

First Book

Kurt Schmidt

Aug 2014

Contents

1	In a Galaxy, Far Far Away	1
1.1	Notes	1
1.2	Heather Graham	1
1.3	Compiling	1
1.3.1	Compiler Warnings and Errors	2
2	Basics	3
2.1	Hello, World	3
2.2	Modes	4
2.3	Metacharacters	5
2.3.1	Braces for Grouping	5
2.3.2	Spaces	5
2.3.3	Quotes	6
2.3.4	Dashes and Hyphens	6
2.4	Special Characters	6
2.4.1	Composition and Decoration	6
2.4.2	Other Defined Characters and Symbols	7
2.4.3	Common Symbols and Characters from Packages	7
2.4.4	Math Symbols	8
3	Math	11
4	Tables	13
4.1	Simple Tables	13
4.1.1	Vertical Lines	13
4.1.2	Content of Tables	13
4.2	Wrapping Text	14
4.3	Aligning on the Radix	14
4.3.1	@-Expressions	14

4.3.2	Tables of Numbers	14
4.4	Spanning	15
4.4.1	Cell Spanning Multiple Columns	15
4.4.2	Cell Spanning Multiple Rows	15
4.5	Don't know yet	16
5	Embedding Code in Text	17
5.1	Inlining Code in Text	17
5.1.1	Define Your Own Command	17
5.2	Blocks of code	17
5.2.1	<code>verbatim</code> Environment	17
5.2.2	The <code>listings</code> Package	18
5.2.3	Language Supported by <code>listings</code>	20

Chapter 1

In a Galaxy, Far Far Away

1.1 Notes

I'm just working my way through L^AT_EX. I put some Latex code here, and as I gain proficiency, perhaps I'll put more. For now, it would be helpful to look at the source, as you read the book. They should be in the same directory.

Note, if the margins look odd on alternate pages, this is a **book** document, meant to be bound.

For a text, this style of paragraph indentation/spacing is not so ideal. I'll play later, but, for now, try playing with `\setlength{\parindent}{width}` and `\setlength{\parskip}{width}`

Read up on *rubber lengths*. Also, be careful using **parskip**, as it affects spacing in lists and other places. The **parskip** package might be helpful here.

1.2 Heather Graham

Talented, lovely. And just gets more so.

1.3 Compiling

There should be a makefile in this directory, but I'm not at all happy with it.

Also, I'm having issues using CYGWIN_NT6.1 XXXXX 1.7.9(0.237/5/3) 2011-03-29 10:10 i686 Cygwin. Packages are missing, compilation is somehow more painful. But, it's a pretty old install, so. Ah, versions:

```
$ latex --version
pdfTeX 3.141592-1.21a-2.2 (Web2C 7.5.4)
kpathsea version 3.5.4
```

Oh, did ya catch some approximation of π in there?
On tux, we have:

```
$ latex --version
pdfTeX 3.1415926-1.40.10-2.2 (TeX Live 2009/Debian)
kpathsea version 5.0.0
```

Much newer version, anyway. I should maybe update this thing.

Anyway, two ways I've been compiling TEX to PDF. The first is two steps, TEX→DVI, then DVI→PDF:

```
$ latex book.tex
...
$ dvipdf book.dvi
...
```

The second way accomplishes the task in a single step:

```
$ pdflatex book.tex ...
```

Hmmmm. `pdflatex` isn't creating a table of contents for me on tux, either. Oh! Run it a couple times in succession. `bibtex` might be in there somewhere.

Also, `pdflatex` will allow you to use PDF-specific commands, and seems to be recommended.

Note, if using references, indices, table of contents, etc., pay attention to the output. A 2nd run might be suggested.

1.3.1 Compiler Warnings and Errors

You'll see various informational warnings:

```
LaTeX Font Warning: Font shape 'OMS/cmtt/m/n' undefined
...
```

You want to make sure that a correct substitution was made, but these are fairly harmless.

Errors, of course, need to be corrected, and will leave you at an interactive prompt (until I figure out how to signal batchmode). I find **x or q to be helpful.**

Chapter 2

Basics

Okay, I'm still working on my own understanding here, so, this is not etched in stone.

2.1 Hello, World

You knew it was coming. You expected it. You'd've been disappointed by its absence.

A basic \LaTeX document, article-style:

```
1 % hello.tex - simple example using the article
2 %
3 %
4 \documentclass[a4paper,12pt,titlepage]{article}
5 \pagestyle{plain}
6 \title{Hello, World}
7 \author{Kurt Schmidt \\\
8     Drexel, Computer Science}
9 \date{Sept. 2014}
10
11 \begin{document}
12 \maketitle
13
14 Here's your obligatory 'hello': 'Hello. Welcome to \LaTeX.' \TeX is the
15 basic language, Developed by Donald Knuth. \LaTeX is a way handy extension
16 to \TeX. So, basic syntax probably applies to both. Once we get into
17 packages, I've not clue, as yet, so, I'll just be talking about \LaTeX.
18
19 Easiest way to compile to PDF is using \texttt{pdflatex}.
20
```

```

21 \section{Math}
22 \label{hellomath}
23
24 And now, some math. Remember,  $\sum_{i=1}^m i = \frac{m(m+1)}{2}$ , along
25 with identities for sums, and the sum of a geometric series. You'll be
26 needing them.
27
28 Also recall these gems, you'll be needing them, too:
29
30 \[ b^{\log_b\{x\}} = x \]
31 \[ \log_b\{b^x\} = x \]
32 \[ \log_b\{xy\} = \log_b\{x\} + \log_b\{y\} \]
33 \[ \log_b\{x/y\} = \log_b\{x\} - \log_b\{y\} \]
34 \[ \log_b\{x^n\} = n\log_b\{x\} \]
35
36 So,
37
38 \[ x^{\log_b\{y\}} = y^{\log_b\{x\}} \]
39
40 \section*{El Fin du Monde}
41 \label{end}
42
43 And that's it for now. See section \ref{hellomath}.
44
45 \end{document}

```

Until I get better, you're gonna want to grab the file yourself and compile using `pdflatex`, see what the output looks like.

2.2 Modes

`TeX` supports 2 basic modes, math and text.

When you're typing along, you're entering text. Mostly you can just type as you would, with a few exceptions. There are metacharacters, paragraphs are separated by two (or more) newlines. We'll also want to talk about quotes, hyphens and spaces.

Lines 24 and 30-34, in our hello example, show some uses of math mode. See Section 3 for a bit longer discussion.

2.3 Metacharacters

The following characters can not simply be typed, as they have special meaning to \LaTeX :

`{ } $ % _ & # ^ \ ~`

Here is how to print these characters in \LaTeX . These metacharacters will be explained, but for the moment, trust me.

Symbol	\TeX sequence
<code>{</code>	<code>\{</code>
<code>}</code>	<code>\}</code>
<code>\$</code>	<code>\\$</code>
<code>%</code>	<code>\%</code>
<code>&</code>	<code>\&</code>
<code>#</code>	<code>\#</code>
<code>_</code>	<code>_</code>
<code>\</code>	<code>\textbackslash</code>
<code>'</code>	<code>\'{}{}</code>
<code>`</code>	<code>\`{}{}</code>
<code>^</code>	<code>\^{}{}</code>
<code>~</code>	<code>\~{}{}</code>
<code>~</code>	<code>\textasciitilde</code>
<code>~</code>	<code>\sim\$</code>

2.3.1 Braces for Grouping

`{ }` are used for grouping. Empty braces can be used to protect a special symbol from immediate following text. E.g., `\textbackslash{}foo` would display `\foo` , since `\textbackslashfoo` isn't a character.

2.3.2 Spaces

Whitespace serves to separate words, etc, but in a typeset environment, sequences of spaces, e.g., aren't of fixed size.

The `~` in \LaTeX is a *non-breaking space*.

If you want visible spaces, use `\textvisiblespace{}`.

2.3.3 Quotes

\LaTeX uses `` for left double quote, and '' for right double quote. Similarly, ` and ´ are used for left and right single quotes.

George said, ‘I heard an oldtimer once exclaim “Oh, batshit!”’

2.3.4 Dashes and Hyphens

I believe the solitary - is okay, though it might signal a wordsplit, so, keep it in mind, if you have problems. Well, let’s see: jack-in-the-box .

Two hyphens, -- , yields an endash, –, and three, --- , will give you an emdash, —.

So, if you want 2 or more literal adjacent hyphens, --, to keep the sequence from being interpreted, use an empty string to separate them: -{}-{}- yields --- .

2.4 Special Characters

Even if I knew what I was doing, this is a messy area. For sanity’s sake, you want to restrict yourself to the 7-bit ASCII characters, and use \LaTeX to create other characters.

You can change input encodings, latin1, utf8, etc., and we’re already past what I know.

We can represent many characters using escape codes. Some special characters are recognised in text mode, others only in math mode.

2.4.1 Composition and Decoration

We can add various accents and other decorations to any character. Here are a few:

Symbol	\TeX sequence
$\backslash`{a}$	à
$\backslash´{e}$	é
$\backslash¨{u}$	ü
$\backslash. {o}$	ó
$\backslash\sim{n}$	ñ

Use `\i` and `\j` for the dotless versions, so, `\^{i}` to get \hat{i} .

Note, if you want a literal backtick, rather than a left single quote, you'd be tempted to use `\``, but that is an escape sequence for the grave accent, expecting a character to follow, so, give it an empty string to act upon: `\`{}`

2.4.2 Other Defined Characters and Symbols

Here's a quick list of some:

<code>\l</code>	\textl
<code>\o</code>	\texto
<code>\TeX</code>	\TeX
<code>\LaTeX</code>	\LaTeX
<code>\textless</code>	\textless
<code>\textgreater</code>	\textgreater
<code>\S</code>	\S
<code>\P</code>	\P
<code>\dag</code>	\dag
<code>\ddag</code>	\ddag
<code>\copyright</code>	\copyright

2.4.3 Common Symbols and Characters from Packages

Okay, if it's a character, then it's available somewhere, often in several flavors. E.g., the official Euro sign, compared to one that'll display nicely w/the currently selected font (bold, italic, etc.).

```
\usepackage[gen]{eurosym}
```

will let you use `\euro{}` .

```
\usepackage{textcomp}
```

will give you `\textdegree` angle\$ (in math mode, confusingly enough).

Or, for temperatures, you might instead

```
\usepackage{gensymb}
```

, and use `\degreeC` , or `\celsius`

2.4.4 Math Symbols

Okay, in math mode you have access to many other symbols. You'll see examples of writing equations and such in Chapter 3.

A few common symbols:

Relational Operators	
$=$	<code>=</code>
\neq	<code>\neq</code>
\equiv	<code>\equiv</code>
\approx	<code>\approx</code>
\sim	<code>\sim</code>
\propto	<code>\propto</code>
\succ	<code>\succ</code>
\gg	<code>\gg</code>
\leq	<code>\leq</code>
\succcurlyeq	<code>\succcurlyeq</code>
\preceq	<code>\preceq</code>
\subseteq	<code>\subseteq</code>
\supseteq	<code>\supseteq</code>
\in	<code>\in</code>
\ni	<code>\ni</code>
\cap	<code>\cap</code>
\cup	<code>\cup</code>
\vee	<code>\vee</code>
\wedge	<code>\wedge</code>
\parallel	<code>\parallel</code>
\perp	<code>\perp</code>

Binary Operators	
\times	<code>\times</code>
\div	<code>\div</code>
\setminus	<code>\setminus</code>
\cap	<code>\cap</code>
\cup	<code>\cup</code>
\wedge	<code>\wedge</code>
\vee	<code>\vee</code>
\oplus	<code>\oplus</code>
\otimes	<code>\otimes</code>

Other Set	
\emptyset	<code>\emptyset</code>
∞	<code>\infty</code>

Logic	
\exists	<code>\exists</code>
$\exists!$	<code>\exists!</code>
\forall	<code>\forall</code>
\neg	<code>\neg</code>
\vee	<code>\vee</code>
\wedge	<code>\wedge</code>
\leftarrow	<code>\leftarrow</code>
\iff	<code>\iff</code>

Greek Letters	
Σ	<code>\Sigma</code>
σ	<code>\sigma</code>

Other Arrows	
\rightarrow	<code>\rightarrow</code>
\Leftarrow	<code>\Leftarrow</code>
\longrightarrow	<code>\longrightarrow</code>
\uparrow	<code>\uparrow</code>
\Downarrow	<code>\Downarrow</code>
\updownarrow	<code>\updownarrow</code>

Delimiters	
\lceil	<code>\lceil</code>
\langle	<code>\langle</code>
\rceil	<code>\rceil</code>
\lfloor	<code>\lfloor</code>

Other Fun	
\dots	<code>\ldots</code>
\cdots	<code>\cdots</code>
$\sqrt{}$	<code>\sqrt{}</code>
\flat	<code>\flat</code>
\natural	<code>\natural</code>
\sharp	<code>\sharp</code>
\spadesuit	<code>\spadesuit</code>
\heartsuit	<code>\heartsuit</code>
\diamondsuit	<code>\diamondsuit</code>
\clubsuit	<code>\clubsuit</code>

There are others, and packages containing still more.

Functions and other mathematical constructs will be covered in Chapter 3—Math Equations.

Chapter 3

Math

Placeholder. I've not gotten here yet.

Chapter 4

Tables

I took this mostly from <http://www.andy-roberts.net/writing/latex/tables>.

4.1 Simple Tables

Introduce tables with the *tabular* environment, describing columns.

```
\begin{tabular}{l c r }  
l    left justified  
c    centered  
r    right justified
```

4.1.1 Vertical Lines

Use the `|` or `||` in the declaration for vertical separators.

```
l || left justified |  
c || centered      |  
r || right justified|
```

4.1.2 Content of Tables

Cells in a record are separated by `&`. `\\` to start a new record.

Finally, use `\hline` to put horizontal lines in:

l	left justified
c	centered
r	right justified

4.2 Wrapping Text

Sadly, \LaTeX doesn't wrap long lines in tables, by default. It does provide 3 other column specifiers which take a width argument, in pt, in, cm, mm, or em:

Character	Meaning
<code>p{width}</code>	Paragraph column with text aligned vertically at the top
<code>m{width}</code>	Paragraph column with text vertically aligned in the middle (requires array package)
<code>b{width}</code>	Paragraph column with text vertically aligned at the bottom (requires array package)

The declaration for this table looks like:

```
\begin{tabular}{| 1 | p{8cm} | }
```

So, you'll need to play around a little, get it to come out nice. Or, there are packages already developed which'll do much of the work for you.

4.3 Aligning on the Radix

Again, there are packages that'll do this for you. But, we're here, so...

4.3.1 @-Expressions

Generally, the @ specifier takes a text argument, and a width specifier. When appended to a column, it suppresses normal cell spacing, and inserts the text before each cell's contents.

We're going to use it without any space, to just use the radix to join two columns of numbers (the integer and fractional parts).

4.3.2 Tables of Numbers

123.456	
3.14159265358979	
98765432.1	

4.4 Spanning

4.4.1 Cell Spanning Multiple Columns

In the data, place:

```
\multicolumn{numcols}{alignment}{contents}
```

, where *numcols* is the number of subsequent columns (the width of this cell, in columns), *alignment* is one of l, c, or r (maybe p?), with vertical separators, and, finally, the contents of the cell.

Endless Summer	
LOA	43'5"
LWL	36'4"
Beam	12'10"
Draught	5'11"
Displacement	19,620 lbs
Fuel (diesel)	63 gal
H ₂ O	153 gal

4.4.2 Cell Spanning Multiple Rows

A cell spanning multiple rows is introduced with:

```
\multirow{numrows}{width}{contents}
```

To use this, you'll need the `multirow` package:

```
\usepackage{multirow}
```

, where *width* could be a fixed width, or just use `\textwidth` for the natural width.

Endless Summer		
Specs	LOA	43'5"
	LWL	36'4"
	Beam	12'10"
	Draught	5'11"
	Displacement	19,620 lbs
	Fuel (diesel)	63 gal
	H ₂ O	153 gal
Accomodations	Cabins	3
	Double Berths	4
	Single Berths	0
	Heads	2
	Showers	3
	Masts	1
Rigging	Furling main	
	Furling genoa	

Huh. That `\hline` went all the way through. I dunno. Either choose a package to help you make tables, or try embedding a table in the table.

4.5 Don't know yet

Whistling.

Chapter 5

Embedding Code in Text

5.1 Inlining Code in Text

Use the `\texttt{code}` to inline a bit of code in a paragraph.

You'll need to change text colors yourself.

5.1.1 Define Your Own Command

You can define your own:

```
\newcommand{\code}[1]{\texttt{#1}}
```

So you could then use the new command in-line `\code{code}`

Honestly, not sure about this yet. We'll get there.

5.2 Blocks of code

5.2.1 verbatim Environment

You can place a code block in a verbatim environment. It seems that all (most? many?) metacharacter behaviors are inhibited, and line breaks are preserved. I'm not so sure about leading white space.

```
\begin{verbatim} ... \end{verbatim}
```

Let's see how this looks:

```
#include <stdio.h>
```

```

char *name = "Kurt" ;

int main( int argc, char *argv[] )
{
    printf( "Hello, %s!\n", name ) ;

    return( 0 ) ;
} /* main */

```

Hmmm. Okay. Moving on...

5.2.2 The listings Package

The listings package is part of L^AT_EX's standard library, I believe. It knows many languages (I hope), converts tabs to spaces you specify, and, with the color package, will highlight literals, keywords, comments, etc. I do not yet know if you can define your own languages.

Here is an example, found on StackOverflow (see comment):

```

\usepackage{listings}
\usepackage{color}

\definecolor{dkgreen}{rgb}{0,0.6,0}
\definecolor{gray}{rgb}{0.5,0.5,0.5}
\definecolor{dkred}{rgb}{1, 0.6, 0.6}

\lstset{frame=tb,
    language=C,
    aboveskip=3mm,
    belowskip=3mm,
    showstringspaces=false,
    columns=flexible,
    basicstyle={\small\ttfamily},
    numbers=none,
    numberstyle=\tiny\color{blue},
    keywordstyle=\color{dkgreen},
    commentstyle=\color{gray},
    stringstyle=\color{dkred},
    breaklines=true,

```

```

        breakatwhitespace=true,
        tabsize=3
    }

```

You can, of course, define and use whichever colors you like. Change default language in the middle of document with `\lstset{language=lang}` .

Then place your code in a `lstlisting` environment. Let's see if we can make our previous example better:

```

#include <stdio.h>

char *name = "Kurt" ;

int main( int argc, char *argv[] )
{
    printf( "Hello, %s!\n", name ) ;

    return( 0 ) ;
} /* main */

```

You can even pull code right from a file, using

```

\lstinputlisting[language=Python]{hello.py}

```

```

#!/usr/bin/env python

import sys

def greet( name="Neighbor" ) :
    print '\nHello,', name + ". How ya doin'?\n"

def main( args=sys.argv ) :
    if len( args ) >= 2 :
        greet( args[1] )
    else :
        greet()

    return 0

if __name__ == '__main__' :
    sys.exit( main() )

```

Nice. Let's see if this is a better way to embed LaTeX. There's a TeX language listed.

You can even pull code right from a file, using `\texttt{\lstinputlisting}`

```
\lstinputlisting[language=Python]{hello.py}
```

So, I suppose this is for indented math expressions (note the super and subscripts in math mode)

```
\[ |f(y) - f(x)| < \epsilon \]
\[ p_{i} = p_{i-1}^2 \]
```

Not exciting, but it works.

5.2.3 Language Supported by listings

This is according to http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings#Supported_languages . I don't see a date, know how current it is.

ABAP	Clean	HTML	MetaPost
ACSL	Cobol	IDL	Miranda
Ada	Comal	inform	Mizar
Algol	csh	Java	ML
Ant	Delphi	JVMIS	Modelica
	Eiffel	ksh	Modula-2
Assembler	Elan	Lisp	MuPAD
Awk	erlang	Logo	
bash	Euphoria	make	NASTRAN
Basic	Fortran		Oberon-2
C	GCL	Mathematica	OCL
C++	Gnuplot	Matlab	Octave
Caml	Haskell	Mercury	Oz

Pascal	Python	Scilab	Verilog
Perl	R	sh	
PHP	Reduce	SHELXL	VHDL
Plasm	Rexx	Simula	
PL/I	RSL	SQL	VRML
POV	Ruby	tcl	XML
Prolog	S	TeX	
Promela	SAS	VBScript	XSLT

Take a peek at the link, above, there are notes I didn't bother with. Various dialects of some of the languages are understood.

I was feelin' pretty good about the number of languages I'm familiar with, 'til I saw this list. I've not even heard of some of these.

