Every LaTeX document brings new$\overset{to\ me}{\wedge}$programming issues

David Walden

Draft of June 5, 2014

[*This is a draft which will undoubted change by the time I prepare and give my TUG2014 presentation; I will submit a revised version for the conference proceedings.*]

There is seldom a time I am not composing a document drafted in LaTeX. Each document brings its own style and efficiency issues and, thus, each time I seem to have to solve a new little LaTeX programming problem.

A natural question might be, "Why not find an existing package that does what you need rather than coding your own thing?" Of course, sometimes what I need may be exist in an existing package. However, mostly I am too lazy to use an existing package if I can't immediately find one that meets my needs; or I find one but it doesn't install and work without difficulty and without much study. I'd rather code my own little thing than struggle with package installation issues or inter-package interface issues. I'd rather do my own little thing (even if it is less efficient than what exists) to avoid having to understand complex documentation.[1]

LaTeX is swell because it is programmable such that I can create little "tools" that help me do what I want to do. Also, like any experienced computer programmer, I collect these little solutions for reuse in future documents by copying rather than new thinking.

In this presentation I give three examples of such little programming problems and the (perhaps quick-and-dirty) solutions I arrived at:

- Issues and ways for typesetting ellipses
- Blank verso sides without using the book style twoside option
- Flexible layout out a photo album

All three examples here derive from my work on which I have written before — self- or private-publishing.[2]

[*I'd be glad to hear about better approaches for doing "out of the box" the things I describe below, especially if the "box" is trivial to install and configure. I'd be happy to include the better approaches (and words of critique of my approaches) in the conference proceedings version of this paper. That would be educational for me and could be for others.*]

1 Typesetting ellipses with LaTeX

There are many situations and approaches for using ellipses in LaTeX. After sketching some of the myriad situations and a few of the approaches, I describe what I do.

1.1 Diversity of situations

I am mainly concered with the use of ellipses in American English, non-math writing. In this context, ellipses seem to have two purposes: (1) indicating where something is has been

[1] Perhaps I am a bad guy, but I lack motivation for developing my little tools into packages that might help others (although I certainly appreciate that others develop packages that help me).

[2] Self-publishing: Experiences and opinions, `http://tug.org/TUGboat/tb30-2/tb95walden.pdf`

left out of quoted text; (2) to indicate a pause or something never stated — an unfinished thought or an implicit thought ("and so on").

Here are three examples (in which I have not tried to perfect the typesetting of the ellipses):

1. "Four score and seven years ago our fathers brought forth ... a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal."
   The words "on this continent" have been left out.

2. My wife may think that I am fussy about little things...
   The ellipsis here might indicate that I have a lot more to say about this subject that will go unspoken.

3. "Leave me alone ... I'm to tired to talk about it," he said.
   If the quoted words are in dialog, the ellipsis might indicate a pause in the speech.

Ellipses are complicated for at least three reasons:[3] (1) they can have many different uses; (2) there are different sets of conventions for how to indicate the elision, for example, always using three dots or sometimes using three and sometimes using four dots depending on context within sentences; (3) there are different approaches for typesetting ellipses.

Here are some of the possible contexts of ellipses use:

- at the end of a sentence
- within a sentence
- at beginning of a line
- at the end of a line
- within a line
- with other punctuation after the ellipsis
- with other punctuation before the ellipsis
- without other punctuation before or after the ellipsis

Lots of combinations of those and other situations also can happen.

1.2 Different conventions

On pages 82–83 of his *The Elements of Typographic Style*,[4] Robert Bringhurst gives a sketches covering some of the conventions for using ellipses:[5]

> Most digital fonts now include, among other things, a prefabricated *ellipsis* (row of three baseline dots). Many typographers nevertheless prefer to make their own. Some prefer to see the three dots flush ... with a normal word space before and after. Other prefer ... to add *thin* spaces between the dots. Thick spaces (M/3) are prescribe by the *Chicago Manual of Style*, ... In most cases the Chicago ellipsis is much too wide.
>
> Flush set ellipses work well with some fonts and faces but not with all.... At small text sizes ... it is generally best to add space ... between the dots. Extra space may also look best in the midst of light, open letterforms, ..., and less space in the company of a dark font, ..., or when setting in bold face....

[3] While in this note I only discuss non-math use in American English, it provides enough variety of situations and problems to perhaps suggest things to think about when using and typesetting ellipses in another language.

[4] Version 3.1, Hartley & Marks, Publishers, Vancouver, BC, 2005

[5] The first two instances and the last instance of an ellipsis in this quote are part of Bringhurst's text. The rest of the ellipses are by me to indicate my elisions from the Bringhurst quote.

> In English ..., when the ellipsis occurs at the end of a sentence, a fourth dot, the period, is added and the space beginning the ellipsis disappears.... When the ellipsis combines with a comma, exclamation mark or question mark, the same typographic principle applies. Otherwise a word space is required fore and aft.

However, the Bringhurst summary leaves out some common conventions. For instance, pages 292–296 of my copy of *Chicago Manual of Style*[6] starts by giving two main conventions for the form of ellipses: (1) always only three dots, and (2) four dots at the end of sentences (or three dots and another punctuation mark, as described by Bringhurst) and three dots elsewhere. The latter is the manual's preferred convention. Ellipses in block quotes also provide additional circumstances beyond those mentioned in Bringhurst's sketch.

The are a number of useful on-line discussions of the use of ellipses, for example,

- in the Wikipedia: `http://en.wikipedia.org/wiki/Ellipsis`
- in Doc Scribe's Guide to research styles, where you can lookup the approaches recommended in five well known style guides (AMA, APA, ASA, Chicago, and MLA): `http://www.docstyles.com/`

1.3 Standard tools versus hand crafting

It seems to me that just using `\dots` or `\ldots` in LaTeX is often not enough to address some of the potential needs mentioned in the previous sections.[7]

Also, naive use of `\dots` apparently has a problem that Peter Heslin's ellipsis style (`http://www.ctan.org/tex-archive/macros/latex/contrib/ellipsis/`) works on fixing. As Heslin says,

> There is a problem in the way LaTeX handles ellipses: it always puts a tiny bit more space after `\dots` in text mode than before it, which often results in the ellipsis being off-center when set between two other things.

It is worth reading the documentation of Heslin's package,[8] which also describes some of the issues relating to using ellipses. The package also allows one to specify the Chicago or MLA style and to specify the spacing between dots (e.g., in terms of an em) in an ellipsis.

Another package is `lips.sty` (`http://www.tex.ac.uk/ctan/macros/latex/contrib/frankenstein/lips.sty`), which Heslin suggests one use if one wants the full Chicago style.

With so many possibilities and needs, it is not surprising that, as Bringhurst says, "Many typographers nevertheless prefer to make their own." It is hard for me to imagine a package with sufficient capabilities and options for everyone. (But maybe I'm wrong.)

1.4 My approach

My approach has been to define a few macros to handle common situations for using ellipses in *the writing I do*. These also implement my own preferences, such as for inter-dot spacing and spacing before and after an ellipsis. I have used a couple of versions of these.

[6] I'm looking at the 13th edition and not a later edition.

[7] Also see the appendix.

[8] `http://www.ctan.org/tex-archive/macros/latex/contrib/ellipsis/ellipsis.pdf`

Version 1

The following definitions were sufficient for the *Breakthrough Management* book I co-authored and typeset. I used the Minion typeface for this book. The comments in the follow code provide the relevant explanations.

```
  %dots for main text
\def\bigdotsspace{3pt}

  %three dots
\def\mydots{\hbox{\hspace{\bigdotsspace}.\hspace{\bigdotsspace}.
\hspace{\bigdotsspace}.\hspace{\bigdotsspace}}}
  %I like the same size space on each side of the ellipsis as
  % is between the dots of the ellipsis

  %period and three dots = four altogether
\def\fmydots{\hskip0pt{}\hbox{.\hspace{\bigdotsspace}.
\hspace{\bigdotsspace}.\hspace{\bigdotsspace}.\hspace{\bigdotsspace}}}
  %And the same size space after a period and before 3 dots

  %dots with only beginning space -- no following space
\def\mydotsnfs{\hbox{\hspace{\bigdotsspace}.\hspace{\bigdotsspace}.
\hspace{\bigdotsspace}.}}
  %I used this def with an ellipsis and following comma, etc.

  %dots for block quote text, which is smaller than main text
\def\smalldotsspace{2pt}

  %three dots without end spaces
\def\minsmalldots{\hbox{.\hspace{\smalldotsspace}.
\hspace{\smalldotsspace}.}}

  %three dots -- small my dots
\def\smydots{\hbox{\hspace{\smalldotsspace}\minsmalldots
\hspace{\smalldotsspace}}}

  %period and three dots = four altogether
\def\fsmydots{\hskip0pt{}\hbox{\hspace{.3pt}.\hspace{\smalldotsspace}
\minsmalldots\hspace{\smalldotsspace}}}
```

Version 2

I used the following set of definitions with the book I compiled and typeset about the technology history of the company Bolt Beranek and Newman[9] and that I am now typesetting. This book uses the Lucida Bright typeface. For this second book, I had learned about doing things in terms em which varies with font size, e.g., among main, footnote, and block quote text. My decisions in the following definitions are only about what looks good to me, not about the conventions of a particular style manual. (The comments in the following code provide the relevant explanations.)

```
\def\fourdots{\hbox{.\hspace{.33em}.\hspace{.33em}.\hspace{.33em}.\,}}
   %with an end-of-sentence ellipsis I use a thin, not word, space
```

[9] *A Culture of Innovation: Insider Accounts of Computing and Life at BBN*, David Walden and Raymond Nickerson, editors, Waterside Publishing, 2011, `walden-family.com/bbn/bbn-print2.pdf`

```
\def\fourdotstightright{\hbox{.\hspace{.33em}.\hspace{.33em}.\hspace{.33em}.}}
   %sometimes I don't want the trailing thinspace, e.g., at end of line

\def\threedots{\hbox{\,.\hspace{.33em}.\hspace{.33em}.\,}}
   %for a non-end-of-sentence ellipsis

\def\threedotstightleft{\hbox{.\hspace{.33em}.\hspace{.33em}.\,}}
   %for beginning of line, e.g., in a block quote

\def\threedotstightright{\hbox{\,.\hspace{.33em}.\hspace{.33em}.}}
   %for the end of line, e.g., in a block quote

%\def\sentencespace{\unkern\spacefactor=3000 \space\ignorespaces}
   %tried this but didn't use it

%\def\fourdots{\unskip\kern\fontdimen3\font.\kern.1667em\ldots\sentencespace{}}
   %tried this but didn't use it

%\def\threedots{\unskip\ \ldots\unkern{}}
   %tried this but didn't use it

\def\fnfourdots{\fourdots{}}
   %for use in a footnote -- I originally thought I might need
   %  a different definition, but then I was happy with the
   %  main text ratios

\def\fnthreedots{\threedots{}}
   %ditto
```

Summary

It is easy to see how my set of definitions could be adapted to using word or sentence spaces before and after an ellipsis while using some other appropriate inter-dot spacing, or to adapt the definitions to other traditional or personal conventions. For instance,

```
\def\fourdots{\hbox{.\hspace{.33em}.\hspace{.33em}.\hspace{.33em}}. }
   %sentence space after four dots
```

I am also sure that there are better approaches than mine for handling a variety of ellipsis situations in LaTeX, or at least better ways to do what I am doing (perhaps automatically detecting whether an ellipsis is at the beginning or end of a line and thus eliminating the need for those definitions).

1.5 Appendix for this section

By the way, in the file `latex.ltx`, I found the following definitions.

```
\DeclareTextCommandDefault{\textellipsis}{%
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font}

\DeclareRobustCommand{\dots}{%
   \ifmmode\mathellipsis\else\textellipsis\fi}
```

```
\let\ldots\dots
```

2 Blank verso sides without using the book style `twoside` option

Several years ago I was involved in creating a small book (approximately 100 pages[10]), and a year later I did the LaTeXing of a small pamphlet (approximately 60 pages[11]). In both cases, the document needed to look like a book, but using all the built-in capabilities of the book class wasn't necessary. Therefore, I drafted my own class file (which, in the first case, Karl Berry significantly improved) and loaded that on top of the standard book class, e.g.,

```
\documentclass{book}
\usepackage{ctssbook}
```

Naturally, the added style file included a macro, `\beginnewchapter`, which reset the various counters (such as footnote and figure numbers), formatted the chapter title, changed the running headings, and to put the chapter title in the table of contents using the command

```
\addcontentsline{toc}{chapter}
 {\protect\fmttocnumber{\thechapter}#1}
```

where #1 is the chapter title passed to the macro via the macro call (and `\fmttocnumber` is a macro that formats a right justified chapter number in a properly sized field).

Because the command `\chapter` is never given in the LaTeX for these two books, the two-side and one-side capabilities of the book style aren't available. This is not a problem. For a document that will be printed and needs to start new chapter on a recto side, it is easy enough (in the last stages of typesetting) to, first, perfect the page breaks: for this I typically use calls to macros such as

```
\newcommand{\Lpushlines}[1]
 {\enlargethispage{-#1\baselineskip}}
\newcommand{\Lpulllines}[1]
 {\enlargethispage{#1\baselineskip}}
```

And then, second, to go through the root file of the document and add a macro call (after the commands to input the content of chapter, frontmatter and backmatter files) to create the necessary blank verso sides where needed. This end-of-chapter macro definition is something like

```
\def\EOC{\newpage

 \null\thispagestyle{empty}\newpage
}
```

and the resulting root file looked as follows:

```
\documentclass{book}
\usepackage{ctssbook}
\begin{document}
```

[10] Karl Berry and David Walden, editors, *TEX's 2^5 Anniversary: A Commemorative Collection*, TEX Users Group, Portland, OR, 2010, `http://tug.org/store/tug10/`

[11] David Walden and Tom Van Vleck, editors, *Compatible Time-Sharing System (1961–1973): Fiftieth Anniversary Commemorative Overview*, IEEE Computer Society, Washinton, DC, 2011, `http://www.walden-family.com/ieee/ctss.pdf`

```
\frontmatter
\include{title-pages}
\include{preface}
\mainmatter
\include{history}
\EOC
\include{toms-webpage-r1}
\EOC
\include{uses-r}
\include{views}
\EOC
\include{other}
\backmatter
\include{biblio}
\EOC
\input{colophon}
\EOC
\end{document}
```

In the pamphlet shown in this example, some chapters already end on verso sides and calls to `\EOC` are not needed. Also, the command `\tableofcontents` is included in the `title-pages` file; and, since the table of contents in this case is only one page long, it also includes a call of `\EOC`.

As I was finishing this pamphlet, I needed PDFs both for sending to the printer and for posting on the web. For the printer, there needed to be two PDFs: one PDF for the color cover (i.e., a single file of the back cover, spine, and front cover), and one PDF of the grayscale interior of the pamphlet including blank pages at the end of chapters as needed to start chapters on recto sides. For the web, I needed a single PDF with the front and back covers at the beginning and end of the interior pages, and I decided I wanted to leave out blank verso sides of the interior but keep the same page numbers as in the print version.

Thus, I created a macro to conditionally add the covers to the interior and augmented the `\EOC` macro to add blank verso sides only when needed for the print version.

```
\def\Forweb{0} %0 = print
%\def\Forweb{1} %1 = web

\RequirePackage[final]{pdfpages}
\def\Covers#1{\ifodd\Forweb
 \includepdf[pages=1-1]
 {#1.pdf}\fi} %#1 is cover filenames

\RequirePackage{ifthen,changepage}
%if for web, increment page counter
%if for print, output blank page
\def\EOC{\newpage\checkoddpage
 \ifthenelse{\boolean{oddpage}}{}
 {\ifodd\Forweb\stepcounter{page}
 \else\null\thispagestyle{empty}
 \newpage\fi}}
```

Thus, I added `\Covers` macro calls bracketing the rest of the document as follows:

```
\begin{document}
\Covers{front-cover}
...
```

```
...
\Covers{back-cover}
\end{document}
```

I also then include a call to the revised `\EOC` macro after including *each* chapter, frontmatter, and backmatter file (the `title-pages` file already contained a call `\EOC`).

3 Flexible layout out a photo album

This past year I decided to print a dozen of so copies of an album of old photos to distribute to family members. The photos had been pulled from several photos albums of a deceased parent that were broken up and various photos of individuals sent to the individual or a family member of the individual. However, some photos needed to go to more than one person; hence, I decided to create an album of these remaining photos which could be distributed to multiple family members.

The photos came in a variety of sizes ranging from 8x10 inches to smaller than 3.5x5 inches, many of the sizes being non-standard for today's typical digital printing businesses that serve amateur photographers (they tend to assume 8x10, 5x7, 4x6, and 3.5x5). The photos also came in a variety of conditions from quite good to quite bad (faded or otherwise discolored, or never high quality in the first place). I scanned all of these photos at either 600 pixels per inch or 300 ppi, cropped off the borders on the digital version, and then did lots of Photoshopping to bring as much quality back to the images as I could manage. Because I thought it might be useful, I put the images into six separate directories for 8x10 (the few instances of this only had a vertical orientation), 5x7 (also the instances all were vertically oriented), 4x6 tall orientation, 4x6 wide orientation, 3.5x5 tall orientation, and 3.5x5 wide orientation.

I decided that I wanted the photos in the album I was creating to be the exact size of the originals in inches on the printed page. Consequently, I decided the albums trim size when perfect bound would be 10x11.5 inches. The next step was to figure out how to layout the photos on printed pages.

3.1 First effort

The first macro I wrote,`\image`, took two arguments, an image directory/file-name and a draft caption (the file name without its directory), and displayed the image with the caption beneath it at the current location. Then I wrote three other macros:

1. `\oneperpage`, which called `\image` once and centered the specified image and its caption on a page

2. `\sidebyside`, which called `\image` twice and placed the two images side by side, centered on a page

3. `\overunder`, which called `\image` twice and placed the two images (and their captions) on the page centered horizontally and spaced out equally in the vertical direction

I wrote a Perl program to generate (in alphabetical order by file name) calls to the three page-layout macros for all the image files in each of the six directories, with the 8x10 and 5x7 images being place alone on pages, the 4x6 and 3.5x5 tall images place side by side on a page, and the 4x6 and 3.5x5 wide images place in an over-under position on the page. With a little manual text editing, the macro definitions and the Perl-generated calls to the

page-layout macros became the LaTeX program to generate a first draft album of all of the images.

However, there was a problem. My intention was to print the images at the actual size of the scanned photographs, counting on `\includegraphics` to read the metadata in the image file to specify the print size. This worked well for most of the images. But for some of images in the 3.5x5-inch-tall directory, the images printed at much too big a size. Rather than sort out the reason, I chose the brute force course of temporarily changing the definition of `\image` so `\includegraphics` used a width of 3.5 inches in calls by the `\sidebyside` macro.

With the printout of this first draft in hand, I was able to begin to improve the captions and to begin to seriously think about layout issues. With actual captions written, many were wider than their image which didn't look good on horizontal images and which was completely broken on side-by-side images. So I redefined `\image`, as follows, to measure the width of the image and have the caption be that width:

```
\def\image#1#2{
\centerline{\includegraphics{#1}}
\smallskip

\settowidth\imagewidth{\includegraphics{#1}}
\begin{minipage}[b]{\imagewidth}

\centering
\large#2
\end{minipage}

}
```

3.2 Next approach

With the new (to me) concept of measuring the image width and adjusting the caption width accordingly, I redid the page layout macros.

The `\overunder` macro could use the new definition of `\image` directly as shown in the following definition and example call:

```
\def\overunder#1#2{

\clearpage
\vspace*{\fill}

#1

\vfill

#2

\vfill
\clearpage
}
```

and

```
\overunder{\image{<filename>}{>caption>}}{\image{<filename>}{>caption>}}
```

I redid the `oneperpage` and `sidebyside` macros to use the `\imagewidth` approach without calling the `image` macro, e.g.,

```
\def\oneperpage#1#2{
\clearpage
\vspace*{\fill}

\centering
\includegraphics{#1}

\medskip
\settowidth\imagewidth{\includegraphics{#1}}%
\begin{minipage}[b]{\imagewidth}
\centering
\large#2
\end{minipage}
\vfill
\clearpage
}

\def\sidebyside#1#2#3#4{
\clearpage
\vspace*{\fill}
\centerline{\includegraphics{#1}\quad\includegraphics{#3}}

\settowidth\imagewidth{\includegraphics{#1}\quad\includegraphics{#3}}
\smallskip
\begin{minipage}[b]{\imagewidth}
\centering
Left: #2\\Right: #4
\end{minipage}
%\centerline{Left: #2; right: #4}
\vfill
\clearpage
}
```

Notice that by this time I had created inconsistency in how I included images: sometimes

```
\macrocall{filename1}{caption1}{filename2}{caption2}
```

and sometimes

```
\macrocall{\macrocall{filename1}{caption1}}{\macrocall{filename2}{caption2}}
```

3.3 Final approach

At this point, I concluded I needed to do several things differently:

1. try top justifying side-by-side images rather than bottom justifying them as happened without doing anything special
2. fix the page layout macros so they called in images in a consistent way
3. make it easy to move around the calls of image-caption pairs in the overall LaTeX file for the album

Regarding the first point above, top justification of side-by-side images, I looked at or tried four different methods, three from a question and answer on `tex.stackexchange.com` (`http://tex.stackexchange.com/questions/101858/make-two-figures-aligned-at-top`) and

one I made up myself. One of the `tex.stackexchange.com` suggestions didn't seem quite relevant and I couldn't manage to install and use the suggested packages in the other two suggestions there. The method I tried to develop myself was to measure the height of the images, find the difference in heights as a positive number, and build some vertical space of the size of that different to put under the shorter of the images; unfortunately, I couldn't get the units of the various parts of these calculations to match well enough to make the method work. After several hours of trying things spread over a couple of days, the bottom justified approach began to look better and better, and I gave up trying for top justification.

Regarding the second point above, consistent calling sequences, it came to me that dealing with the third point above, moving around image-caption pairs, would naturally address the second point.

Regarding the third point above, it seemed to me that the best approach was to separate specifying images and their captions from the page layout macros, that is, to not have the page layout macros call the image-caption specification macros. My idea was to allow something like the following:

```
\specifyanimage
\specifyanimage
\layoutpagewithsidebysideimages

\specifyanimage
\layoutpagewithsingleimage

\specifyanimage
\specifyanimage
\layoutpagewithoverunderimages
```

Then I could simply drag the `\specifyanimage` macro calls around to the places I wanted them to be in my LaTeX source file for the album, although I would still have to be aware of what size images could fit within the bounds of a page.

For the `\specifyimage` macro I developed the following macro (Karl Berry pointed out the `\ifcase` TeX language construct to me):

```
\newcounter{savedphotocount}     %define counter
\setcounter{savedphotocount}{0}  %clear counter

\def\savephoto#1#2{%
  \stepcounter{savedphotocount}%
  \ifcase\value{savedphotocount}
      \errmessage{case zero should never happen}%
  \or \gdef\photoa{#1}\gdef\captiona{#2}% case 1
  \or \gdef\photob{#1}\gdef\captionb{#2}% case 2
  \or \gdef\photoc{#1}\gdef\captionc{#2}% case 3
  \or \gdef\photod{#1}\gdef\captiond{#2}% case 4
  \else \errmessage{trying to save more photos than expected!}
  \fi
}
```

This macro saves up to four images in well known places from which the page layout macros can use them; obviously, this macro could have been extended to save more images between instances of zeroing the `savedphotocount` which was done at the end of each page layout macro.

Below is an example definition of one of the page layout macros that used `\savephoto`.

```
\def\oneperpage{
\clearpage
\vspace*{\fill}

\centering
\includegraphics{\photoa}

\medskip
\settowidth\imagewidth{\includegraphics{\photoa}}%
\begin{minipage}[b]{\imagewidth}
\centering
\large\captiona
\end{minipage}
\vfill
\clearpage
\setcounter{savedphotocount}{0}
}
```

Other page layout macros I needed to define for the album were:

```
\overunder  %two images centered horizontally, with equal top, between, and
            %below spacing (top right example in Figure~1)
\sidebyside %two images side by side with the pair centered horizontally,
            %bottom justified, with equal top and bottom spacing (top left
            %example)
\oneovertwo %three images with the top one over the bottom pair, with equal
            %top, between, and bottom spacing among the two rows of images,
            %and the image and image-pair centered horizontally (bottom left
            %example)
\twooverone %reverse of the above
\twoovertwo %a side-by-side pair over another side-by-side pair with equal top,
            %between, and bottom spacing among the rows, and the rows centered
            %horizontally (bottom right example)
```

This may have not been the optimal approach in terms of writing a bunch of different page layout macros, but it was very useful in terms of flexibly moving images around within the LaTeX file and experimenting with image ordering and page layouts until a satisfactory overall album layout was determined. If I was going to do lots of such albums, I might have wanted to include a lot of calculation in the macros and let LaTeX figure out how to lay out pages, but I didn't need that in this case (but I do have a good starting point if I ever want to create something more automatic).

3.4 Sort of an aside

Through all of the above, I had maintain some specials versions of the macro for the 3.5x5 inch images with the tall orientation that `\includegraphics` was not displaying at the correct size in the PDF. This mostly resulting in these images being bigger than real size and thus with less pixels per inch than the desirable 300 pixel minimum. Thus, I embarked on trying to figure out why `\includegraphics` wasn't correctly reading (or processing) the image size metadata from the image files. I could see no reason why not, and I asked the `tex.stackexchange.com` list if someone knew how `includegraphics` read and processed JPG image metadata. The list tried to help but had no definitive and workable answers. Eventually, I converted the problem JPGs to be NPGs, and then `\includegraphics` correctly sized the images. I don't know why this worked for NPGs and not for JPGs (the problem

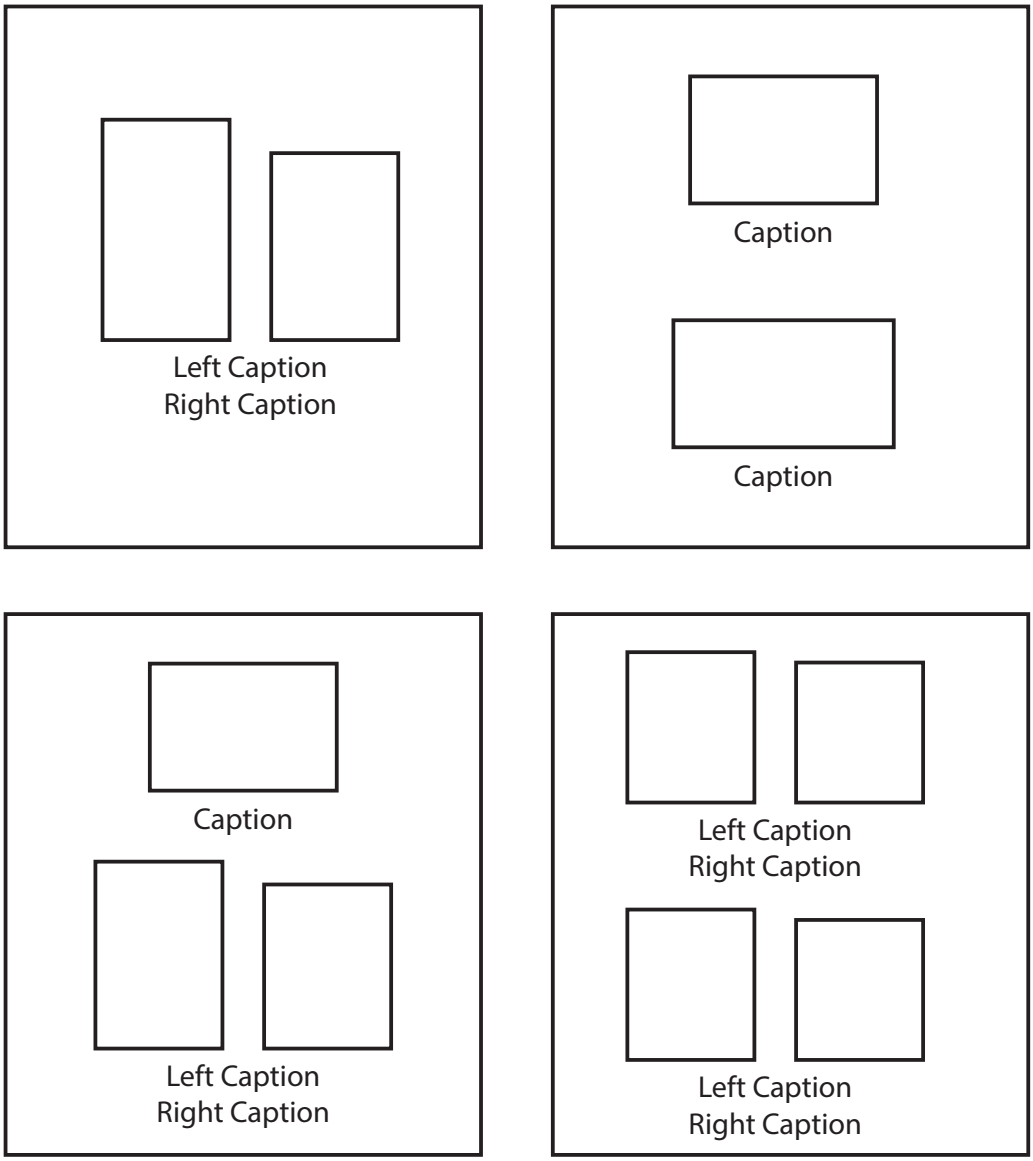

Figure 1: Some examples of page layouts; example images from the actual album are available at `walden-family.com/texland/photo-album/`

JPGs may have been originally scanned at 600 pixels per inch rather than 300 pixels per inch as was done for most of the images); however, now I have a work-around that I will try immediately if I see this problem again sometime.

3.5 Finishing the album

With the work-around found for the problem noted in the prior subsection, I could now do a final compilation of the PDF for delivery to the print shop. There, although almost all the images were grayscale originally (with only a few color images), I had the print shop print them all in color which made the old brownish grayscale images look better than they would have using a black-and-white based grayscale.

I made the cover (front, spine, and back) of the photo album with Adobe Illustrator rather than LaTeX. It might have been easier to do the whole layout with a graphical-user-interface (GUI) document layout tool. Then again it might have been more work with a GUI to match the caption widths to the images and to drag whole photo-caption pairs around (rather than dragging calls to `\savephoto` around in my text editor. Who knows? I have LaTeX, and I don't have the alternative tool.

4 Reflection

In this paper I have discussed three examples of little programming problems that I was led to by my work, and by the fact I was using TeX and had been a programmer by trade. TeX suits me particularly well as I dislike learning the ins-and-outs of user interfaces (especially user interfaces that change with product updates); and I often want something a little different than is built into the user interface of a less programmable tool (or built into the part of the tool I have not bothered to learn about). With TeX/LaTeX I can get a surprising amount done with the relatively modest amount of TeX/LaTeX programming I know (and the packages that "just work" without much study), and over time I have built quite a library of ad hoc TeX/LaTeX tools. The library is not very organized. I am careful to keep the sources for all my LaTeX-based documents, and when I need a tool I try to remember for which document I developed that tool, and then I go there and copy it.

A final note: another place I learn about programming TeX/LaTeX, in addition to fumbling around on my own, is looking at how Karl Berry changes my coding in things I submit for publication by TUG. He doesn't systematically go through a source file of mind improving it all; rather, he works with what I have done and makes it a little better as necessary, for example, for *TUGboat*. For instance, in the paper prior to this that I submitted to TUG for publication, I wrote the following line of code:

```
\centerline{\includegraphics[width=6in]{html.jpg}}
```

When I looked at the source file of the published version, Karl had changed that to

```
\centerline{\includegraphics[width=\hsize]{html.jpg}}
```

It's a trivial change, but I don't think I had ever used `\hsize` before (if I had, I had forgotten). I used the construction in that line again in this paper for inserting the image in Figure 1. Each paper Karl edits has several such "learning moments" that help me be a better TeX/LaTeX programmer without systematic study (which I will never do).