally, I will demonstrate a few examples of common LaTeX documents and show how to generate these documents in different output formats, including formatting your documents for the web.

For these examples, you can use any LaTeX environment you wish - TeXShop, OzTeX, or the commandline version. At this point it may be helpful to briefly discuss the environment I use. I use TeXShop if I wish to use a Mac OS X Aqua-based program. Other programs have more feature, but I like the simplicity and stability of TeXShop.

For writing, I use GNU Emacs (21.2.1 from Fink) under Apple's X11. I do not rely on document previewing much, but when I need it, I do the following. I use a Perl script that looks at my LaTeX document every few seconds and calls an output generation program when the document changes; when I save the LaTeX document, the preview is automatically generated. I also have the script generate and display the current word count for the document.

Recently, I have been using Enrico Franconi Enhanced Carbon Emacs [http://www.inf.unibz.it/ franconi/mac-emacs]. This implementation was pointed out by a reader in the Trackback section. This Emacs implementation is based on the emacs-21.3.50 CVS distribution. Overall, I really like this version, which includes LaTeX tools such as AucTeX and RefTeX. In addition, it has good integration with Aqua-based previewing programs like Acrobat Reader and TeXShop.

The script generates output in PDF. To view the PDF file, I use TeXShop's external editor feature, which permits TeXShop to redisplay the PDF when it has changed.

I also use the Emacs' Flyspell mode to automatically check spelling as I type and suggests corrections.

Another way to generate your output is by using a Makefile.

Take a look at the Perl script and Makefile. These will give you examples of how to process a LaTeX document and generate various output formats.

II. LATEX PRELIMINARIES - COMMANDS, COMMENTS, AND WHITESPACE

Before we begin composing a document, it's helpful to know a bit about how LaTeX handles some common document elements. Like most programming languages, LaTeX commands are case sensitive. A LaTeX command is formatted as follows:

```
\command-name[optional-parameters]{parameter}
```

A LaTeX command begins with a backslash followed by the command name. Commands can have parameters, which are enclosed by curly braces () as well as optional parameters, which are surrounded by square brackets ([]).

Any line beginning with the % character is a comment. When LaTeX reads the % character in a file, it treats the rest of that line as a comment. In addition to the comment character, you can use the comment command.

```
\begin{comment}
This is a comment and is not displayed in the final document.
\end{comment}
```

III. LATEX DOCUMENT STRUCTURE

Now that you know some basics, lets move on to the structure of a LaTeX document. When you program in C++, Python, or Java, you usually structure source files in a certain way. Same holds for LaTeX. Here is the format of a LaTeX document.

```
% -- Begin LaTeX document.
% Begin document preamble.
\documentclass[options]{document-class-name}
\usepackage{package-name}
...
% End document preamble.
\begin{document}
% -- Your text and LaTeX commands go here.
\end{document}
% -- End LaTeX document.
end{verbatim}

A LaTeX document begins with the document class command. This command
specifies the class of your document, and consequently, what commands
and environments are supported and how LaTeX formats the
document. LaTeX document classes include article, report, and book, to
name a few. Think of the document class as defining the type of
writing you are doing as well as its appearance. For example, if you
are going to write an article, use the article class; a book, use the
book class.

Here are the most common LaTeX document classes.

\begin{itemize}
\item letter - for writing letters
\item article - for writing short reports, papers, and documentation
\item report - for larger writing tasks such as large reports having
  several chapters, or a theses
\item book - for writing books
\item slides - for preparing slides
\end{itemize}

The options parameter is used to change the settings for a document
class. For example, most classes slides, accepts the typeface size
options 10pt, 11pt, 12pt - 10pt is the default. Another option is the
paper size, which includes a4paper, a5paper, b5paper, letterpaper,
legalpaper, executivepaper - letter is the default.  You can also
specify the documents page orientation. For example, landscape tells
LaTeX to display the document in landscape mode.

Here are some example of common document classes.

\begin{verbatim}
\documentclass{letter}
\documentclass[11pt]{report}
\documentclass[12pt,landscape] {article}
\documentclass{slide}
```

You add extra functionally and commands through packages. Many packages come with LaTeX such as color, graphicx, and makeidx. In addition, you can add packages written by others, or even write your own for specific writing needs. Following the package command, you can define other commands that alter the formatting of your document.

The section of your document from document class to the begin document command is called the Document Preamble. The document preamble contains statements, packages, and other commands that set values and change the appearance of your document. LaTeX documents end with the end document command. Any text after this command is ignored.

The begin document command signals the beginning of your actual writing. Between the begin and end document commands is where you add your text and various LaTeX commands. What commands you can use is determined by the document class and the packages you include.

Before concluding this discussion, let's look at one more thing - the environment command. The environment command enables you to control the formatting and appearance of your document's content. For example, imagine you need to create a list of text, add an abstract, or display a table. For these cases, LaTeX provides environments that make it easy to accomplish these goals. You can even define your own environments, or redefine existing envoronments, with the new environment command.

IV. COMMON LATEX OPERATIONS

No mater what kind of document you write, there are always common operations that you need to perform. For example, when writing a report or an article, you need to divide your writing into sections, add tables, include footnotes, or possibly include graphics.

This section shows you how to perform many common operations in LaTeX. These suggestions are in no way inclusive, or enumerate all of the possibilities. The goal here is to get you started with some simple versions of commands. This discussion should be augmented with a good LaTeX reference.

The output for each example is displayed in a single output document. As you read about these commands, its a good idea to take a look at the example document's source file [Insert link to examples.tex] and output file [Insert link to examples.pdf]. Also, see the LaTeX source file for this article [Insert link to mac_osx_latex.tex].

A. Sections

LaTeX sections provide a mechanism for structuring your document. LaTeX supports several commands that you use to organize your document into sections.

- part
- chapter
- section
- paragraph
- subparagraph

To see sections in action, see the example and source files for this article.

B. Including Files

When working on large documents, it can be difficult to maneuver and edit a single file that contains all your text and LaTeX commands. Fortunately, LaTeX provides a mechanism for breaking large documents into smaller pieces. To accomplish this, you use the include and input commands.

The include command generates a page break and inserts the named file at the location of the command. The input command also inserts the file, but with no page break. The named file does not include the file extension, just the file name. For example, to include the file chapter.tex you would just specify the name chapter.

```
\include{file-name}
\input{file-name}
```

There are certain rules for doing this so consult the LaTeX documentation for more information.

C. Page Numbers

You control the appearance of page numbers with the page numbering command. LaTeX offers several options for numbering pages, including arabic, roman, Roman, alpha, and Alpha.

```
\pagenumbering{roman}
```

Page numbers are one of several commands for changing the appearance of pages. You can also change the margins, number of columns, add headers and footers to a page, and create a title page and abstract. See the Page Styles section of a LaTeX reference for more information.

D. Table of Contents/List of Figures/List of Tables

You create a table of contents in your document with the table of contents command. This command places a table of contents at the location of the command. LaTeX generates the table of contents entries from the section commands of the document.

LaTeX will automatically handle the numbering of sections and chapters for you. You can suppress adding sections to the table of contents, as well as section numbers, by using the starred version of the command.

```
\subsection*{My Title}
```

This is also useful if you are writing an article with no table of contents, and do not want numbers to appear for each section.

The list of figures and the list of tables command places a list of all figures and tables in your document. The list is inserted at the location of the command. If the caption command appears in the figure environment, that figure is added to the list of figures. If the caption command appears in the table environment, the table is added to the table list. The text of the caption is what's displayed in figure or table list.

To generate the initial list of figures, tables, and table of contents, and need to run the LaTeX command twice.

```
% latex latex-file
% latex latex-file
```

E. Generating an Index

If you are writing a reference document, a book, or a report, it's sometimes useful to include an index section at the end of the document. An index is a list of key words in the document and the page, or pages, where they occur. You add an index to your document with the makeinx package.

To create an index, you first include and enable the makeinx package in the document preamble. Next, add the index command to each word you wish to include in the index.

```
...
\usepackage{makeidx}
\makeindex
...
\begin{document}
As you learned in the first article, in LaTeX\index{LaTeX}, you can manage your bibliographic citations using 
...
\end{document}
end{verbatim}

Finally, you generate the index by running the following
programs (do not add the extension, just the file name).

\begin{verbatim}
% latex latex-file
% makeindex latex-file
% latex latex-file

or

% pdflatex latex-file
% makeindex latex-file
% pdflatex latex-file
```

Here's how it works. First, LaTeX processes the file and writes each index entry to a file, with a .idx extension. Next, makeindex creates the sorted index, which it stores in a file with a .ind extension. Finally, you run LaTeX again to place the index into the document.

You can create many different index formats by changing the syntax of the index command. See a LaTeX reference for more information.

F. BibTeX

As you learned in the first article, you manage your bibliographic citations using BibTeX. BibTeX is a separate program that you use to manage your bibliographic citations and merge selected citation information onto your LaTeX documents.

BibTeX solves some very common problems that arise when constructing a bibliography. For example, imagine you are writing a paper that includes many references and citations. Further imagine that the publication you are submitting to requires citation information to be formatted in a certain format; such as ACM or IEEE. Without BibTeX, you would need to maintain a list of your references and format each entry by hand in the required format. If you need to submit the paper to another publication, with a different citation format, you have to reformat each citation by hand.

BibTeX solves this problem quite easily. Here's what you need to do. First, maintain your citations in a format that BibTeX understands. You do this by formatting each citation in a defined format and placing it into a file with a .bib extension. You can place all citations in a single file, or break them up as you wish.

Here's an example of the format:

```
@misc{ wilson-ltxx,
  author = "Peter R. Wilson",
  title = "LTX2X: A LATEX to X Auto-tagger",
  url = "citeseer.nj.nec.com/wilson97ltxx.html" }
```

BibTeX identifies each record by its key. In the above example, the key is wilson-ltxx. This is the value you use to reference the citation within your LaTeX document.

As you can imagine, the format has many options and parameters. See the Resource section for more information. In fact, you can get many citations already formatted in BibTeX format at sites such as CiteSeer [http://citeseer.ist.psu.edu].

To create a citation in your text, use the cite command, with the parameter specifying the citation key. This command places a citation mark into the final text pointing to the reference and includes the reference in the bibliography. You can add a reference to the bibliography, without adding a reference mark to the main text with the no cite command.

To create a bibliography, add the following commands just before the end document command.

```
...
\bibliographystyle{plain}
\bibliography{sources}
...
\end{document}
```

The bibliography style command specifies the bibliographic style used to format each citation. The bibliography command instructs LaTeX to insert the appropriate citations into the document. BibTeX supports many bibliographic styles. See its documentation for more information.

To generate your bibliography, perform the following steps.

- Run LaTeX on the LaTeX file
- Run BibTeX on the LaTeX file
- Run LaTeX again on the LaTeX file
- Run LaTeX yet again on the LaTeX file

```
% latex latexfile
% bibtex latex-file
% latex latex-file
% latex latex-file
```

G. Footnotes

LaTeX supports the inclusion of footnotes in your text with the footnote command. The footnote command takes an optional argument, num, that you can use to change the default footnote number.

```
\footnote{Text for the footnote.}
```

In addition to the footnote command, you can also include footnotes with the footnote mark and footnote test commands. The footnote mark and footnote test commands are used in concert to place footnotes at the botton of a page.

H. Tables

Table are something that we all use in our documents. LaTeX enables you to create high-quality tables through the table and tabular environments. Using these environments, you can easily produce very well-structured and readable tables of information. The basic syntax for creating a table is as follows:

```
\begin{table}[where]
\caption{Table caption}
\centering
\begin{tabular}[pos]{cols}
 column 1 & column 2 ... & column k \\
 ...
\end{tabular}
\end{table}
```

Where LaTeX places a table is very important. For example, if you have a large table in your document, you would rather see it on a single page, rather than broken up across pages. In the LaTeX literature, you will see the term ”float’” or ”floating object” to refer to this idea. This describes the situation where a table or figure can not fit on its current page, and is placed on a separate so-called floating page.

LaTeX enables control over the placement of tables through the where parameter. The where parameter defines where the table is displayed on the page. A value of b places the table at the bottom of the page, h places the table here, t at the top of the page, and p on a separate float page containing no text, only floats.

The concept of floating objects also applied to figures and footnotes.

You use the tabular environment to construct the table. The pos and cols parameters control how the table is formatted. The pos parameter controls the vertical position of the whole tabular environment. The values are either t (align with top row) or b (align with bottom row). The cols parameter controls the column formatting; l = format text left, r = format text right, c = format text center. The p{wd} parameter controls the size of a column and makes columns with multi lines.

I. Figures

You add figures to a document with the figure command. Like tables, this places a figure at the location of the command, conditioned by the where parameter. Figures can also float over pages so make sure you read the description of floating objects in the Tables section.

You insert a figure as follows:

```
\begin{figure}[where]
\caption{Figure caption}
...
\end{figure}
```

J. Graphics

Most documents are composed of not only text, but also graphics. YOu insert graphics into a LaTeX file with the insert graphic command. However, you can not insert a JPG or PDF file into a LaTeX document, only EPS files. To insert other file formats like JPG, you have a few choices. One way is to convert your graphic file to EPS and then insert it into the document.

To convert the file you can use a Mac OS X Aqua-based programs, such as Graphics Converter, or install ImageMagick from Fink and use the following command:

```
% convert [jpeg-file] [eps-file]
```

You insert a graphics file into a document by first using the graphicx package and then using the insert graphics command..

```
\usepackage{graphicx}
..
\begin{document}
...
\includegraphics{mjh.eps}
...
```

To insert a JPG file directly, use the PDFLaTeX program to process your LaTeX document.

```
\usepackage{graphicx}
..
\begin{document}
...
\includegraphics{mjh.jpg}
...
```

This explanation is quite basic and there are far more issues to working with graphics in LaTeX. In addition to inserting graphics, LaTeX supports commands that enable you to draw picture elements such as boxes, trees, and curves in your document. Consult the Resource sections for more information.

K. Math

One of the major reasons for using LaTeX is its strong handling of mathematical equations and formulas. This is a complex subject so I'm just going to touch on the basics to give you a feel for how it works, and looks.

You insert mathematical symbols into your document by either surrounding the expressions with the $ symbol, or using the begin/end math commands.

The time-points x_i to x_j are embedded in the main text.

$(l_1 \leq x_j - x_i \leq u_i) \vee ... \vee (l_n \leq x_j - x_i \leq u_n)$

L. Lists

There are many times when you need to include a list of information in a document. To create lists, LaTeX provides the itemize, enumerate, and description environments. Each of these environments enable you to easily add different types of lists to your documents. All of these lists can be nested so you can produce various kinds of lists with different sub-list levels.

```
\begin{list-environment}
\item item 1 text
\item item 2 text
\end{list-environment}
```

V. EXAMPLES

Now that you understand some basic LaTeX commands, lets put it all together by looking at some sample LaTeX documents. We will look at two kinds of documents; an article and a slide presentation.

For these example, I will provide a sample document source file that you can use as a starting point for creating your own documents. Be advised - these are simple examples that contain minimal commands.

A. An Article

Composing papers and reports is something many of us do daily. For example, if you are a student, you need to take class notes and write papers for your classes. If you are a developer, you can write many kinds of project documents such as design documents and reference manuals.

Imagine you need to write a short paper for a class or maybe a series of articles. For this project you use the article document class.

See the LaTeX source file for this article [Insert link to mac_osx_latex.tex].

B. A Presentations

At one point or another, we have all had to give a presentation. Faced with this, most of us will reach for PowerPoint or Apple's Keynote. Both programs are excellent choices that enable you to quickly build very professional presentations. However, you can also create slides and computer-based presentations with LaTeX.

There are many advantages to using LaTeX for your presentations. First, LaTeX presentations require less disk space to store than their PowerPoint or Keynote counterparts. With disks being cheep these days, this is not much of an issue. But, I still like the idea of keeping file sizes down when I can.

Another reason for choosing LaTeX is if your presentation uses lots of mathematical equations.. If you use LaTeX, you can easily copy and paste these into your slides. Otherwise, you need to some way of getting them out of LaTeX and into your presentation software (maybe using tex2im or TeXShop).

Finally there's the simplicity factor. I don't know about you, but the world we live in has become preoccupied with appearance at the expense of content. I find it distracting to go to a talk and watch cute animations and graphics when plan old text would have been a lot clearer. Yes, a picture is worth a thousand words, but the picture does not need to fly on to the screen for the slide to be effective.

With LaTeX, there are many ways to create quality slides. The simplest is to use the LaTeX slides class, which comes with your LaTeX distribution. In the past, if you wanted to create slides you used SLiTeX. Today, this has been replace by the slide class. The slides class enables you to create simple slides with very little work.

There are many other ways to creating slides with LaTeX. Some of the most popular are FoilTeX, Prosper, Seminar, ifmslide, pdfslide, PPower4.

I create my presentations using a combination of PDFLaTeX, FoilTeX, and PPower4. My needs are quite basic, as you will see in the example slides [Insert link to slides.tex]. For a more complete solution, see "Creating Presentations in PDFLaTeX" by Matt Welsh (see the Reference sections).

VI. OUTPUT FORMATS

As you learned in the first article, a great reason for using LaTeX is that it can render your source document in many output formats, including PDF, Postscript, HTML, and RTF.

A. Putting Your Documents on the Web

These days, putting documents on the web is a common task. For example, imagine keeping notes in LaTeX and generating HTML so you can view them from anywhere. The best way to accomplish this is to install latex2html by Nikos Drakos (available from Fink). Once you have it installed, all you need to do is type the following command to generate a HTML version of your document.

```
% latex2html latex-file
```

The latex2html Perl script processes the specified LaTeX file, converts it to a set of HTML pages, and places the HTML files into a new directory, titled with the name of the file. There are lots of options for this script. One useful option is split (-split [n]). If n = -1, latex2html generates one large HTML file. If n == some number then that many pages are generated.

Here is an example of the HTML pages [Insert link to mac_osx_latex directory] generated from this article.

B. Converting to RTF

The latex2rtf program (available from Fink) converts a LaTeX file into RTF format. This provides some compatibility with users of other programs, such as MS-Word. In truth, the compatibility it affords is minimal and is not very useful for larger, more complex documents.

C. Converting to Text

In the first article, you were introduced to a program called DeTeX. DeTeX is a filter that removes LaTeX/TeX control sequences from its input. You use DeTeX to extract your text from a LaTeX document. DeTex has several options so be sure to read its man page.

Another option is to convert the document to a single HTML file, using latex2html. Then, open the HTML file in a web browser and save it as text.

D. Converting to MS-Word

An often asked question is can how I take a document I created in LaTeX and convert it to Word. The short answer is that I have not found any reliable way to do this. There are some strategies you can use such as converting the LaTeX document to text (using detex) and importing it into Word, but all formatting is lost. Another is to convert to RTF, and import the RTF to Word; but, this will also lose some formatting information. Another is to convert to HTML and then import to Word, but again, you will lose formatting information.

If anyone has discovered a reliable way to handle this, please post it using Talkback.

VII. LATEX INTEGRATION WITH MAC OS X APPLICATIONS

Many Mac OS X applications support integration with LaTeX. As you saw in the first article, BBEdit supports a user-contributed glossaries of LaTeX commands. OmniOutliner (http://www.omnigroup.com/applications/omnioutliner/) from The Omni Group is a Mac OS X application for creating many different kinds of lists and outlines. OmniOutliner LaTeX Export (http://www.opendarwin.org/ landonf/software/Omni-LaTeX/) is a set of AppleScripts by Landon Fuller that export an OmniOutliner document to LaTeX format.

VIII. CONCLUSION

I hope you've enjoyed this brief tour of LaTeX on Mac OS X. As you've seen, Mac OS X supports a rich set of LaTeX environments and implementations, making it a formidable platform for composing LaTeX documents. If you use a word processor for your writing, using LaTeX will be a shift in focus, and will probably require some time to learn. But, as these articles have pointed out, there are many advantages to using LaTeX.

In any event, I hope these articles have inspired you to begin using, or return to using, LaTeX. Good luck and as always, if you discover something interesting, post it using Talkback or send me an email.