LaTeX Guidelines for Simple, Two-Column Papers (Without worrying about figure formats)

Edward A. Lee, Jonathan Sprinkle

EECS 290N LaTeX Tutorial
September 23, 2004

University of California, Berkeley
Berkeley, CA, 94720, USA

{eal,sprinkle}@eecs.berkeley.edu

Abstract

This is a simple sample of a document created using LaTeX that includes a figure from the Vergil visual editor for Ptolemy II. It also provides an example to follow for the installation of figures so that either DVI/PS or PDF files can be generated using the appropriate command in LaTeX. Additionally, this file illustrates a simple two-column conference paper style, and use of bibtex to handle bibligraphies. If you can compile this file in your LaTeX environment, then you should be able to create your own TeX documents for the project paper.

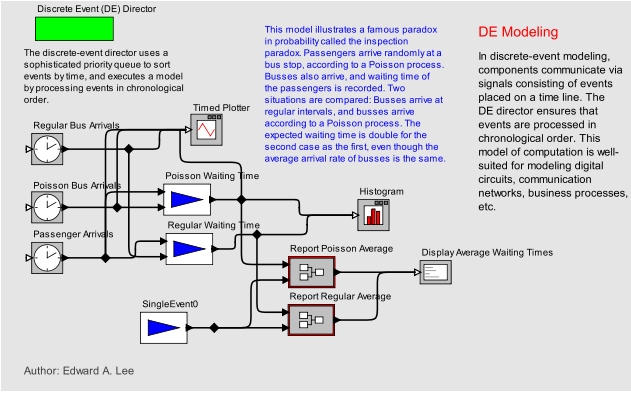

Figure 1. Figure caption. To get a figure to span two columns, use the environment figure* rather than figure.

1 Why use LaTeX?

One advantage that WYSIWYG[1] word processors have is that, well, what you see is what you get. Those of us who can remember typing term papers out on IBM Selectrics (or worse, Remington Manuals . . .) will immediately point out that those, too, were WYSIWYG document creation tools. The difference between them is that a typewriter is more predictable than a computer, and has to be rebooted less. Oh, and also that a typewriter is only as flexible as its spacing complexity, and interchangeable typefaces, and may not have justification built in, and (well, you get the idea).

One major frustration found in WYSIWYG editors such as Microsoft Word is that often it is hard to make the document look like you want it to in the first place. Examples of this are captions that don't stay with figures, numbering routines that inexplicably shift to the middle of the page, or magically change into Roman (or roman) numerals add unforeseen times, and (of course) the phantom **Error! Reference Source Not Found** that is most likely to appear when one is printing the final version of a thesis onto expensive rag paper, or just after the camera-ready copy of a conference paper has been submitted one minute before the deadline.

LaTeX is a very old (by word processing standards) program that attempts to address problems such as these for documents that must appear as if they had been typeset. The program TeX was designed to serve as an electronic typesetting program. It provides, however, some macro capabilities, which is where LaTeX enters the picture. LaTeX is

[1] What You See Is What You Get

designed to be a WSEFIWYG[2] word processing program, which uses TEX as its execution semantics, but provides high-level domain-specific commands for authors of *technical documents*.

Note that we emphasize *technical documents*. While you may have just now suddenly realized the great pun that LATEX makes, more importantly you should realize that when composing a letter to your grandmother, or perhaps an e-mail to your significant other to ask whether lunch is in order, LATEX may not actually be the right tool for you to use. It is designed to simplify the work required for a large number of people to produce a format-restricted and/or technically rich document that must conform to certain standards or be reformatted by another program. For these purposes, LATEX is far superior to Word. However, a great deal of time (or experience) is required to produce the formatting commands that make LATEX useful as a processor, so before you decide to rewrite your resumé just for fun, be warned that you are probably in for a long ride.

2 Creating DVI, PS, and PDF outputs from LATEX Source

This sample document will directly produce the DVI, PS, and PDF output formats from LATEX sources, when compiled in conjunction with the `figure.ps` and `figure.eps` file. To run LATEX on this file, you need the following files:

1. `templateUniversal.tex` (this file)
2. `figure.{ps,pdf}` (the appropriate figure file)
3. `simpleConference.sty` (style file)
4. `refs.bib` (bibiliography file)

DVI and PS output both require a certain set of figure types—which unfortunately do not correspond to those for PDF output. This is explained in a following section. For the iterative writing of a paper, it can be useful to generate DVIs over and over, rather than PDFs, since they are very easy to view using previewers such as YAP (which ships with MiKTEX).

2.1 Using an IDE

IDEs such as TeXnicCenter [1] are useful in reducing the amount of hacking required to produce a paper in LATEX. These IDEs are usually a thin frame over the TEX execution, providing an editor (of varying capability) along with abilities to rapidly compile DVI/PS/PDF formats with little or no hassle, and generally doing the busywork of hooking in with the TEX implemenation (e.g., MiKTEX) in use on your computer.

While it is possible to create a LATEX source file using vi or Emacs, if you are struggling (or new) in the world of *nix, or you generally prefer the look and feel of an IDE such as Visual Studio or Eclipse to do your regular run-of-the-mill programming, then you will appreciate LATEX IDEs as well. Refer to their documentation for the finer points of TEX execution.

However, if you maintain a masochistic flair—or if you have access to a LATEX installation only through a command line interface, such as a *nix server—please refer to the following section to learn how to use the command line interface for LATEX execution.

2.2 Using the command line

An IDE takes care of the different kinds of formats that you might want to produce by actually executing the following commands. You can do these yourself (but with a greater likelihood of typo) by following these directions.

2.2.1 Creating PDF output

To create a PDF file, execute the following commands:

```
% pdflatex templateUniversal
% bibtex templateUniversal
% pdflatex templateUniversal
% pdflatex templateUniversal
```

Yes (strangely) it is necessary to run pdflatex three times. The result will be a PDF file (plus several other files that LATEX produces). You will need a mechanism, of course, for executing commands on the command line. If you are using Windows, I recommend installing Cygwin and using its bash shell.

2.2.2 Creating DVI output

To create a PDF file, execute the following commands:

```
% latex templateUniversal
% bibtex templateUniversal
% latex templateUniversal
% latex templateUniversal
```

As noted in Section 2.2.1 it is necessary to run pdflatex three times. The result will be a DVI file (plus several other files that LATEX produces) which renders as close to the actual print as required for basic proofreading. An advantage to

[2] What Someone Else Formats Is What You Get

generating a DVI (as opposed to a PDF) is that it generates faster, and using programs such as YAP, you can actually update the file while you have it open. This reduces your headache in recovering from "Error: output file not writeable" errors.

2.2.3 Creating PS output

To create a PS file, you must first create a DVI as explained in Section 2.2.2. Once this file is created, the next step is to run

```
% dvips templateUniversal.dvi
```

3 Using Figures

When generating PS and DVI outputs, the most popular image format is EPS (Embedded PostScript). This format is easily obtained by printing to a file on a PostScript printer, and setting the bounding box of the graphic using a tool such as GhostView. PDF output is unfortunately unable to use EPS as a source for images; its most common image format is PDF.

The question then arises: are you forced, at the writing of a document, to decide whether you want your final output to be PDF or PS? The answer is: yes (if you want to), and no (if you are clever).

Actually, it is LaTeX that is clever when it comes to this, and rightly so, since it is LaTeX that created this problem in the first place by not creating a way for multiple image formats to be used by the same output formats. It turns out that if you include your images without an extension, that the commands will figure out which images to use based on whether they are generating PDF or DVI formats.

For example, this document uses the file `figure`.

3.1 How to Include Vergil Diagrams as Figures

Suppose you wish to include a figure, like that in figure 1. The simplest mechanism is to install Adobe Acrobat, which includes a "printer" called "Acrobat Distiller." Printing to this printer creates a PDF file, which can be included in a document as shown here. To include Ptolemy II models [2], just print to the distiller from within Vergil and reference the PDF file in your LaTeX document.

There is a bit more work to do, however. The file that is produced by the distiller represents a complete page, not the individual figure. You can open it in using Acrobat (version 5.0 or later), and select Document → Crop Pages from the menu. In the resulting dialog, check "Remove White Margins." Save the modified PDF file in a file and then reference it in the LaTeX file as shown in this example.

An alternative is to generate EPS (encapsulated postscript), but the process is much more complex and fragile. I recommend using pdflatex and Adobe Acrobat.

References

[1] TeXnicCenter (IDE for writing LaTeX papers). Technical report, `http://www.toolscenter.org/`.

[2] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng. Heterogeneous concurrent modeling and design in java. Technical Report Technical Memorandum UCB/ERL M04/27, University of California, July 29 2004.