Summary

This document is meant to give you a quick introduction to TEX and LATEX. It covers a lot of material, but still barely manages to scratch the surface. It should provide you with some inspiration and, I hope, with some useful code you can copy, modify, and use in your report.

You should use the `blank-template.tex` file as a basis for your report rather than this file. Be sure to change its name to something sensible (maybe your team control number), and to set the values of the `\title`, `\question`, and `\team` commands to appropriate values.

Good luck!

– Claire

LaTeX Hints for ICM/MCM Contest Reports

ICM/MCM Contest Question Report Sample

Team # 12345

March 8, 2016

Contents

List of Figures

List of Tables

1 Introduction: What Is LaTeX?

LaTeX is a tool that allows you to concentrate on your writing while taking advantage of the TeX typesetting system to produce high-quality typeset documents.

LaTeX's benefits include

1. Standardized document classes
2. Structural frameworks for organizing documents
3. Automatic numbering and cross-referencing of structural elements
4. "Floating" figures and tables
5. High-level programming interface for accessing TeX's typesetting capabilities
6. Access to LaTeX extensions through loading "packages"

2 Structured Writing

Like HTML,[1] LaTeX is a markup language rather than a WYSIWYG[2] system. You write plain text files that use special *commands* and *environments* that govern the appearance and function of parts of your text in your final typeset document.

[1] HyperText Markup Language
[2] What You See Is What You Get.

2.1 Document Classes

The general appearance of your document is determined by your choice of *document class*. Document classes also load LaTeX packages to provide additional functionality.

LaTeX provides a number of basic classes, including article, letter, report, and book. There are also a large number of other document classes available, including amsart and amsbook, created by the American Mathematical Society and providing some additional mathematically useful structures and commands; foils, prosper, and seminar, which allow you to create "slides" for presentations; the math department's thesis class, for formatting senior theses; and many journal- or company-specific classes that format your document to match the "house style" of a particular periodical or publisher.

2.2 Packages

LaTeX packages, or *style files*, define additional commands and environments, or change the way that previously defined commands and environments work. By loading packages, you can change the fonts used in your document, write your document in a non-English language with a non-ASCII font encoding, include graphics, format program listings, add custom headers and footers to your document, and much more.

A typical TeX installation includes hundreds of style files, and hundreds more are available from the Comprehensive TeX Archive Network (CTAN), at `http://www.ctan.org/`.

2.3 Structural Commands

LaTeX provides a set of structural commands for defining sections of your document, as shown in Table 1.

Note that the argument to structural commands are moving arguments (see Section 4.1.1) because they can be reused in the table of contents or in page headers or footers. Structural commands can take an optional argument in which you specify nonfragile commands or a shorter version of the actual section title that fits. You'll generally know when you need to provide an optional argument by TeX's behavior.

Command	Notes
`\part`	book & report only
`\chapter`	book & report only
`\section`	
`\subsection`	
`\subsubsection`	
`\paragraph`	
`\subparagraph`	

Table 1: Structural commands in LaTeX.

2.4 Labels and References

Sections are numbered automatically by LaTeX during typesetting. If you change your mind and decide that a subsection should be promoted to a section, or moved to the end of your document, the sections will be renumbered so that the numbers are consistent.

Sections can also be `\labeled` with a tag such as

```
\section{Our Complicated Equations}%
\label{sec:complicated-eqs}
```

and referred to with a `\ref` or `\pageref` command, as in

```
In Section~\ref{sec:complicated-eqs}, we pointed out...
```

or

```
On page~\pageref{fig:gordian-knot}, we illustrated...
```

LaTeX substitutes the correct section number when typesetting your document.

The same commands can be used with numbered environments such as `equation`, `theorem`, and so forth.

Use *meaningful* labels—labeling a section as `sec12` may seem useful, but it will be confusing if you end up moving it to a different place in the document and its number changes to Section 34. It's also easier to remember what reference you want if you use a meaningful name.

You may also want to impose some additional organization through the use of *namespaces*, as I've done in this document. Rather than give different types of objects undistinguished labels, I precede section labels with `sec:`, equations with `eq:`, figures with `fig:`, tables with `tab:`, and so on.

Emacs with AuxTEX and RefTEX gives you easy access to these labels, as do many other editors with TEX-specific features. It's much easier to find the particular label you're looking for if you have some additional information to help you. Adding the prefixes also reminds you of what text should precede the `\ref` command.

2.5 Commands

LATEX uses commands for changes that are very limited in scope (a few words) or are unlimited in scope (the rest of a document). For example, the commands

```
\textbf{bold}
\emph{italic (emphasized)}
\textsf{sans serif}
```

produce the following output in a typeset document:

bold *italic (emphasized)* sans serif

These are "commands with arguments"—the command itself starts with a backslash (\), and its *argument* appears inside braces `{ }`). Some commands may also have *optional arguments*, which are typed inside brackets (`[ ]`).

There are also commands that take no arguments, such as `\noindent`, `\raggedright`, and `\pagebreak`.

You can define your own commands, as discussed in Section 2.8.

2.6 Environments

LATEX provides a number of *environments* that affect the appearance of text, and are generally used for more structurally significant purposes. For example, the commands listed above are typeset inside a `verbatim` environment typed inside a `quote` environment. Their results were typeset inside a `quote` environment.

Environments use special commands to start and close—`\begin` and `\end`, followed by the name of the environment in braces, as in

```
\begin{quote}
  ``This is disgusting---I can't eat this.  That arugala is so
  bitter\ldots{} It's like my algebra teacher on bread.''
  \flushright -- Julia Roberts in \emph{Full Frontal}
\end{quote}
```

producing

> "This is disgusting—I can't eat this. That arugala is so bitter…
> It's like my algebra teacher on bread."
>
> – Julia Roberts in *Full Frontal*

Some environments may take additional arguments in braces (required) or brackets (optional).

Note that the order in which environments nest is extremely important. If you type an environment inside another environment, the inner environment must be `\ended` *before* the second environment is closed. It's also vitally important that you have an `\end` line for each `\begin` line, or LaTeX will complain.

2.6.1 The `document` Environment and the Preamble

The most important environment is the `document` environment, which encloses the *body* of your document. The code before the `\begin{document}` line is called the *preamble*, and includes the all-powerful `\documentclass` command, which loads a particular document class (see Section 2.1); optional `\usepackage` commands, which load in additional LaTeX packages (see Section 2.2); and other setup commands, such as user-defined commands and environments, counter settings, and so forth.

I generally also include the commands defining the title, author, and date in my preambles, but other people include them just after `\begin{document}`, before the `\maketitle` command, which creates the title block of your document.

2.6.2 Math Environments

One of the major hallmarks of TeX is its ability to typeset mathematical equations.

The two primary ways of doing so are with the use of *inline* and *display math environments*. These environments are used so often that there are shorthands provided for typing them. Inline math environments, such as $a^2 + b^2 = c^2$, can be typed as

```
\begin{math}
a^{2} + b^{2} = c^{2}
\end{math}
```

or

```
$a^{2} + b^{2} = c^{2}$.
```

Display math environments set your equation apart from your running text. They're generally used for more complicated expressions, such as

$$f(x) = \int \left(\frac{x^2 + x^3}{1} \right) dx$$

which can be typed as

```
\begin{displaymath}
f(x) = \int \left( \frac{x^2 + x^3}{1} \right)dx
\end{displaymath}
```

or

```
\[
f(x) = \int \left( \frac{x^2 + x^3}{1} \right)dx
\]
```

Generally, you'll want to use the `$`
delimited form for inline math, and the `\[ \]` form for display math environments. [Besides being easy to type, these forms are *robust*, which means that they can be used in *moving arguments*, elements that TEX may need to typeset in more than one place (such as a table of contents) or adjust (such as footnotes).]

The `equation` Environment You'll probably want to use the `equation` environment for any formula you plan to refer to. LATEX not only typesets the contents of an `equation` environment in display mode, it also numbers it, as in

$$f(x) = \int \left(\frac{x^2 + x^3}{1} \right) dx \tag{1}$$

written as

```
\begin{equation}
\label{eq:myequation}
f(x) = \int \left( \frac{x^2 + x^3}{1} \right)dx
\end{equation}
```

Note that you can refer to this formula as Equation 1 with

`\ref{eq:myequation}`.

2.7 Fonts

Generally you'll want to let LaTeX handle the fonts for you—Knuth's Computer Modern fonts are used by default, and include a wide range of variations that can cover most any use you can think of.

If you want to get fancy (and portable; see Section 6.6), you can use Type 1 PostScript fonts, such as Times, Palatino, Utopia, and so forth. These font sets are accessible with packages with names like times, palatino, and utopia. There are others, as well—a command such as `\locate psnfss | grep sty` will find most of them.

You can also get fonts from CTAN (see Section 2.2), both bitmap and Type 1. There's even support for TrueType fonts in some TeX systems.

2.7.1 Font Commands

Most of your concern about fonts is probably related to what you're writing. You might want some *emphasized* or **bold** text to stress a point or highlight a key term. Filenames might be set in `typewriter text` (although you should consider using the url package to help you out—by default, text set in typewriter text isn't hyphenated, which can lead to some unattractive line breaks).

You can also set text in sans serif or SMALL CAPS. Table 2 shows you some of the most commonly used font commands provided by LaTeX.

Command	Result
`\emph`	*emphasized text*
`\textsf`	sans-serif text
`\texttt`	`typewriter text`
`\textbf`	**bold text**
`\textsc`	SMALL CAPS TEXT
`\textsl`	*slanted text*
`\textit`	*italic text*

Table 2: Commonly used font commands.

I recommend that you use `\emph` in preference to `\textit`, and use `\textbf` sparingly. `\emph` is a smarter command than `\textit`—it switches back to the roman font when necessary. For example, *She loved* Scooby Doo. versus *He loved Titanic.*

For complicated font changes, or for special font usages that you're typing a lot, creating a macro (Section 2.8) is the way to go. I often just write, tossing in custom commands as I go, and waiting to define them until just before I compile the document.

2.8 Customization

The main advantage of using commands and environments is that they allow you to organize your writing. A useful side-effect is that you can change your mind about the way an element is typeset, and change all the appearances of that element in document by editing one piece of code. For example, in this document the names of environments have been set in "typewriter text", using a command I created called `\env`, which is defined as

```
\newcommand{\env}[1]{\texttt{#1}\xspace}
```

All I have to do to make the names of all the environments in the document appear in sans-serif type instead is to change that one line to

```
\newcommand{\env}[1]{\textsf{#1}\xspace}
```

You can do the same with almost anything you can conceptualize—key terms, people's names (especially names of people from non-English-speaking countries), files, functions, and so on.

3 Mathematical Notation

As we saw in Section 2.6.2, math is typed into one of several kinds of math environments. Choose your environment based on the context and importance of the content. Any formula you plan to refer to should be typed in an `equation` environment (or a similar environment that supports labels).

You should punctuate your mathematics as if the formulae were normal parts of English sentences. Reading them aloud is often a useful method for ensuring that you have all the commas in the right places. Where appropriate, you should also follow a displayed formula at the end of a sentence with a period.

3.1 Sums and Products

It's easy to typeset sums and products. For example,

$$f(n) = \sqrt[n]{\sum_{k=1}^{n} \binom{n}{k} f(n-k)}, \quad \prod_{n=2}^{\infty} \frac{n^3-1}{n^3+1} = \frac{2}{3}. \tag{2}$$

3.2 Matrices

It's a little more difficult to create matrices, but not too bad:

$$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 & 2 \\ 3 & -2 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

3.3 Symbols

LaTeXprovides an enormous number of symbols. Additional packages (loaded with `\usepackage`) may provide additional symbols and fonts.

For example, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ require you to load the amsfonts package (which is automatically loaded by the `icmmcm` class). These symbols are generated by `\mathbb`, which only works in math mode.

Subscripts and superscripts are easy—`$a_n$` produces a_n, and `$x^2$` produces x^2. Ordinal numbers, such as 3^{rd}, n^{th}, and so forth,[3] can be produced with code like `$3^{\textrm{rd}}$`, `$n^{\textrm{th}}$`. Equation 4 shows a formula with a superscript.

$$\int_0^{\pi} \cos^{2n+1} x \, \mathrm{d}x = 0 \qquad \forall\, n \in \mathbb{N}. \tag{4}$$

Notice that `\cos` produces a nice roman "cos" within math mode. There are similar commands for common functions like `\log`, `\exp`, and so forth. More can be defined with the `\DeclareMathOperator` command provided by the amsmath package.

You can stack symbols over other symbols. In math formulas,

$$m\ddot{x} + \gamma\dot{x} + kx = 0, \tag{5}$$

or to produce diacritical accents, as in

[3]Some fonts may include their own ordinals that can be accessed with special commands.

$$\sqrt{\begin{bmatrix} \alpha & \beta & \gamma & \delta & \epsilon & \zeta \\ \eta & \theta & \iota & \kappa & \lambda & \mu \\ \nu & \xi & o & \rho & \pi & \sigma \\ \tau & \upsilon & \phi & \chi & \psi & \omega \\ \Gamma & \Delta & \Theta & \Lambda & \Xi & \Pi \\ \Sigma & \Upsilon & \Phi & \Psi & \Omega & \varphi \\ \cdots & \ldots & \vdots & \ddots & \vdots & \cdot \end{bmatrix}}. \tag{6}$$

Figure 1: Greek letters and some symbols.

> Paul Erdős s'est reveillé tôt pour enseigner le français à son frère et sa sœur.

LaTeX has lots of Greek letters and ellipses too, some of which are shown in Figure 1.

See **?**, pp. 455–474, or **?**, pp. 123–127, for lists of the symbols available. Intext, you might see some of these symbols used as

> The Strong Induction Principle asserts that if a statement holds for the integers 1, 2,..., n, and if whenever it holds for $n = 1$, ..., k then it also holds for $n = k + 1$, then the statement holds for the integers 1, 2, 3, ... Using this Principle, it can be shown that $1 + 2 + \cdots + n = n(n + 1)/2$ for all positive integers n.

Notice that in the lists of integers, the ellipsis was made using the `\ldots` command, and that the periods were nicely spaced between the commas. In the sum, the dots were made with `\cdots` and were centered on the line. The amsmath package provides a "smart" `\dots` command that can generally get things right based on the context.

So, with `\dots` alone, the previous examples come out as

> $1, 2, \ldots, n$
> $n = 1, \ldots, k$
> $1, 2, 3, \ldots$
> $1 + 2 + \cdots + n = n(n + 1)/2$

The general $n \times n$ matrix can be typeset as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}. \tag{7}$$

A fine point: lists of numbers that you're using in a mathematical sense (as opposed to dates, numbers of objects, etc.) should be typed in math mode. For example, 341, 541, 561, and 641. The same numbers without math mode are 341, 541, 561, and 641. Depending on the fonts and packages that you're using, you may notice a little bit more space around the first set than the second. With some packages, numbers intext may be set using old-style figures by default, as in 341, 541, 561, and 641.

3.4 More Math

In Fourier analysis, we talk about the z-domain.

If a is an even number, then

$$a + \phi(a) < \frac{3a}{2},$$

and

$$\sigma(a) > \frac{2^{\alpha+1} - 1}{2^{\alpha}}\, a \geq \frac{3a}{2},$$

where α is the greatest power of 2 that divides a, $\phi(a)$ is the number of integers less than a and relatively prime to a, and $\sigma(a)$ is the sum of the divisors of a (including 1 and a).

Typeset a piecewise function using the `cases` environment (from the `amsmath` package) as follows:

$$|x| = \begin{cases} x, & \text{if } x \geq 0; \\ -x, & \text{otherwise.} \end{cases}$$

In frosh physics, students come to know the true meaning of $\mathbf{F} = m\mathbf{a}$, $E = mc^2$, and $-\frac{\hbar^2}{2m}\nabla^2\psi + V\psi = i\hbar\frac{\partial\psi}{\partial t}$.

3.5 Aligning Equations

In days gone by, people used the `eqnarray` environment to align equations. `eqnarray` has generally been replaced by `align` and some variants such as `flalign`, which places the leftmost column as far left as possible and the rightmost column as far right as possible; `alignat`, which allows you to specify the spacing; and more. See **?**, **?**, **?**, or **?** for more information about the alternatives.

In Equations 8–11, the = signs have been aligned using the `eqnarray` environment.

$$\begin{aligned} 3x^4 + \frac{1}{x^4} &= x^4 + 4x^2 + 6 + \frac{4}{x^2} + \frac{1}{x^4} - 4x^2 - 6 - \frac{4}{x^2} && (8)\\ &= \left(x + \frac{1}{x}\right)^4 - 4x^2 - 6 - \frac{4}{x^4} && (9)\\ &= \left(x + \frac{1}{x}\right)^4 - 4x^4 - 8 - \frac{4}{x^4} + 8 - 6 && (10)\\ &= \left(x + \frac{1}{x}\right)^4 - 4\left(x + \frac{1}{x}\right)^2 - 6. && (11) \end{aligned}$$

Equations 12–15 show the same set of equations aligned with the `align` environment.

$$\begin{aligned} 3x^4 + \frac{1}{x^4} &= x^4 + 4x^2 + 6 + \frac{4}{x^2} + \frac{1}{x^4} - 4x^2 - 6 - \frac{4}{x^2} && (12)\\ &= \left(x + \frac{1}{x}\right)^4 - 4x^2 - 6 - \frac{4}{x^4} && (13)\\ &= \left(x + \frac{1}{x}\right)^4 - 4x^4 - 8 - \frac{4}{x^4} + 8 - 6 && (14)\\ &= \left(x + \frac{1}{x}\right)^4 - 4\left(x + \frac{1}{x}\right)^2 - 6. && (15) \end{aligned}$$

3.6 Adjusting Spacing

Sometimes you need to adjust the spacing and fonts inside integrals. Typically, the "d" (as in dx) is set in Roman type. Rather than

$$\int\int \frac{1}{1-xy} dxdy. \qquad (16)$$

you want

$$\iint \frac{1}{1-xy}\,\mathrm{d}x\,\mathrm{d}y. \tag{17}$$

The integral signs have been moved together using the "negative space" command `\!`. Extra space has been added between the elements of integration, dx and dy, and between those elements and the integrand with the "thin space" command, `\,`.

Table 3 shows the different spacing commands available in math mode. There are additional spacing commands provided by the amsmath package, not shown here.

	LaTeX Command	
Name	Short	Long
Positive Space		
thinspace	`\,`	`\thinspace`
medspace	`\:`	
thickspace	`\;`	
1 em		`\quad`
2 em		`\qquad`
Negative Space		
thinspace	`\!`	`\negthinspace`

Table 3: LaTeX math spacing commands.

3.7 Specifying Equation Numbers or Names

All the equations you've seen so far are numbered consecutively. You can specify a number (or name) for a single equation by placing the formula in a display math environment (but not an `equation` environment) and giving the desired number or name as the argument to an `\eqno` command. For example,

$$\int_0^\infty e^{-x^2}\,\mathrm{d}x = \frac{\sqrt{\pi}}{2}. \qquad (42),$$

or

$$\int_0^\infty e^{-x^2}\,\mathrm{d}x = \frac{\sqrt{\pi}}{2}. \qquad \text{(cool formula)},$$

Note that you have to specify parentheses in the argument to the `\eqno` command. If you name a formula, you also have to enclose the text within

a command such as `\mathrm`, or it will be set as if it was a string of variables (and without any spaces).

If you'd like to have many aligned equations without numbers, use the starred form of the `align` environment, `align*`, as in

The area K of $\triangle ABC$ is given by

$$\begin{aligned} K &= \frac{ah_a}{2} \\ &= \frac{ab}{2}\sin C \\ &= rs \\ &= \sqrt{s(s-a)(s-b)(s-c)} \\ &= \frac{abc}{4R} \\ &= \frac{a^2 \sin B \sin C}{2\sin A} \\ &= 2R^2 \sin A \sin B \sin C, \end{aligned}$$

where A, B, and C are the angles in $\triangle ABC$, r is the radius of the inscribed circle, R is the radius of the circumscribed circle, s is one-half of the perimeter, and h_a is the length of the altitude from the vertex A to the side BC.

3.8 Sizing Delimiters

The `\left` and `\right` commands, followed by a bracket, brace, or parenthesis, tell TEX to adjust the size of the delimiter to its contents.

$$f(x) = \left(1 + (1+x)^2\right)^n. \tag{18}$$

You can also use commands such as `\big`, `\Big`, `\bigg`, or `\Bigg` to specify larger delimiters (useful if you have multiple levels of delimiters), as in

$$\big(\quad \Big(\quad \bigg(\quad \Big($$

or

$$\Big(\Big(\Big(\big((a+b)+c\big)+d\Big)+e\Big)+f\Big)$$

3.9 Theorems

The `theorem` environment is an easy way to typeset theorems in the text. To use it, type a `\newtheorem` command in the preamble of your document like

```
\newtheorem{Theo1}{Theorem}
```

You can then type a theorem using your theorem environment.
This document includes three such definitions,

```
\newtheorem{Theo1}{Theorem}
\newtheorem{Theo2}{Theorem}[section]
\newtheorem{Lemma}[Theo2]{Lemma}
```

which show you some of the possibilities available. Examples of each appear below.

3.9.1 A `Theo1` Environment

Theorem 1 *The equation $x^4 + y^4 = z^4$ has no solutions where x, y, and z are positive integers.*

3.9.2 A `Theo2` Environment

Theorem 3.1 (Wilson) *A positive integer p is prime if and only if*

$$(p-1)! \equiv -1 \pmod{p}.$$

3.9.3 A `Lemma` Environment

Lemma 3.2 *Prof. Bernoff's Putnam mug is a multiply connected 2-manifold of genus 1.*

3.10 Proofs

Adding a proof is very simple. Two positive integers a and b are amicable if $\sigma(a) = \sigma(b) = a + b$, where $\sigma(N)$ denotes the sum of the divisors of N, as above. The following is a theorem with an associated proof.

Theorem 3.3 *There do not exist two consecutive integers which are amicable.*

Proof: Since even numbers are annoying, no integers are amicable with even numbers. Thus, if two consecutive integers are amicable, they are both odd. However, two consecutive odd numbers do not exist. □

To create the end-of-proof marker shown here, use `\hfill$\Box$`. The `\hfill` makes the box print flush right at the end of the line, as here.

You can also use the `proof` environment provided by the `amsthm` package.

4 Figures and Tables—LaTeX's Float Environments

LaTeX provides two "float" environments, `figure` and `table`. Float environments are so called because they can be typeset on a later page in your document than their location in the source code.

The `table` environment is generally used for—surprise!—tables. The `figure` environment is often used for graphs or diagrams, but could also be used for other illustrative graphics.

The basic float environments don't format their contents specially. If you want an illustration or table to appear centered, you will need to type it inside a `center` environment or add a `\centering` command.

4.1 Captions

By adding a `\caption` command, you can specify a caption that will appear with the float. Its position depends on where in the environment you type it—if the command is at the top of the environment, the caption will be typeset above its contents; if at the bottom, the caption will appear beneath its contents. Captions are usually set at the bottom of a float, but if a particular publisher or journal prefers the captions on top, you can accommodate them.

Captions should generally be written as brief, complete sentences, ending with a period. They should either be capitalized as normal sentences or use headline capitaliztion—capitalized as you would the title of a document or a section. So

> Production Statistics from Soviet Russia, 1977–1987.

or

> Production statistics from Soviet Russia, 1977–1987.

rather than

production statistics from Soviet Russia, 1977–1987

Whichever style you choose, be consistent!

Avoid explaining the whole float in the caption. Do your explanation in the text that refers to the float.

The `\caption` command takes an optional argument, which is typed inside brackets (`[ ]`). This argument is used in the list of tables or list of figures in place of your actual caption.

4.1.1 Fragile Commands and Moving Arguments

Both arguments to the `\caption` command are *moving arguments* (because TEX can move them). Some commands are *fragile*, that is, they produce output that can cause problems if the typeset text is moved somewhere other than the place that TEX originally thought it would be typeset.

To prevent fragile commands from being expanded too early and causing problems, you can use the `\protect` command just before the command you want to keep unexpanded.

4.1.2 Labels

The `\label` command for a float is generally typed immediately after the `\caption` command.

4.2 Figures

Graphic images are included in your document in a `figure` environment. The state-of-the-art method requires you to load the graphics or graphicx package. Both packages provide the same functionality, but take arguments in a slightly different format.[4] More information about the graphics and graphicx packages is available in its manual, `grfguide ?`, which is included in DVI, PostScript, or PDF format with most TEX systems and accessible by typing `texdoc grfguide` at a shell prompt.

Modern TEX systems use PDF as their default output format, and expects that your included graphics will be in PDF format (for vector images) or PNG or JPEG format (for bitmaps). Note that whenever possible you should try to save graphics in the vector PDF format, because they can be

[4]The graphicx package defines commands that take their arguments in key–value pairs, and is the one that most people use; the graphics package is often loaded by document classes to avoid forcing the key–value argument form on people who don't want to use them.

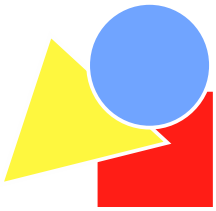

Figure 2: Some shapes.

rescaled to any size without losing detail and can be rendered clearly both on screen and on paper.

Encapsulated PostScript or *EPS* is an older format used by the old LaTeX to DVI to PostScript toolchain. EPS files are special PostScript files that define their "bounding box", may include a bitmap representation for use in previewers, and are restricted from using some PostScript operators.

EPS files are generally created with a vector graphics application such as Adobe Illustrator, Dia, OmniGraffle, or Visio. They can also be created from TeX files using the `-E` flag with `dvips` or from TeX code using an application such as LaTeXiT.

With the development of PDFTeX, which is now the default TeX engine, generating Portable Document Format files has become much easier. PDFTeX requires that your graphics are also PDF files (or PNGs, if they're bitmaps). See Section 6.5 for some hints.

Both formats use the same command, however: `\includegraphics`.

The following code produces the graphic in Figure 2:

```
\begin{figure}
  \begin{center}
    \scalebox{.50}{\includegraphics{shapes}}
  \end{center}
  \caption[Some shapes]{Some shapes.}%
  \label{fig:a-graphic}
\end{figure}
```

Notice that we didn't specify the extension in the filename argument to the `\includegraphics` command. By dropping the extension, we can typeset this document with PDFLaTeX or with the older toolchain (provided that we have graphics in the appropriate formats) by changing the commands that we run. The `\includegraphics` package searches for the graphic formats supported by the particular engine you're using.

4.2.1 LaTeX Diagrams

LaTeX can also be used to create both simple pictures and sophisticated diagrams. Figure 3 shows a graph created with the `picture` environment.

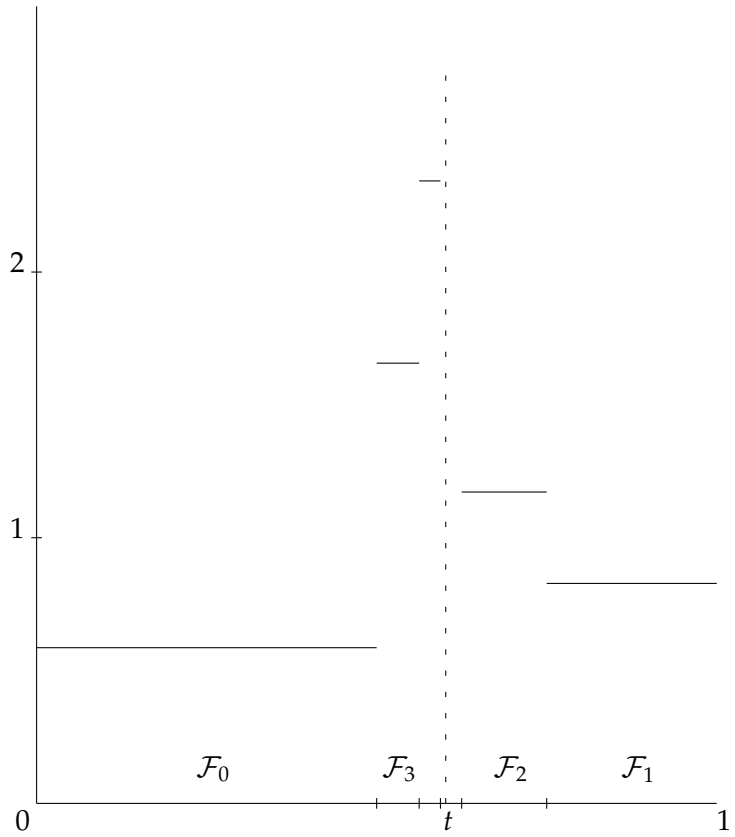

Figure 3: A step function with a peak at t.

Chapter 6 of **?** describes how you can create diagrams such as that shown in Figure 3.

4.3 Tables

Tables are a complicated subject, not because they're difficult to do in LaTeX, but because they're difficult to do *right*. Most books on LaTeX cover tables, but present what Simon Fear, author of the booktabs package, calls "tableaux". One such tableau is illustrated in Table 4.

gnats	gram	$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

Table 4: A tableau. (Taken from **?**, pg. 64.)

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Table 5: The tableau as a table.

Simon argues that such a tableau would be better presented as the table shown in Table 5. *The Chicago Manual of Style*, **?**, and Edward Tufte (**?**) support his assertion, and provide excellent references and inspiration.

The `booktabs` package, which is automatically loaded by the `icmmcm` class, has some special commands for creating lines of different thicknesses for use as top, bottom, and midrules. It also has some code that provides the `\cmidrule` command, for creating spanner rules for decked spanner heads. The rest is up to you and your style guide.

As an example of a table to strive towards, Table 6 was taken from an example table in *The Chicago Manual of Style*. All the tables (except for those in the section describing tables, alas) in Grätzer's *Math into LaTeX* were prepared with `booktabs`.

5 Typesetting

So you've got a LaTeX source document. How do you get a typeset document that you can print or put on the web?

Typesetting a document is referred to as "TeXing", "compiling", or "typesetting". Generally, you want to create a PostScript file (for printing) or a

	1900		1906		1910	
Party	% of Vote	Seats Won	% of Vote	Seats Won	% of Vote	Seats Won
	Provincial Assembly					
Conservative	35.6	47	26.0	37	30.9	52
Socialist	12.4	18	27.1	44	24.8	39
Christian Democrat	49.2	85	41.2	68	39.2	59
Other	2.8	0	5.7	1	5.1	0
Total	100.0	150	100.0	150	100.0	150
	National Assembly					
Conservative	32.6	4	23.8	3	28.3	3
Socialist	13.5	1	27.3	3	24.1	2
Christian Democrat	52.0	7	42.8	6	46.4	8
Other	1.8	0	6.1	0	1.2	0
Total	100.0	12	100.0	12	100.0	13

Table 6: Elections in Götefrith province, 1900–1910. (Taken from **?**, pg. 414.)

PDF file (for placing on the web). There are multiple ways to do both tasks.

5.1 Getting to Paper

Starting with a LaTeX document, `foo.tex`, you can create a PDF file by running the following commands:

```
unix% pdflatex foo
```

5.2 General Comments

LaTeX does its numbering (and some other functions) by writing information to an *auxiliary file*. It then reads that information in on the next pass, and uses it to typeset references. Thus you have to run `latex` or `pdflatex` at least twice whenever you make a change that affects the numbering of elements or the flow of text across pages. It's generally good practice to run LaTeX three times, or until it stops warning you about possible changes.

Character		Function	To Typeset
#	octothorp	Macro parameter character	`\#`
$	dollar sign	Start/end inline math mode	`\$`
%	percent sign	Comment character	`\%`
&	ampersand	Column separator	`\&`
_	underscore	Subscripts, as in x_2	`\_`
{, }	braces	Parameters	`\{, \}`
~	tilde	Nonbreaking space	`\~{}`
^	caret	Superscripts, as in x^2	`\^{}`
\	backslash	Starts commands	`\verb ||`

Table 7: Special characters in LaTeX.

5.3 Additional Programs

There are some additional functions, such as indexing and bibliographies, that use external programs to read auxiliary files and produce LaTeX code for inclusion on later runs. We won't cover those programs in this document.

6 Tips and Tricks

LaTeX is a very complicated and powerful language. As a result, there are many sneaky aspects to it that will cause you problems if you don't know about them. Here are a few.

6.1 Special Characters

TeX and LaTeX have a number of "special" characters that are reserved for use by the language. Using these characters in your writing requires you to do a bit of extra work, as shown in Table 7.

6.2 Comments and Spacing

You can add comments to your source file that won't appear in your typeset document by starting them with a %. Any line that starts with a % will be "commented out", and won't be interpreted. You can also add a % at the end of a line, with or without text, and it will make the end of the line disappear.

For example,

```
% This is a comment line.
This is not a comment line.

This line has a comment at the end%
% This line should be invisible.
of the line.
```

will typeset as

This is not a comment line.

This line has a comment at the endof the line.

Notice the lack of a space in "endof" on the last line of the typeset output. TEX expects a carriage-return character at the end of a line, and interprets that carriage return as an interword space. If you comment out the end of a line, you also comment out the carriage return on that line, and you'll have words run into one another unless you have a space before the %.

TEX collapses multiple spaces into one, and ignores whitespace at the beginning of a line. Thus

```
No spaces.

    Five spaces.

    A tab.
```

typesets as

No spaces.

Five spaces.

A tab.

(The lines are indented because they are at the start of a paragraph. You can suppress paragraph indentation with `\noindent`.)

Paragraphs are delimited by two carriage returns (with or without whitespace between them).

6.3 Quotes and Dashes

Because TEX was designed to do high-quality typesetting, it cares about which quotation mark and dash you're using, and requires you to specify the correct punctuation (although most text editors with special TEX modes will do the substitution for you).

Open and close double quotes—" and "—are created by typing ` ` and ' ', respectively. The double-quote mark, ", is typeset as " (and is useful for abbreviating "inches", as in 36").

Single-quotes, ' and ', are typed with ` and '.

There are three basic forms of dashes:

1. The hyphen, -, is typed as a single dash, -
2. The en dash, –, is typed as two dashes, --
3. The em dash, —, is typed as three dashes, ---

Hyphens are used in hyphenated words, as in "complex-typesetting mechanism". En dashes are used to indicate ranges, as in "there are 35–50 of them". Em dashes are used to separate independent phrases, as in "John believed—honestly believed—that he was right."

Note that you shouldn't type spaces around any of these dashes—they run directly against the words on either side, as in `35--50`.

6.4 Controlling Pagination

Sometimes you may need to override LATEX's choices for line or page breaks. your document into lines and/or pages. The `\pagebreak` command causes LATEX to start a new page immediately after the command appears. The `\\` command can be used to tell LATEX where to break a line.

In general, you should let LATEX have its way, especially if your document is going to be published by someone else, as they will undoubtedly have many changes that will have to be made before your document works for them. If you do need to tinker with your document's layout, you should avoid doing so until you're very nearly done. If you go back and add or remove text after forcing LATEX to do your will, you may find that new blank spaces appear as a result of your changes.

George Grätzer's *Math into LATEX* includes a chapter on preparing books that covers this topic in depth.

6.5 Updating EPS Graphics for Use With PDFTEX

PDFTEX supports PDF, PNG, and JPEG as native graphic file formats. EPS is not directly supported—to use EPS figures with PDFTEX, you must first convert your EPS files to PDF.

If you're using a graphics program such as Adobe Illustrator to prepare your figures, just save them as PDF instead of (or in addition to) EPS.

If you don't have access to the tool you used to create your images, but you still need to convert them, you can use the program `epstopdf`.

`epstopdf` writes to standard output by default, so you'll have to redirect the output to a file, as in

```
unix% epstopdf foo.eps > foo.pdf
```

To convert a whole slew of files, you could use a command such as the following (for the C-shell):

```
unix% foreach f ( `find . -type f -name '*.eps'` )
foreach? eps2pdf $f -o=$f:r.pdf
foreach? end
```

6.6 Fonts Look Fuzzy in PostScript or PDF Files

When Knuth wrote TEX, typesetting was done by trained typesetters using expensive equipment to cast molten lead into runs of type. Knuth created his own font family, Computer Modern, by writing a tool called METAFONT. METAFONT reads in programs that define various aspects of every character in a font, and generates bitmap representations of those characters at a particular resolution, ready for printing.

Unfortunately, bitmaps with resolutions suited for printing look terrible on screen. The solution is to use Type 1 PostScript fonts instead of bitmaps. If you're using PDFTEX (or PDFLATEX), you get Type 1 fonts without having to do anything special (but see Section 6.5).

If you're using `dvips` to get PostScript as an intermediate step (using `ps2pdf` or Acrobat Distiller to get PDF), you can force `dvips` to use Type 1 fonts by specifying the `-Ppdf` flag, as in

```
unix% dvips -Ppdf foo.dvi -o foo.ps
```

6.7 Debugging

One of the trickiest things about using LATEX is interpreting LATEX's sometimes cryptic error messages.

In particular, the line numbers that LaTeX reports are often not the line numbers where the problem *is*, but the line numbers where LaTeX noticed there was a problem.

One useful way of getting a bit more context to help you understand the problem is to put the line

```
\setcounter{errorcontextlines}{1000}
```

in the preamble of your document, which will provide you with a (perhaps excessive) amount of context for an error.

The most common errors are probably

- Using one of the special characters (see Section 6.1)
- Leaving off or mismatching a brace or bracket
- Leaving out or swapping arguments to a command or environment

If you've tried everything and you can't find the source of an error message, try the following procedure:

1. Create a new file, copying your preamble into it
2. Try typesetting it—if you have an error, the problem is in your preamble
3. If it typesets, copy half of your document's body into the new file, and typeset that
4. If you see your error, then continue halving the document until you narrow it down to the problem section
5. If you don't see your error, try the other half

7 Resources

There are lots of great resources available for using TeX and LaTeX. Here are a few (there are also links available online at `http://www.math.hmc.edu/computing/support/tex/`).

7.1 Online Documentation

Much of the documentation for TEX and LATEX is available online, as part of the TEX system. teTEX, the TEX system installed on the math lab computers, includes a script called `texdoc` to access this documentation. All you have to do is type `texdoc` followed by a string that you believe is the name of the document you're looking for. For example, `texdoc booktabs` will give you the documentation for the booktabs package that I used to create the tables in this document.

Unfortunately, `texdoc` only works for documentation that is sensibly named. The authors of the graphics package, for instance, called their manual `grfguide`. Still others decided that `manual` was a good name for their manual (after all, it's the only *manual* in their distribution).

Sometimes you can find documentation using the `locate` command, which lists all the files on your system that match a string that you provide. For example, you could find `grfguide` by trying `locate graphics` and `grep` ping out the results with `texmf` in them, and passing that list to another `grep` for the string `doc`:

```
unix% locate graphics | grep texmf | grep doc
```

Another hard to find, but very useful, document, is the "User's Guide for the amsmath Package" (**?**), which is called `amsldoc`.

7.2 UK-TUG FAQ

The primary list of frequently asked questions in the TEX world is the UK TUG FAQ, available at `http://www.tex.ac.uk/cgi-bin/texfaq2html`. If you're not sure how to do something, or you've got a problem that you're pretty sure isn't being caused by a typo, check here first.

7.3 `comp.text.tex`

If you can't find an answer in the UK-TUG FAQ, then your next step is to check `comp.text.tex`, the Usenet newsgroup devoted to TEX and LATEX. Chances are, whatever your problem is, someone else already had it, asked about it on `c.t.t`, and got an answer. Thanks to Google, Usenet's past is preserved in an easily searchable format. Go to Google Groups (`http://groups.google.com/`), type in some search terms, and check out the answers. (If you specify `group:comp.text.tex` at the end of your search terms, you'll only see results from `comp.text.tex`.)

8 Books

`http://www.math.hmc.edu/computing/support/tex/` has some brief reviews of a number of significant books about TEX and LATEX.

My pick for the best introductory/reference book is the third edition of George Grätzer's *Math into LATEX* (**?**).[5] It's the only book I'm aware of that discusses the latest version of AMSLATEX in depth. It also has excellent reference tables and a thorough index.

Another book I highly recommend is Lyn Dupré's *BUGS in Writing* (**?**). Dupré is one of Addison Wesley's senior editors, and has edited many of the most significant books published by Addison Wesley. *BUGS* is an accessible guide to writing clearly and effectively. It's the kind of book you leave in the bathroom so you'll always have something interesting and amusing to read. Learning how to write better is almost a byproduct!

If you get serious about typesetting, and want to start doing some fancy page design or want to be sure you're using the right kind of type, Robert Bringhurst's *The Elements of Typographic Style* (**?**) will show you the way.

9 Acknowledgments

Thanks to Darryl Yong, who wrote a similar document in November, 2000, from which I borrowed some ideas. Thanks also to Zeke Burgess, who wrote the `thesis.cls` class and, presumably, the original version of the `samplethesis.tex` file from which I stole some of the mathematical examples, and Jeremy Rouse, who modified that sample file.

[5] Which I edited.