LaTeX Wikibook

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Mon, 26 Sep 2011 17:12:46 UTC

Contents

Articles

Introduction

What is TeX?

TeX (X or chi is pronounced as in Scottish *loch*) is a low-level markup and programming language created by Donald Knuth to typeset documents attractively and consistently. Its name originates from the Greek word "τεχνολογία" (*technologìa*), which translates as "technology" in English; its first syllable is "τεχ", similar to *TeX* in the Latin alphabet. According to a different approach the name originates from the Greek word "τέχνη" (*techni*), which translates as *art* in English. In the second case the first syllable is "τεχ" again.

Knuth started writing the TeX typesetting engine in 1977 to explore the potential of the digital printing equipment that was beginning to infiltrate the publishing industry at that time, especially in the hope that he could reverse the trend of deteriorating typographical quality that he saw affecting his own books and articles.

TeX is a programming language, in the sense that it supports the if-else construct, you can make calculations with it (that are performed while compiling the document), etc., but you would find it very hard to do anything else but typesetting with it. The fine control TeX offers makes it very powerful, but also difficult and time-consuming to use. TeX is renowned for being extremely stable, for running on many different kinds of computers, and for being virtually bug free.

Nowadays practically nobody uses plain TeX to produce documents in the TeX language. Instead, different TeX distributions such as **LaTeX** are used to save time, automate certain tasks and reduce user introduced errors.

What is LaTeX?

LaTeX (pronounced either "Lah-tech" or "Lay-tech") is a macro package based on TeX created by Leslie Lamport. Its purpose is to simplify TeX typesetting, especially for documents containing mathematical formulae.

Many later authors have contributed extensions, called *packages* or *styles*, to LaTeX. Some of these are bundled with most TeX/LaTeX software distributions; more can be found in the Comprehensive TeX Archive Network (CTAN [1]).

Since LaTeX comprises a group of TeX commands, LaTeX document processing is essentially programming. You create a text file in LaTeX markup. The LaTeX macro reads this to produce the final document.

This approach has some disadvantages in comparison with a WYSIWYG (What You See Is What You Get) program such as Openoffice.org Writer or Microsoft Word.

In LaTeX:

- You don't (usually) see the final version of the document when editing it.
- You generally need to know the necessary commands for LaTeX markup.
- It can sometimes be difficult to obtain a certain look for the document.

On the other hand, there are certain advantages to the LaTeX approach:

- Document sources can be read with any text editor and understood, unlike the complex binary and XML formats used with WYSIWYG programs.
- You can concentrate purely on the structure and contents of the document, not get caught up with superficial layout issues.
- You don't need to manually adjust fonts, text sizes, line heights nor text flow for readability, as LaTeX takes care of them automatically.
- In LaTeX the document stucture is visible to the user, and can be easily copied to another document. In WYSIWYG applications it is often not obvious how a certain formatting was produced, and it might be impossible to copy it directly for use in another document.

- The layout, fonts, tables and so on are consistent throughout the document.
- Mathematical formulae can be easily typeset.
- Indexes, footnotes, citations and references are generated easily.
- You are forced to structure your documents correctly.

The LaTeX-like approach can be called WYSIWYM, i.e. What You See Is What You Mean: you can't see what the final version will look like while typing. Instead you see the logical structure of the document. LaTeX takes care of the formatting for you.

The LaTeX document is a plain text file containing the content of the document, with additional markup. When the source file is processed by the macro package, it can produce documents in several formats. LaTeX natively supports DVI and PDF, but by using other software you can easily create PostScript, PNG, JPG, etc.

Other TeX distributions

When searching for information on LaTeX, you might also stumble upon XeTeX, ConTeXt, LuaTeX or other names with a -TeX suffix. Some of them are TeX distributions, others are TeX engines. They differ from LaTeX in many ways, for example XeTeX is a TeX engine which uses Unicode and supports widely popular `.ttf` and `.otf` fonts, and ConTeXt is a TeX distribution with a very consistent and easy syntax and support for pdfTeX, XeTeX and LuaTeX engines. Before using code snippets or packages found on the web, you should always check they are written in LaTeX.

Prerequisites

At a minimum, you'll need a TeX distribution, a good text editor and a DVI or PDF viewer. But, if this is the first time you are trying out LaTeX, you don't even need to install anything. Just create an user account at:

- ScribTeX.com [2]

and continue this tutorial in the next chapter. ScribTeX is a web based online editor for LaTeX documents with collaboration capabilities allowing you to experiment with LaTeX syntax without having to bother with installing and configuring a distribution and an editor. When you later feel that you would benefit from having a standalone LaTeX installation, you can return to this chapter and follow the instructions below.

Installing a distribution

If you want to use LaTeX locally on your computer, you generally need to install a TeX distribution. TeX distributions are packaged collections of packages and programs that enable you to typeset without having to manually fetch files and configure things. The recommended distributions for each of the major operating systems are:

- TeX Live [3] is a major TeX distribution for Unix/Linux, Mac OS and Windows.
- MiKTeX [4] is a Windows-specific distribution.
- MacTeX [5] is a Mac OS-specific distribution based on TeX Live.

Windows

TeX live and MikTeX have easy installers that take care of setting up the environment and downloading packages.

- TeX Live can be downloaded here [6].
- MiKTeX can be downloaded here [7].

Linux

- Ubuntu has a 2009 version of TeX Live in the repositories, so you can use: `sudo apt-get install texlive`. Here [8] is a script to automate the installation of TeX Live 2010 on Ubuntu.
- Fedora has only 2007 version of TeX Live, but luckily there is a good repository here [9] that you can use to directly install the latest version: `yum install texlive` (plus `yum install texlive-latex` and any of the `texlive-scheme-` packages).

If your distribution does not have the TeX Live packages, you should report a wish to the bug tracking system. In the worst case you will need to download TeX Live [6] yourself and run the installer by hand. It is not hard, but requires that you know how to use a console and make multiple choices during the installation.

Mac OS

- Download MacTeX.mpkg.zip on the MacTeX page [5], unzip it and follow the instructions.

Getting a text editor

You will also need a text editor to write LaTeX code. You should use a text editor (e.g. Notepad), not a word processor (Word, Openoffice). Dedicated LaTeX editors are more useful than plain text editors, because they usually have autocompletion of commands, spell and error checking and handy macros.

TeXworks

TeXworks [10] is a dedicated TeX editor that is included in MiKTeX and TeX Live. It was developed with the idea that a simple interface is better than a cluttered one, and thus to make it easier for people in their early days with LaTeX to get to what they want to do: write their documents. TeXworks originally came about precisely because a math professor wanted his students to have a better initial experience with LaTeX.

You can install TeXworks with the package manager of your Linux distribution or choose it as an install option in the Windows or Mac installer.

LyX

LyX [11] is a popular LaTeX editor for Windows, Linux and Mac OS. It contains formula and table editors and shows visual clues of the final document on the screen enabling users to write LaTeX documents without worrying about the actual syntax.

Kile

Kile [12] is a LaTeX editor for KDE [13] (cross platform), providing a powerful GUI for editing multiple documents and compiling them with many different TeX compilers. Kile is based on Kate editor, has a quick access toolbar for symbols, document structure viewer, a console and customizable build options. Kile can be run in all operating systems that can run KDE.

TeXmaker

TeXmaker [14] is a cross-platform editor very similar to Kile in features and user interface. In addition it has its own PDF viewer.

TeXnicCenter

TeXnicCenter [15] is a popular free and open source LaTeX editor for Windows. It also has a similar user interface to TeXmaker and Kile.

BaKoMa TeX

BaKoMa TeX [16] is a editor for Windows with WYSIWYG-like features. It takes care of compiling the LaTeX source and updating it constantly to view changes to document almost in real time.

TeXShop

TeXShop [17] is a TeXworks-like editor for Mac OS.

gedit-latex-plugin

Gedit with gedit-latex-plugin [18] is also worth trying out for users of GNOME. GEdit is a cross-platform application for Windows, Mac, and Linux

Gummi

Gummi [19] is a LaTeX editor for Linux, which compiles the output of pdflatex in realtime and shows it on the right half of the screen.

Emacs

Emacs [20] is a general purpose text processing system. When used in combination with Auctex and Reftex (extensions that may be installed into the Emacs program), Emacs provides a complete LaTeX editing environment complete with table of contents view, document preview and many other features. Emacs is a very mature editing system with a unique set of keyboard commands.

LaTeXila

LaTeXila [21] is another text editor for Linux (Gnome).

Viewers

Finally, you will need a viewer for the files LaTeX outputs. Normally LaTeX saves the final document as a `.dvi` (Device independent file format), but you will rarely want it to. DVI files do not contain embedded fonts and many document viewers are unable to open them.

Usually you will use a LaTeX compiler like `pdflatex` to produce a PDF file directly, or a tool like `dvi2pdf` to convert the DVI file to PDF format. Then you can view the result with your preferred PDF viewer (Adobe Reader, Okular, Evince, Sumatra, Foxit).

Practically all LaTeX distributions have a DVI viewer for viewing the default output of `latex`, and also tools such as `dvi2pdf` for converting the result automatically to PDF and PS formats.

Applications within a distribution

Here are the main programs you expect to find in any LaTeX distribution:

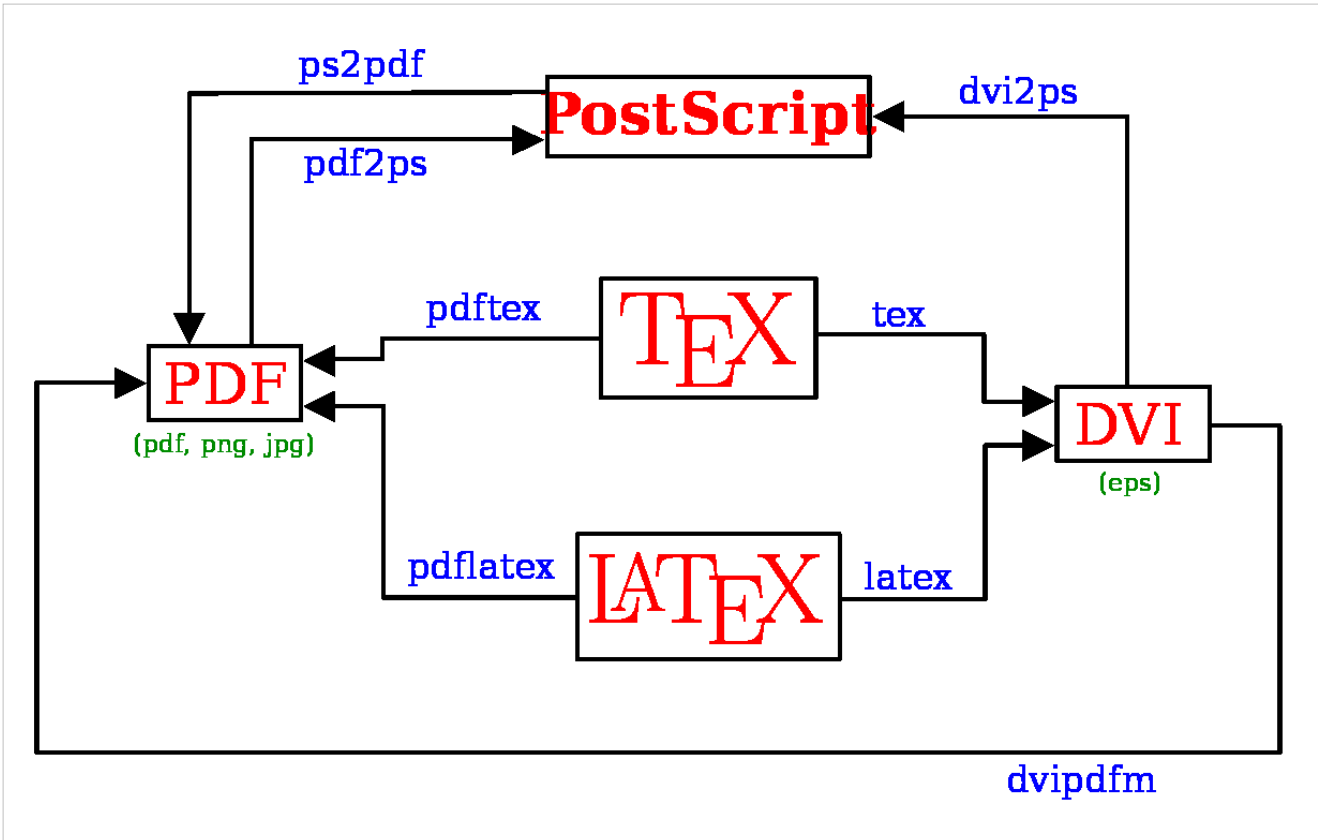
- `latex` compiler reads a LaTeX `.tex` file and creates a `.dvi`
- `pdflatex` compiler reads a LaTeX `.tex` file and creates a `.pdf`
- `dvi2ps` converts the `.dvi` file to `.ps` (postscript).
- `dvi2pdf` converts the `.dvi` file to `.pdf` (`dvi2pdfm` is an improved version).

Also `tex` and `pdftex` compilers are included, but you should be careful with using them, because they support only plain TeX. Note that since LaTeX is just a collection of macros for TeX if you compile a plain TeX document with a LaTeX compiler (such as `pdflatex`) it will work while the opposite is not true: if you try to compile a LaTeX source with a TeX compiler you will get many errors.

When LaTeX was created, the only format it could create was DVI; later PDF support was added by `pdflatex`. PDF files can be created with both `pdflatex` and `dvipdfm`. The output of `pdflatex` takes direct advantage of modern features of PDF such as hyperlinks and embedded fonts, which are not part of DVI. Passing through DVI imposes limitations of its older format. On the other hand, some packages, such as PSTricks, exploit the process of conversion to DVI, and therefore will not work with pdflatex. Some of those packages embed information in the DVI that doesn't appear when the DVI is viewed, but reemerges when the DVI is converted to another, newer format.

You would write your document slightly differently depending on the compiler you are using (`latex` or `pdflatex`). But as we will see later it is possible to add a sort of abstraction layer to hide the details of which compiler you're using, while the compiler can handle the translation itself.

The following diagram shows the relationships between the LaTeX source code and the formats you can create from it:



The boxed red text represents the file formats, the blue text on the arrows represents the commands you have to use, the small dark green text under the boxes represents the image formats that are supported. Any time you pass

through an arrow you lose some information, which might decrease the quality of your document. Therefore, in order to achieve the highest quality in your output file, you should choose the shortest route to reach your target format. This is probably the most convenient way to obtain an output in your desired format anyway. Starting from a LaTeX source, the best way is to use only *latex* for a DVI output or *pdflatex* for a PDF output, converting to PostScript only when it is necessary to print the document.

Most of the programs should be already within your LaTeX distribution; the others come with w:Ghostscript, which is a free and multi-platform software as well.

Chapter ../Export To Other Formats/ discusses more about exporting LaTeX source to other file formats.

What next?

Now you should check that your editor and LaTeX distribution are functioning properly by trying to compile the first example source in the beginning of the next chapter.

Throughout this book you should also utilise other means for learning about LaTeX. Good sources are:

- the `#latex` IRC channel on Freenode,
- the TeX Stack Exchange [22] Q&A,
- the TeX [23] FAQ,
- and the TeXample.net [24] Community.

References

[1] http://www.ctan.org
[2] http://scribtex.com/
[3] http://www.tug.org/texlive/
[4] http://www.miktex.org/
[5] http://www.tug.org/mactex/
[6] http://www.tug.org/texlive/acquire.html
[7] http://miktex.org/
[8] http://alexkrispin.wordpress.com/2010/10/12/a-script-to-automate-the-installation-of-texlive-2010/
[9] http://fedoraproject.org/wiki/Features/TeXLive
[10] http://www.tug.org/texworks/
[11] http://www.lyx.org/
[12] http://kile.sourceforge.net/
[13] http://en.wikipedia.org/wiki/KDE_Software_Compilation_4
[14] http://www.xm1math.net/texmaker/
[15] http://www.texniccenter.org/
[16] http://bakoma-tex.com/menu/about.php
[17] http://en.wikipedia.org/wiki/TeXShop
[18] http://www.michaels-website.de/gedit-latex-plugin/
[19] http://gummi.midnightcoding.org/
[20] http://www.gnu.org/software/emacs
[21] http://latexila.sourceforge.net/
[22] http://tex.stackexchange.com/
[23] http://www.tex.ac.uk/cgi-bin/texfaq2html
[24] http://www.texample.net/

Absolute Beginners

This tutorial is aimed at getting familiar with the bare bones of LaTeX.

Before starting, ensure you have LaTeX installed on your computer (see Installation for instructions of what you will need). We begin by creating the source LaTeX file, and then take you through how to feed this file through the LaTeX system to produce quality output, such as postscript or PDF.

The LaTeX source

LaTeX uses a markup language in order to describe document structure and presentation. LaTeX converts your source text, combined with the markup, into a high quality document. For the purpose of analogy, web pages work in a similar way: the HTML is used to describe the document, but it is your browser that presents it in its full glory - with different colours, fonts, sizes, etc.

The input for LaTeX is a plain ASCII text file. You can create it with any text editor. It contains the text of the document, as well as the commands that tell LaTeX how to typeset the text.

For the truly impatient, a minimal example looks something like the following (the commands will be explained later):

```
\documentclass{article} \begin{document} Hello world! \end{document}
```

Spaces

"Whitespace" characters, such as blank or tab, are treated uniformly as "space" by LaTeX. Several consecutive whitespace characters are treated as one "space". Whitespace at the start of a line is generally ignored, and a single line break is treated as “whitespace.” An empty line between two lines of text defines the end of a paragraph. Several empty lines are treated the same as one empty line. The text below is an example. On the left hand side is the text from the input file, and on the right hand side is the formatted output.

```
It does not matter whether you enter one or several
spaces after a word. An empty line starts a new
paragraph.
```

It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.

Special Characters

The following symbols are reserved characters that either have a special meaning under LaTeX or are unavailable in all the fonts. If you enter them directly in your text, they will normally not print, but rather make LaTeX do things you did not intend.

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

As you will see, these characters can be used in your documents all the same by adding a prefix backslash:

```
\# \$ \% \textasciicircum{} \& \_ \{ \} \~{} \textbackslash{}
```

The other symbols and many more can be printed with special commands in mathematical formulae or as accents.

The backslash character `\` can *not* be entered by adding another backslash in front of it (`\\`); this sequence is used for line breaking. For introducing a backslash in math mode, you can use `\backslash` instead.

The command `\~` produces a tilde which is placed over the next letter. For example `\~n` gives ñ. To produce just the character ~, use `\~{ }` which places a ~ over an empty box.

Similarly, the command `\^` produces a hat over the next character, for example `\^{o }` produces ô. If you need in text to display the ^ symbol you have to use `\textasciicircum` .

If you want to insert text that might contain several particular symbols (such as URIs), you can consider using the `\verb` command, that will be discussed later in the section on formatting.

LaTeX Commands

LaTeX commands are case sensitive, and take one of the following two formats:

- They start with a backslash `\` and then have a name consisting of letters only. Command names are terminated by a space, a number or any other "non-letter".
- They consist of a backslash `\` and exactly one non-letter.

Some commands need an argument, which has to be given between curly braces `{ }` after the command name. Some commands support optional parameters, which are added after the command name in square brackets `[ ]` . The general syntax is:

```
\commandname[option1,option2,...]{argument1}{argument2}...
```

LaTeX environments

Environments in LaTeX have a role that is quite similar to commands, but they usually have effect on a wider part of the document. Their syntax is:

```
\begin{environmentname} text to be influenced \end{environmentname}
```

Between the `\begin` and the `\end` you can put other commands and nested environments. In general, environments can accept arguments as well, but this feature is not commonly used and so it will be discussed in more advanced parts of the document.

Anything in LaTeX can be expressed in terms of commands and environments.

Comments

When LaTeX encounters a `%` character while processing an input file, it ignores the rest of the current line, the line break, and all whitespace at the beginning of the next line.

This can be used to write notes into the input file, which will not show up in the printed version.

```
This is an % stupid % Better: instructive <----
example: Supercal% ifragilist% icexpialidocious
```

This is an example: Supercalifragilisticexpialidocious

Note that the `%` character can be used to split long input lines that do not allow whitespace or line breaks, as with Supercali...cious above.

The core LaTeX language does not have a predefined syntax for commenting out regions spanning multiple lines. Refer to multi-line comments for simple workarounds.

Input File Structure

When LaTeX processes an input file, it expects it to follow a certain structure. Thus every input file must start with the command

```
\documentclass{...}
```

This specifies what sort of document you intend to write. After that, you can include commands that influence the style of the whole document, or you can load packages that add new features to the LaTeX system. To load such a package you use the command

```
\usepackage{...}
```

When all the setup work is done, you start the body of the text with the command

```
\begin{document}
```

Now you enter the text mixed with some useful LaTeX commands. At the end of the document you add the

```
\end{document}
```

command, which tells LaTeX to call it a day. Anything that follows this command will be ignored by LaTeX. The area between `\documentclass` and `\begin{document}` is called the *preamble*.

A Typical Command Line Session

LaTeX itself does not have a GUI (graphical user interface), since it is just a program that crunches away at your input files, and produces either a DVI or PDF file. Some LaTeX installations feature a graphical front-end where you can click LaTeX into compiling your input file. On other systems there might be some typing involved, so here is how to coax LaTeX into compiling your input file on a text based system. Please note: this description assumes that you already have a working LaTeX installation on your computer.

1. Edit/Create your LaTeX input file. This file must be plain ASCII text. On Unix all the editors will create just that. On Windows you might want to make sure that you save the file in ASCII or Plain Text format. When picking a name for your file, make sure it bears a `.tex` extension.
2. Run LaTeX on your input file. If successful you will end up with a `.dvi` file. It may be necessary to run LaTeX several times to get the table of contents and all internal references right. When your input file has a bug LaTeX will tell you about it and stop processing your input file.

Type ctrl-D to get back to the command line.

```
latex foo.tex
```

Now you may view the DVI file. On Unix with X11 you can type `xdvi foo.dvi`, on Windows you can use a program called *yap* (yet another previewer). (Now evince and okular, the standard document viewers for many Linux distributions are able to view DVI files.)

You can run a similar procedure with `pdflatex` to produce a PDF document from the original `.tex` source. Similar to above, type the commands:

```
pdflatex foo.tex
```

Now you may view the PDF file, `foo.pdf`.

Our first document

Now we can create our first document. We will produce the absolute bare minimum that is needed in order to get some output; the well known **Hello World!** approach will be suitable here.

- Open your favorite text-editor. vim, emacs, Notepad++, and other text editors will have syntax highlighting that will help to write your files.
- Reproduce the following text in your editor. This is the LaTeX source.

```
% hello.tex - Our first LaTeX example! \documentclass{article} \begin{document}
Hello World! \end{document}
```

- Save your file as `hello.tex`.

What does it all mean?

`% hello.tex - Our first LaTeX example!`	The first line is a *comment*. This is because it begins with the percent symbol (%); when LaTeX sees this, it simply ignores the rest of the line. Comments are useful for people to annotate parts of the source file. For example, you could put information about the author and the date, or whatever you wish.
`\documentclass{article}`	This line is a command and tells LaTeX to use the `article` document class. A document class file defines the formatting, which in this case is a generic article format. The handy thing is that if you want to change the appearance of your document, substitute article for another class file that exists.
`\begin{document}`	This line is the beginning of the environment called `document`; it alerts LaTeX that content of the document is about to commence. Anything above this command is known generally to belong in the *preamble*.
`Hello World!`	This was the only actual line containing real content - the text that we wanted displayed on the page.
`\end{document}`	The `document` environment ends here. It tells LaTeX that the document source is complete, anything after this line will be ignored.

As we have said before, each of the LaTeX commands begins with a backslash (`\`). This is LaTeX's way of knowing that whenever it sees a backslash, to expect some commands. Comments are not classed as a command, since all they tell LaTeX is to ignore the line. Comments never affect the output of the document.

Generating the document

It is clearly not going to be the most exciting document you have ever seen, but we want to see it nonetheless. I am assuming that you are at a command prompt, already in the directory where `hello.tex` is stored.

1. Type the command: `latex hello` (the `.tex` extension is not required, although you can include it if you wish)
2. Various bits of info about LaTeX and its progress will be displayed. If all went well, the last two lines displayed in the console will be:

```
Output written on hello.dvi (1 page, 232 bytes).
Transcript written on hello.log.
```

This means that your source file has been processed and the resulting document is called *hello.dvi*, which takes up 1 page and 232 bytes of space. This way you created the DVI file, but with the same source file you can create a PDF document. The steps are exactly the same as before, but you have to replace the command `latex` with `pdflatex`:

1. Type the command: `pdflatex hello` (as before, the `.tex` extension is not required)

2. Various bits of info about LaTeX and its progress will be displayed. If all went well, the last two lines displayed in the console will be:

```
Output written on hello.pdf (1 page, 5548 bytes).
Transcript written on hello.log.
```

you can notice that the PDF document is bigger than the DVI, even if it contains exactly the same information. The main differences between the DVI and PDF formats are:

- **DVI** needs less disk space and it is faster to create. It does not include the fonts within the document, so if you want the document to be viewed properly on another computer, there must be all the necessary fonts installed. It does not support any interactivity such as hyperlinks or animated images. DVI viewers are not very common, so you can consider using it for previewing your document while typesetting.
- **PDF** needs more disk space and it is slower to create, but it includes all the necessary fonts within the document, so you will not have any problem of portability. It supports internal and external hyperlinks. It also supports advanced typographic features: hanging punctuation, font expansion and margin kerning resulting in more flexibility available to the TeX engine and better looking output. Nowadays it is the *de facto* standard for sharing and publishing documents, so you can consider using it for the final version of your document.

About now, you saw you can create both DVI and PDF document from the same source. This is true, but it gets a bit more complicated if you want to introduce images or links. This will be explained in detail in the next chapters, but for now assume you can compile in both DVI and PDF without any problem.

Note, in this instance, due to the simplicity of the file, you only need to run the LaTeX command once. However, if you begin to create complex documents, including bibliographies and cross-references, etc, LaTeX needs to be executed multiple times to resolve the references. But this will be discussed in the future when it comes up.

Basics

Document Classes

The first information LaTeX needs to know when processing an input file is the type of document the author wants to create. This is specified with the `\documentclass` command.

```
\documentclass[options]{class}
```

Here class specifies the type of document to be created. The LaTeX distribution provides additional classes for other documents, including letters and slides. The options parameter customizes the behavior of the document class. The options have to be separated by commas.

Example: an input file for a LaTeX document could start with the line

```
\documentclass[11pt,twoside,a4paper]{article}
```

which instructs LaTeX to typeset the document as an article with a base font size of 11 points, and to produce a layout suitable for double sided printing on A4 paper.

Here are some document classes that can be used with LaTeX:

Document Classes

`article`	for articles in scientific journals, presentations, short reports, program documentation, invitations, ...
`IEEEtran`	for articles with the IEEE Transactions format.
`proc`	a class for proceedings based on the article class.
`minimal`	is as small as it can get. It only sets a page size and a base font. It is mainly used for debugging purposes.
`report`	for longer reports containing several chapters, small books, thesis, ...
`book`	for real books
`slides`	for slides. The class uses big sans serif letters.
`memoir`	for changing sensibly the output of the document. It is based on the `book` class, but you can create any kind of document with it [1]
`letter`	for writing letters.
`beamer`	for writing presentations (see LaTeX/Presentations).

The most common options for the standard document classes are listed in the following table:

Document Class Options

`10pt, 11pt, 12pt`	Sets the size of the main font in the document. If no option is specified, 10pt is assumed.
`a4paper, letterpaper,...`	Defines the paper size. The default size is `letterpaper`; However, many European distributions of TeX now come pre-set for A4, not Letter, and this is also true of all distributions of pdfLaTeX. Besides that, `a5paper, b5paper, executivepaper`, and `legalpaper` can be specified.
`fleqn`	Typesets displayed formulas left-aligned instead of centered.
`leqno`	Places the numbering of formulas on the left hand side instead of the right.
`titlepage, notitlepage`	Specifies whether a new page should be started after the document title or not. The article class does not start a new page by default, while report and book do.
`onecolumn, twocolumn`	Instructs LaTeX to typeset the document in one column or two columns.
`twoside, oneside`	Specifies whether double or single sided output should be generated. The classes `article` and `report` are single sided and the `book` class is double sided by default. Note that this option concerns the style of the document only. The option `twoside` does not tell the printer you use that it should actually make a two-sided printout.
`landscape`	Changes the layout of the document to print in landscape mode.
`openright, openany`	Makes chapters begin either only on right hand pages or on the next page available. This does not work with the `article` class, as it does not know about chapters. The `report` class by default starts chapters on the next page available and the `book` class starts them on right hand pages.
`draft`	makes LaTeX indicate hyphenation and justification problems with a small square in the right-hand margin of the problem line so they can be located quickly by a human. It also suppresses the inclusion of images and shows only a frame where they would normally occur.

For example, if you want a report to be in 12pt type on A4, but printed one-sided in draft mode, you would use:

```
\documentclass[12pt,a4paper,oneside,draft]{report}
```

Packages

While writing your document, you will probably find that there are some areas where basic LaTeX cannot solve your problem. If you want to include graphics, colored text or source code from a file into your document, you need to enhance the capabilities of LaTeX. Such enhancements are called packages. Packages are activated with the

```
\usepackage[options]{package}
```

command, where package is the name of the package and options is a list of keywords that trigger special features in the package. Some packages come with the LaTeX base distribution. Others are provided separately.

Modern TeX distributions come with a large number of packages pre-installed. If you are working on a Unix system, use the command `texdoc` for accessing package documentation. For more information, see the Packages section.

Files you might Encounter

When you work with LaTeX you will soon find yourself in a maze of files with various extensions and probably no clue. The following list explains the most common file types you might encounter when working with TeX:

Common file extensions in LaTeX

`.aux`	A file that transports information from one compiler run to the next. Among other things, the `.aux` file is used to store information associated with cross-references.
`.bbl`	Bibliography file output by BiBTeX and used by LaTeX
`.bib`	Bibliography database file
`.blg`	BiBTeX log file.
`.bst`	BiBTeX style file.
`.cls`	Class files define what your document looks like. They are selected with the `\documentclass` command.
`.dtx`	Documented TeX. This is the main distribution format for LaTeX style files. If you process a `.dtx` file you get documented macro code of the LaTeX package contained in the `.dtx` file.
`.ins`	The installer for the files contained in the matching `.dtx` file. If you download a LaTeX package from the net, you will normally get a `.dtx` and a `.ins` file. Run LaTeX on the `.ins` file to unpack the `.dtx` file.
`.fd`	Font description file telling LaTeX about new fonts.
`.dvi`	Device Independent File. This is the main result of a LaTeX compile run with *latex*. You can look at its content with a DVI previewer program or you can send it to a printer with dvips or a similar application.
`.pdf`	Portable Document Format. This is the main result of a LaTeX compile run with *pdflatex*. You can look at its content or print it with any PDF viewer.
`.log`	Gives a detailed account of what happened during the last compiler run.
`.toc`	Stores all your section headers. It gets read in for the next compiler run and is used to produce the table of contents.
`.lof`	This is like `.toc` but for the list of figures.
`.lot`	And again the same for the list of tables.
`.idx`	If your document contains an index. LaTeX stores all the words that go into the index in this file. Process this file with makeindex.
`.ind`	The processed .idx file, ready for inclusion into your document on the next compile cycle.
`.ilg`	Logfile telling what makeindex did.
`.sty`	LaTeX Macro package. This is a file you can load into your LaTeX document using the \usepackage command.
`.tex`	LaTeX or TeX input file. It can be compiled with latex.

Big Projects

When working on big documents, you might want to split the input file into several parts. LaTeX has three commands to insert a file into another when building the document.

The simplest is the \input command:

```
\input{filename}
```

\input inserts the contents of another file, named *filename.tex*; note that the .tex extension is omitted. For all practical purposes, `\input` is no more than a simple, automated cut-and-paste of the source code in *filename.tex*.

The other main inclusion command is \include:

```
\include{filename}
```

The `\include` command is different from `\input` in that it's the output that is added instead of the commands from the other files. Therefore a new page will be created at every `\include` command, which makes it appropriate to use it for large entities such as book chapters.

Very large documents (that usually include many files) take a very long time to compile, and most users find it convenient to test their last changes by including only the files they have been working on. One option is to hunt down all `\include` commands in the inclusion hierarchy and to comment them out:

```
%\include{filename1}          \include{filename2}          \include{filename3}
%\include{filename4}
```

In this case, the user wants to include only *filename2.tex* and *filename3.tex*. If the inclusion hierarchy is intricate, commenting can become error-prone: page numbering will change, and any cross references won't work. A better alternative is to retain the include calls and use the `\includeonly` command *in the preamble*:

```
\includeonly{filename2,filename3}
```

This way, only `\include` commands for the specified files will be executed, and inclusion will be handled in only one place. Note that there must be no spaces between the filenames and the commas.

Remember that the input file should omit all the commands referring to the main document structure, which should be kept in the original document file. This includes lines containing usepackages, document class, and everything but the code strictly referring to the section that is to be included. In this way you'll avoid finding characters of your code in the output document, or worse, not finding anything after the included file, in case you forget to erase the

```
\end{document}
```

line of your included file.

Picking suitable filenames

Never, ever use directories (folders) or file names that contain spaces. Although your operating system probably supports them, some don't, and they will only cause grief and tears with TeX. Make filenames as short or as long as you wish, but strictly avoid spaces. Stick to lower-case letters without accents (a-z), the digits 0-9, the hyphen (-), and the full point or period (.), (similar to the conventions for a Web URL): it will let you refer to TeX files over the Web more easily and make your files more portable. Some operating systems do not distinguish between upper-case and lower-case letters, others do. Therefore it's best not to mix them.

Working in a team

See chapter ../Collaborative Writing of LaTeX Documents/.

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/memoir/memman.pdf

Document Structure

The main point of writing a text is to convey ideas, information, or knowledge to the reader. The reader will understand the text better if these ideas are well-structured, and will see and feel this structure much better if the typographical form reflects the logical and semantic structure of the content.

LaTeX is different from other typesetting systems in that you just have to tell it the logical and semantical structure of a text. It then derives the typographical form of the text according to the "rules" given in the document class file and in various style files. LaTeX allows users to structure their documents with a variety of hierarchal constructs, including chapters, sections, subsections and paragraphs.

The `document` environment

After the document class declaration, the text of your document is enclosed between two commands which identify the beginning and end of the actual document:

```
\documentclass[11pt,a4paper,oneside]{report}        \begin{document}        ...
\end{document}
```

You would put your text where the dots are. The reason for marking off the beginning of your text is that LaTeX allows you to insert extra setup specifications before it (where the blank line is in the example above: we'll be using this soon). The reason for marking off the end of your text is to provide a place for LaTeX to be programmed to do extra stuff automatically at the end of the document, like making an index.

A useful side-effect of marking the end of the document text is that you can store comments or temporary text underneath the `\end{document}` in the knowledge that LaTeX will never try to typeset them:

```
... \end{document} Don't forget to get the extra chapter from Dorando!
```

Preamble

The *preamble* is everything from the start of the LaTeX source file until the `\begin{document}` command. It normally contains commands that affect the entire document.

```
% simple.tex - A simple article to illustrate document structure.
\documentclass{article} \usepackage{mathptmx} \begin{document}
```

The first line is a comment (as denoted by the % sign). The `\documentclass` command takes an argument, which in this case is article, because that's the type of document we want to produce. It is also possible to create your own, as is often done by journal publishers, who simply provide you with their own class file, which tells LaTeX how to format your content. But we'll be happy with the standard article class for now. `\usepackage` is an important command that tells LaTeX to utilize some external macros. In this instance, we specified `mathptmx` which means LaTeX will use the Postscript Times type 1 font instead of the default ComputerModern font. And finally, the `\begin{document}`. This strictly isn't part of the preamble, but I'll put it here anyway, as it implies the end of the preamble by nature of stating that the document is now starting.

Top Matter

At the beginning of most documents there will be information about the document itself, such as the title and date, and also information about the authors, such as name, address, email etc. All of this type of information within LaTeX is collectively referred to as *top matter*. Although never explicitly specified (there is no **\topmatter** command) you are likely to encounter the term within LaTeX documentation.

A simple example:

```
\documentclass[11pt,a4paper,oneside]{report} \begin{document} \title{How to
Structure a LaTeX Document} \author{Andrew Roberts} \date{December 2004}
\maketitle \end{document}
```

The **\title** , **\author** , and **\date** commands are self-explanatory. You put the title, author name, and date in curly braces after the relevant command. The title and author are usually compulsory (at least if you want LaTeX to write the title automatically); if you omit the **\date** command, LaTeX uses today's date by default. You always finish the top matter with the **\maketitle** command, which tells LaTeX that it's complete and it can typeset the title according to the information you have provided and the class (style) you are using. If you omit `\maketitle`, the titling will never be typeset (unless you write your own).

Here is a more complicated example:

```
\title{How to Structure a \LaTeX{} Document} \author{Andrew Roberts\\ School of
Computing,\\ University of Leeds,\\ Leeds,\\ United Kingdom,\\ LS2 1HE\\
\texttt{andyr@comp.leeds.ac.uk
```

\date{\today} \maketitle }}

as you can see, you can use commands as arguments of **\title** and the others. The double backslash (****) is the LaTeX command for forced linebreak. LaTeX normally decides by itself where to break lines, and it's usually right, but sometimes you need to cut a line short, like here, and start a new one.

If there are two authors separate them with the **\and** command:

```
\title{Our Fun Document} \author{Jane Doe \and John Doe} \date{\today}
\maketitle
```

If you are provided with a class file from a publisher, or if you use the AMS article class (amsart), then you can use several different commands to enter author information. The email address is at the end, and the **\texttt** commands formats the email address using a mono-spaced font. The built-in command called **\today** will be replaced with the current date when processed by LaTeX. But you are free to put whatever you want as a date, in no set order. If braces are left empty, then the date is omitted.

Using this approach, you can create only basic output whose layout is very hard to change. If you want to create your title freely, see the Title Creation section.

Abstract

As most research papers have an abstract, there are predefined commands for telling LaTeX which part of the content makes up the abstract. This should appear in its logical order, therefore, after the top matter, but before the main sections of the body. This command is available for the document classes *article* and *report*, but not *book*.

```
\documentclass{article} \begin{document} \begin{abstract} Your abstract goes here... ... \end{abstract} ... \end{document}
```

By default, LaTeX will use the word "Abstract" as a title for your abstract, if you want to change it into anything else, e.g. "Executive Summary", add the following line in the preamble:

```
\renewcommand{\abstractname}{Executive Summary}
```

Sectioning Commands

The commands for inserting sections are fairly intuitive. Of course, certain commands are appropriate to different document classes. For example, a book has chapters but an article doesn't. Here is an edited version of some of the structure commands in use from *simple.tex*.

```
\section{Introduction} This section's content... \section{Structure} This section's content... \subsection{Top Matter} This subsection's content... \subsubsection{Article Information} This subsubsection's content...
```

Notice that you do not need to specify section numbers; LaTeX will sort that out for you. Also, for sections, you do not need to markup which content belongs to a given block, using `\begin` and `\end` commands, for example. LaTeX provides 7 levels of depth for defining sections:

Command	Level	Comment
`\part{`*part*`}`	-1	not in letters
`\chapter{`*chapter*`}`	0	only books and reports
`\section{`*section*`}`	1	not in letters
`\subsection{`*subsection*`}`	2	not in letters
`\subsubsection{`*subsubsection*`}`	3	not in letters
`\paragraph{`*paragraph*`}`	4	not in letters
`\subparagraph{`*subparagraph*`}`	5	not in letters

All the titles of the sections are added automatically to the table of contents (if you decide to insert one). But if you make manual styling changes to your heading, for example a very long title, or some special line-breaks or unusual font-play, this would appear in the Table of Contents as well, which you almost certainly don't want. LaTeX allows you to give an optional extra version of the heading text which only gets used in the Table of Contents and any running heads, if they are in effect. This optional alternative heading goes in [square brackets] before the curly braces:

```
\section[Effect on staff turnover]{An analysis of the effect of the revised recruitment policies on staff turnover at divisional headquarters}
```

Section numbering

Numbering of the sections is performed automatically by LaTeX, so don't bother adding them explicitly, just insert the heading you want between the curly braces. Parts get roman numerals (Part I, Part II, etc.); chapters and sections get decimal numbering like this document, and appendices (which are just a special case of chapters, and share the same structure) are lettered (A, B, C, etc.). You can change the depth to which section numbering occurs, so you can turn it off selectively. By default it is set to 2. If you only want parts, chapters, and sections numbered, not

subsections or subsubsections etc., you can change the value of the `secnumdepth` counter using the `\setcounter` command, giving the depth level from the previous table. For example, if you want to change it to "1":

```
\setcounter{secnumdepth}{1}
```

A related counter is `tocdepth`, which specifies what depth to take the Table of Contents to. It can be reset in exactly the same way as `secnumdepth`. For example:

```
\setcounter{tocdepth}{3}
```

To get an unnumbered section heading which does not go into the Table of Contents, follow the command name with an asterisk before the opening curly brace:

```
\subsection*{Introduction}
```

All the divisional commands from \part* to \subparagraph* have this "starred" version which can be used on special occasions for an unnumbered heading when the setting of `secnumdepth` would normally mean it would be numbered.

If you want the unnumbered section to be in the table of contents anyway, use the `\addcontentsline` command like this:

```
\section*{Introduction} \addcontentsline{toc}{section}{Introduction}
```

Note if you use pdf bookmarks you will need to add a phantom section so that bookmark will lead to the correct place in the document:

```
\phantomsection \addcontentsline{toc}{section}{Introduction}
\section*{Introduction}
```

For chapters you will also need to clear the page (this will also correct page numbering in the ToC):

```
\cleardoublepage \phantomsection \addcontentsline{toc}{chapter}{Bibliography}
\bibliographystyle{unsrt} \bibliography{my_bib_file}
```

The value where the section numbering starts from can be set with the following command:

```
\setcounter{section}{4}
```

The next section after this command will now be numbered 5.

Any counter can be incremented/decremented with the following command:

```
\addtocounter{counter}{integer}
```

The \phantomsection command is defined in the hyperref package.

Appendices

The separate numbering of appendices is also supported by LaTeX. The `\appendix` macro can be used to indicate that following sections or chapters are to be numbered as appendices.

In the report or book classes this gives:

```
\appendix \chapter{First Appendix}
```

For the article class use:

```
\appendix \section{First Appendix}
```

Ordinary paragraphs

Paragraphs of text come after section headings. Simply type the text and leave a blank line between paragraphs. The blank line means "start a new paragraph here": it does **not** mean you get a blank line in the typeset output. For formatting paragraph indents and spacing between paragraphs, refer to the Formatting section.

Table of contents

All auto-numbered headings get entered in the Table of Contents (ToC) automatically. You don't have to print a ToC, but if you want to, just add the command `\tableofcontents` at the point where you want it printed (usually after the Abstract or Summary).

Entries for the ToC are recorded each time you process your document, and reproduced the next time you process it, so you need to re-run LaTeX one extra time to ensure that all ToC pagenumber references are correctly calculated. We've already seen how to use the optional argument to the sectioning commands to add text to the ToC which is slightly different from the one printed in the body of the document. It is also possible to add extra lines to the ToC, to force extra or unnumbered section headings to be included.

The commands `\listoffigures` and `\listoftables` work in exactly the same way as `\tableofcontents` to automatically list all your tables and figures. If you use them, they normally go after the `\tableofcontents` command. The `\tableofcontents` command normally shows only numbered section headings, and only down to the level defined by the `tocdepth` counter, but you can add extra entries with the `\addcontentsline` command. For example if you use an unnumbered section heading command to start a preliminary piece of text like a Foreword or Preface, you can write:

```
\subsection*{Preface} \addcontentsline{toc}{subsection}{Preface}
```

This will format an unnumbered ToC entry for "Preface" in the "subsection" style. You can use the same mechanism to add lines to the List of Figures or List of Tables by substituting `lof` or `lot` for `toc`. If the hyperref package is used and the link does not point correct to the chapter, the command `\phantomsection` in combination with `\clearpage` or `\cleardoublepage` can be used (see also Labels_and_Cross-referencing):

```
\cleardoublepage \phantomsection \addcontentsline{toc}{chapter}{List of Figures} \listoffigures
```

To change the title of the TOC, you have to paste this command `\renewcommand{\contentsname}{<New table of contents title>}` in your document preamble. The List of Figures (LoF) and List of Tables (LoT) names can be changed by replacing the `\contentsname` with `\listfigurename` for LoF and `\listtablename` for LoT.

Depth

The default ToC will list headings of level 3 and above. To change how deep the table of contents displays automatically the following command can be used in the preamble:

```
\setcounter{tocdepth}{4}
```

This will make the table of contents include everything down to paragraphs. The levels are defined above on this page. Note that this solution does not permit changing the depth dynamically.

In order to further tune the display or the numbering of the table of contents, for instance if the appendix should be less detailed, you can make use of the `tocvsec2` package (CTAN [1], doc [2]).

The Bibliography

Any good research paper will have a whole list of references. There are two ways to insert your references into LaTeX:

- you can embed them within the document itself. It's simpler, but it can be time-consuming if you are writing several papers about similar subjects so that you often have to cite the same books.
- you can store them in an external BibTeX file [3] and then link them via a command to your current document and use a Bibtex style [4] to define how they appear. This way you can create a small database of the references you might use and simply link them, letting LaTeX work for you.

In order to know how to add the bibliography to your document, see the Bibliography Management section.

References

[1] http://www.ctan.org/pkg/tocvsec2
[2] http://mirror.ctan.org/macros/latex/contrib/tocvsec2/tocvsec2.pdf
[3] http://www.bibtex.org
[4] http://www.cs.stir.ac.uk/~kjt/software/latex/showbst.html

Errors and Warnings

LaTeX describes what it's typesetting while it does it, and if it encounters something it doesn't understand or can't do, it will display a message saying what's wrong. It may also display warnings for less serious conditions.

Don't panic if you see error messages: it's very common to mistype or mis-spell commands, forget curly braces, type a forward slash instead of a backslash, or use a special character by mistake. Errors are easily spotted and easily corrected in your editor, and you can then run LaTeX again to check you have fixed everything. Some of the most common errors are described in next sections.

Error messages

The format of an error message is always the same. Error messages begin with an exclamation mark at the start of the line, and give a description of the error, followed by another line starting with the number, which refers to the line-number in your document file which LaTeX was processing when the error was spotted. Here's an example, showing that the user mistyped the `\tableofcontents` command:

```
! Undefined control sequence.
l.6 \tableofcotnetns
```

When LaTeX finds an error like this, it displays the error message and pauses. You must type one of the following letters to continue:

Key	Meaning
x	Stop immediately and e**x**it the program.
q	Carry on **q**uietly as best you can and don't bother me with any more error messages.
e	Stop the program but re-position the text in my **e**ditor at the point where you found the error (This only works if you're using an editor which LaTeX can communicate with).
h	Try to give me more **h**elp.
i	(followed by a correction) means **i**nput the correction in place of the error and carry on (This is only a temporary fix to get the file processed. You still have to make that correction in the editor).
r	**r**un in non-stop mode. Plow through any errors, unless too many pile up and it fails (100 errors).

Some systems (Emacs is one example) run LaTeX with a "nonstop" switch turned on, so it will always process through to the end of the file, regardless of errors, or until a limit is reached.

Warnings

Warnings don't begin with an exclamation mark: they are just comments by LaTeX about things you might want to look into, such as overlong or underrun lines (often caused by unusual hyphenations, for example), pages running short or long, and other typographical niceties (most of which you can ignore until later). Unlike other systems, which try to hide unevennesses in the text (usually unsuccessfully) by interfering with the letterspacing, LaTeX takes the view that the author or editor should be able to contribute. While it is certainly possible to set LaTeX's parameters so that the spacing is sufficiently sloppy that you will almost never get a warning about badly-fitting lines or pages, you will almost certainly just be delaying matters until you start to get complaints from your readers or publishers.

Examples

Only a few common error messages are given here: those most likely to be encountered by beginners. If you find another error message not shown here, and it's not clear what you should do, ask for help.

Most error messages are self-explanatory, but be aware that the place where LaTeX spots and reports an error may be later in the file than the place where it actually occurred. For example if you forget to close a curly brace which encloses, say, italics, LaTeX won't report this until something else occurs which can't happen until the curly brace is encountered (e.g. the end of the document!) Some errors can only be righted by humans who can read and understand what the document is supposed to mean or look like.

Newcomers should remember to check the list of special characters: a very large number of errors when you are learning LaTeX are due to accidentally typing a special character when you didn't mean to. This disappears after a few days as you get used to them.

Too many }'s

```
! Too many }'s.
l.6 \date December 2004}
```

The reason LaTeX thinks there are too many }'s here is that the opening curly brace is missing after the `\date` control sequence and before the word December, so the closing curly brace is seen as one too many (which it is!). In fact, there are other things which can follow the `\date` command apart from a date in curly braces, so LaTeX cannot possibly guess that you've missed out the opening curly brace until it finds a closing one!

Undefined control sequence

```
! Undefined control sequence.
l.6 \dtae
{December 2004}
```

In this example, LaTeX is complaining that it has no such command ("control sequence") as `\dtae`. Obviously it's been mistyped, but only a human can detect that fact: all LaTeX knows is that `\dtae` is not a command it knows about: it's undefined. Mistypings are the most common source of errors. If your editor has drop-down menus to insert common commands and environments, use them!

Not in Mathematics Mode

```
! Missing $ inserted
```

A character that can only be used in the mathematics was inserted in normal text. Either switch to mathematic mode via \begin{math}...\end{math} or use the 'quick math mode': \ensuremath{math stuff}

This can also happen if you use the wrong character encoding, for example using utf8 without "\usepackage[utf8]{inputenc}" or using iso8859-1 without "\usepackage[latin1]{inputenc}", there are several character encoding formats, make sure to pick the right one.

Runaway argument

```
Runaway argument?
{December 2004 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
\par
l.8
```

In this error, the closing curly brace has been omitted from the date. It's the opposite of the error of too many }'s, and it results in `\maketitle` trying to format the title page while LaTeX is still expecting more text for the date! As \maketitle creates new paragraphs on the title page, this is detected and LaTeX complains that the previous paragraph has ended but \date is not yet finished.

Underfull hbox

```
Underfull \hbox (badness 1394) in paragraph
at lines 28--30
[][]\LY1/brm/b/n/10 Bull, RJ: \LY1/brm/m/n/10
Ac-count-ing in Busi-
[94]
```

This is a warning that LaTeX cannot stretch the line wide enough to fit, without making the spacing bigger than its currently permitted maximum. The badness (0-10,000) indicates how severe this is (here you can probably ignore a badness of 1394). It says what lines of your file it was typesetting when it found this, and the number in square brackets is the number of the page onto which the offending line was printed. The codes separated by slashes are the typeface and font style and size used in the line. Ignore them for the moment.

This comes up if you force a linebreak, e.g., \\, and have a return before it. Normally TeX ignores linebreaks, providing full paragraphs to ragged text. In this case it is necessary to pull the linebreak up one line to the end of the previous sentence.

Overfull hbox

```
[101]
Overfull \hbox (9.11617pt too wide) in paragraph
at lines 860--861
[]\LY1/brm/m/n/10 Windows, \LY1/brm/m/it/10 see
\LY1/brm/m/n/10 X Win-
```

An overfull \hbox means that there is a hyphenation or justification problem: moving the last word on the line to the next line would make the spaces in the line wider than the current limit; keeping the word on the line would make the spaces smaller than the current limit, so the word is left on the line, but with the minimum allowed space between words, and which makes the line go over the edge.

The warning is given so that you can find the line in the code that originates the problem (in this case: 860-861) and fix it. The line on this example is too long by a shade over 9pt. The chosen hyphenation point which minimizes the error is shown at the end of the line (Win-). Line numbers and page numbers are given as before. In this case, 9pt is too much to ignore (over 3mm), and a manual correction needs making (such as a change to the hyphenation), or the flexibility settings need changing.

If the "overfull" word includes a forward slash, such as "`input/output`", this should be properly typeset as "`input\slash output`". The use of `\slash` has the same effect as using the "`/`" character, except that it can form the end of a line (with the following words appearing at the start of the next line). The "`/`" character is typically used in units, such as "`mm/year`" character, which should not be broken over multiple lines.

Missing package

```
! LaTeX Error: File `paralisy.sty' not found.
Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: sty)
Enter file name:
```

When you use the `\usepackage` command to request LaTeX to use a certain package, it will look for a file with the specified name and the filetype `.sty`. In this case the user has mistyped the name of the paralist package, so it's easy to fix. However, if you get the name right, but the package is not installed on your machine, you will need to download and install it before continuing. If you don't want to affect the global installation of the machine, you can simply download from Internet the necessary `.sty` file and put it in the same folder of the document you are compiling.

Package babel Warning: No hyphenation patterns were loaded for the language X

Although this is a warning from the Babel package and not from LaTeX, this error is very common and (can) give some strange hyphenation (word breaking) problems in your document. Wrong hyphenation rules can decrease the neatness of your document.

```
Package babel Warning: No hyphenation patterns were loaded for
(babel)                the language `Latin'
(babel)                I will use the patterns loaded for \language=0 instead.
```

This can happen after the usage of: (see LaTeX/Internationalization)

```
\usepackage[latin]{babel}
```

The solution is not difficult, just install the used language in your LaTeX distribution.

Creating a Document

Title Creation

There are several situations where you might want to create a title in a custom format, rather than in the format natively supported by LaTeX. For shorter documents such as basic articles, the output of `\maketitle` is often adequate, but longer documents (such as books and reports) often require more involved formatting. While it is possible to change the output of `\maketitle`, it can be complicated even with minor changes to the title. In such cases it is often better to create the title from scratch, and this section will show you how to accomplish this.

Standard Title Pages

Many document classes will form a title page for you. One must specify what to fill it with using these commands placed in the top matter:

```
\title{The Triangulation of Titling Data in Non-Linear Gaussian Fashion via $\rho$ Series} \date{October 31, 475} \author{John Doe\\ Magic Department, Richard Miles University \and Richard Row, \LaTeX\ Academy}
```

Commonly the date is excluded from the title page by using `\date{}`. It defaults to `\today` if not in the source file.

To form a title page, use

```
\maketitle
```

This should go after the preceding commands. For most document styles, this will form a separate page, while the `article` document style will place the title on the top of the first page. Note that the `abstract` environment should precede the `\maketitle` command in AMS documents.

Footnotes within the title page can be specified with the `\thanks` command. For example, one may add

```
\author{John Doe\thanks{Funded by NASA Grant \#42}}
```

The `\thanks` command can also be used in the `\title` too.

It is dependent on the document class which commands are used in the title page generated by `\maketitle`. For example, the `amsart` uses commands such as `\address`, `\dedicatory`, `\email` and more in the title page while other classes may only use `\title`.

Custom Title Pages

Create the title

Normally, the benefit of using LaTeX instead of traditional word processing programs is that LaTeX frees you to concentrate on content by handling margins, justification, and other typesetting concerns. On the other hand, if you want to write your own title format, it is exactly the opposite: you have to take care of everything--this time LaTeX will do nothing to help you. It can be challenging to create your own title format since LaTeX was not designed to be graphically interactive in the adjustment of layout. The process is similar to working with raw HTML with the added step that each time you want to see how your changes look, you have to re-compile the source. While this may seem like a major inconvenience, the benefit is that once the customized title format has been written, it serves as a

template for all other documents that would use the title format you have just made. In other words, once you have a layout you like, you can use it for any other documents where you would like the same layout without any additional fiddling with layout.

First step: since you'll be working only on the first page of your document and you'll have to compile very often, you don't have to compile the whole document each time, you only need to take a look at the first page. That is why we'll first create a dummy document for preparing the title and then we'll simply include it within the existing big document we are writing. Call the dummy document `test_title.tex` and put the following code in it:

```
\documentclass[pdftex,12pt,a4paper]{report}       \usepackage[pdftex]{graphicx}
\newcommand{\HRule}{\rule{\linewidth}{0.5mm}}                    \begin{document}
\input{./title.tex} \end{document}
```

It is meant to be compiled with `pdflatex` to create a PDF in output. It is a very basic document, but take care that it has the same settings of the document you are writing, so the output won't change when you include the title in your document. In this case (see the first line) the font size is set to 12pt and the paper size is an A4. The package `graphicx` is included to insert an image in the title. Then a command is defined called `\HRule`; it will just insert a horizontal line whose length is like the size of the paper and whose thickness is 0.5 mm. If you want you can change its settings in the definition. Finally the document starts and it simply includes the `title.tex` file, that must be placed in the same directory of our dummy file `test_title.tex`.

Now create the `title.tex` and write in it:

```
\begin{titlepage} \end{titlepage}
```

all the things you want to put in the title must be inside the `titlepage` environment. Now if you compile `test_title.tex` you will see a preview of your title in the `test_title.pdf` file. Here is what you need to know to write your title:

Alignment

> if you want to center some text just use `\begin{center} ... \end{center}`. If you want to align it differently you can use the environment `flushright` for *right*-alignment and `flushleft` for *left*-alignment.

Images

> the command for including images is the following (the example is for a small logo, but you can introduce any image of any size): `\includegraphics[width=0.15\textwidth]{./logo}`. There is no `\begin{figure}` as you usually do because you don't want it to be floating, you just want it there where you placed it. When handling it, remember that it is considered like a big box by the TeX engine.

Text size

> If you want to change the size of some text just place it within brackets, *{like this}*, and you can use the following commands (in order of size): `\Huge`, `\huge`, `\LARGE`, `\Large`, `\large`, `\small`, `\footnotesize`, `\tiny`. So for example:

```
{\large this text is slightly bigger than normal}, this one is not
```

> `\normalsize` is used to create text at the default size for the document.

New lines

> you can force the start of a new line by `\\`. If you want to add more vertical space you don't need to use several new-line commands, just insert some vertical space. For example, this way `\\[1cm]` you start a new line after having left 1 cm of empty space.

Date

> you can insert the date of the current day with the command `\today`. If you do not wish to insert any date, keep it blank e.g. `\date{}`

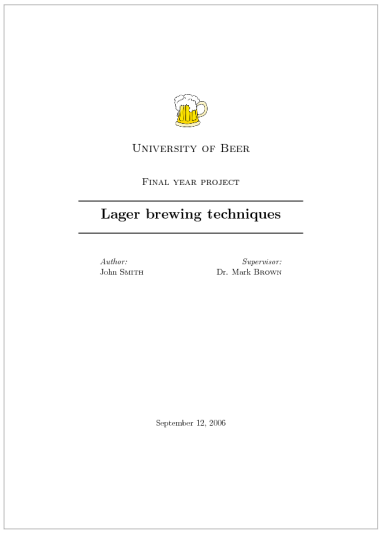
Filling the page

the command `\vfill` keeps on adding empty spaces until the page is full. If you put it in the page, you are sure that all the following text will be placed at the bottom of the page.

A practical example

All these tips might have made you confused. Then, here is a practical example. Get the `test_title.tex` described above and here is an example of a `title.tex`. On the right you can see the output after you compile `test_title.tex` in PDF:

```
\begin{titlepage} \begin{center} % Upper part of the
page
\includegraphics[width=0.15\textwidth]{./logo}\\[1cm]
\textsc{\LARGE University of Beer}\\[1.5cm]
\textsc{\Large Final year project}\\[0.5cm] % Title
\HRule \\[0.4cm] { \huge \bfseries Lager brewing
techniques}\\[0.4cm] \HRule \\[1.5cm] % Author and
supervisor \begin{minipage}{0.4\textwidth}
\begin{flushleft} \large \emph{Author:}\\ John
\textsc{Smith} \end{flushleft} \end{minipage}
\begin{minipage}{0.4\textwidth} \begin{flushright}
\large \emph{Supervisor:} \\ Dr.~Mark \textsc{Brown}
\end{flushright} \end{minipage} \vfill % Bottom of
the page {\large \today} \end{center} \end{titlepage}
```



The picture is from a file called `logo.png` that is in the same directory of both `title.tex` and `test_title.tex`. Since I wanted to insert both the author and supervisor names properly aligned I used a trick: I created two small minipages, one on left and one on the right. Their width is a bit less than half of page width (as you can see, they are exactly 40% of the text width). Within the minipages I have used different alignments. Using `\vfill` I could write the date exactly at the bottom of the page.

As you can see, the code looks "dirtier" than standard LaTeX source because you have to take care of the output as well. If you start changing font's output it will get more confused, but you can do it: it's only for the title and your complicated code will be isolated from all the rest within its own file `title.tex`.

Integrating the title page

Assuming that your title page is now contained in a file named `title.tex`, it must be placed in the same directory as the main document. In order to integrate it, the input command must be used by placing `\input{./title.tex}` at the top of the document. Don't forget to add the commands `\usepackage[pdftex]{graphicx}` and `\newcommand{\HRule}{\rule{\linewidth}{0.5mm}}` in the preamble section as well.

For example, the top section of your document would look like:

```
...                                          \usepackage[pdftex]{graphicx}
\newcommand{\HRule}{\rule{\linewidth}{0.5mm}}          \begin{document}
\input{./title.tex} \tableofcontents ...
```

Additional documentation and packages

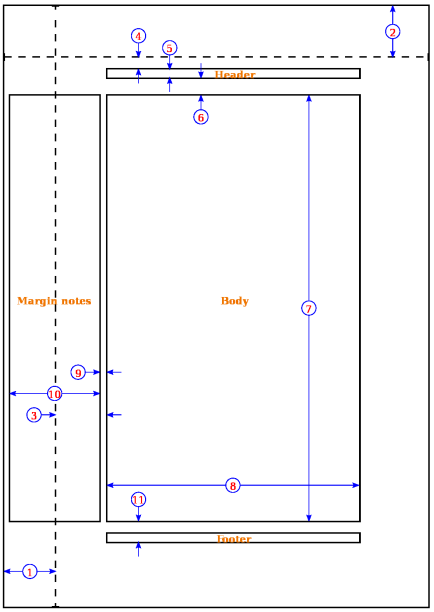
The titlepages package presents many styles of designs for title pages. Italian users may also want to use the frontespizio package.

Page Layout

Latex and the document class will normally take care of page layout issues for you. For submission to an academic publication, this entire topic will be out of your hands, as the publishers want to control the presentation. However, for your own documents, there are some obvious settings that you may wish to change: margins, page orientation and columns, to name but three. The purpose of this tutorial is to show you how to configure your pages.

Page dimensions

A page in Latex is defined by myriad internal parameters. Each parameter corresponds to the length of an element of the page, for example, `\paperheight` is the physical height of the page. Here you can see a diagram showing all the variables defining the page:



- one inch + `\hoffset`
- one inch + `\voffset`
- `\oddsidemargin` = 31pt
- `\topmargin` = 20pt
- `\headheight` = 12pt
- `\headsep` = 25pt
- `\textheight` = 592pt
- `\textwidth` = 390pt
- `\marginparsep` = 10pt
- `\marginparwidth` = 35pt
- `\footskip` = 30pt
- `\marginparpush` = 7pt (not shown)
- `\hoffset` = 0pt
- `\voffset` = 0pt
- `\paperwidth` = 597pt
- `\paperheight` = 845pt

It will not have been immediately obvious - because it doesn't really cause any serious problems - that the default page size for all standard document classes is *US letter*. This is shorter by 18 mm (about 3/4 inch), and slightly wider by 8 mm (about 1/4 inch), compared to A4 (which is the standard in almost all the rest of the world). As I said, it's not a great problem, and most printers will print the page without a hiccup. However, it is possible to specify alternative sizes.

```
\documentclass[a4paper]{article}
```

The above example illustrates how to pass the optional argument to the `\documentclass`, which will then modify the page dimensions accordingly. The standard document classes that are a part of Latex are built to be fairly generic, which is why you have the flexibility of specifying the page size. Other classes may have different options (or none at all). Normally, 3rd party classes come with some documentation to let you know.

Readers from a word processing background are probably thinking why there is so much white space surrounding the text. There is a good reason, and it's all down to readability. Have a look in a few books, and pick a few lines at random. Count the number of characters per line. I bet the average is about 66. Studies have shown that it's easier to read text when there are 60-70 characters per line - and it would seem that 66 is the optimal number. Therefore, the page margins are set to ensure that readability remains as good as possible. Also, white space is often left in the inner margin for the assumption that the document will be bound.

If you wish to change the margins of your document, there are many ways to do so:

- Simply use the `fullpage` package for somewhat standardized smaller margins (around an inch):

```
\usepackage{fullpage}
```

For an even greater effect give it the `cm` option (around 1.5cm):

```
\usepackage[cm]{fullpage}
```

- Use the `a4wide` package for a page with A4 document size with smaller margins.
- Use the `geometry` package. This package allows you to specify the 4 margins without needing to remember the particular page dimensions commands. You can enter the measures in centimeters and inches as well. Use `cm` for centimeters and `in` for inches after each value (e.g: 1.0in or 2.54cm). These values are relative to the edge of paper (0in) and go inward it. It may be implemented as follows:

```
\usepackage[top=tlength, bottom=blength, left=llength,
right=rlength]{geometry}
```

- Edit individual page dimension variables described above, using the `\addtolength` and `\setlength` commands. For instance,

```
\oddsidemargin=-1cm
\setlength{\textwidth}{6.5in}
\addtolength{\voffset}{-5pt}
```

Additionally, there are several packages designed to solve the problem of varying pages sizes, which override any defaults setup by the document class. One of the most versatile packages for page layout is the `geometry` package. For instance, to set the page size, add the following to your preamble:

```
\usepackage[a4paper]{geometry}
```

The `geometry` package has many pre-defined page sizes, like `a4paper`, built in. Others include: `a0paper`, `a1paper`, ..., `a6paper`, `b0paper`, `b1paper`, ..., `b6paper`, `letterpaper`, `legalpaper`, `executivepaper`.

To explicitly change the paper dimensions using the `geometry` package, the paperwidth and paperheight options can be used. For example:

```
\usepackage[margin=1in, paperwidth=5.5in, paperheight=8.5in]{geometry}
```

Top margin above Chapter

The top margin above a chapter can be changed using the `titlesec` package. Example: [1]

```
\usepackage{titlesec}
\titlespacing*{\chapter}{0pt}{-50pt}{20pt}
\titleformat{\chapter}[display]{\normalfont\huge\bfseries}{\chaptertitlename\
 \thechapter}{20pt}{\Huge}
```

The command `\titleformat` must be used when the spacing of a chapter is changed. In case of a section this command can be omitted.

Page size issues

If you intend to get a pdf in the end, there are basically three ways:

```
TeX => PDF
TeX => DVI => PDF
TeX => DVI => PS => PDF
```

Which are in general obtained with

```
pdflatex myfile                # TeX => PDF
latex myfile                   # TeX => DVI
dvipdf myfile                  # DVI => PDF
dvips myfile -o myfile.ps      # DVI => PS
ps2pdf myfile.ps myfile.pdf    # PS => PDF
```

With all the available Ghostscript versions, the safest way to always get the right paper size in the end is to add

```
\documentclass[...,a4paper,...]{...}
\special{papersize=210mm,297mm}
```

to the tex file, `-t a4` after dvips and `-sPAPERSIZE=a4` after the ps2pdf. For pdflatex to work fine, using the package `geometry` usually works.

If you want US Letter instead, replace "210mm,297mm" by "8.5in,11in" and "a4" by "letter".

Page orientation

When you talk about changing page orientation, it usually means changing to landscape mode, since portrait is the default. I shall introduce two slightly different styles of changing orientation.

The first is for when you want all of your document to be in landscape from the very beginning. There are various packages available to achieve this, but the one I prefer is the `geometry` package. All you need to do is call the package, with *landscape* as an option:

```
\usepackage[landscape]{geometry}
```

Although, if you intend to use `geometry` to set your paper size, don't add the `\usepackage` commands twice, simply string all the options together, separating with a comma:

```
\usepackage[a4paper,landscape]{geometry}
```

The second method is for when you are writing a document in portrait, but you have some contents, like a large diagram or table that would be displayed better on a landscape page. However, you still want the consistency of your headers and footers appearing the same place as the other pages.

The `lscape` package is for this very purpose. It supplies a `landscape` environment, and anything inside is basically rotated. No actual page dimensions are changed. This approach is more applicable to books or reports than to typical academic publications. Using `pdflscape` instead of `lscape` when generating a PDF document will make the page appear right side up when viewed: the single page that is in landscape format will be rotated, while the rest will be left in portrait orientation.

Also, to get a table to appear correctly on a landscaped page, one must place the `tabular` environment inside a `table` environment, which is itself inside the `landscape` environment. e.g., it should look like this:

```
\begin{landscape}
\begin{table}
\centering     % optional, probably makes it look better to have it
centered on the page
\begin{tabular}{....}
.......
\end{tabular}
\end{table}
\end{landscape}
```

Page styles

Page styles in Latex terms refers not to page dimensions, but to the running headers and footers of a document. These headers typically contain document titles, chapter or section numbers/names, and page numbers.

Standard page styles

The possibilities of changing the headers in plain Latex are actually quite limited. There are two commands available: `\pagestyle{style}` will apply the specified style to the current and all subsequent pages, and `\thispagestyle{style}` will only affect the current page. The possible styles are:

empty	Both header and footer are clear
plain	Header is clear, but the footer contains the page number in the center.
headings	Footer is blank, header displays information according to document class (e.g., section name) and page number top right.
myheadings	Page number is top right, and it is possible to control the rest of the header.

With myheadings, the commands `\markright` (in the standard document classes, book, report and article) and `\markboth` (only in the book class) are used to control the headings. The following commands placed at the beginning of an article document will set the header of all pages to contain "John Smith" top left, "On page styles" centered and the page number top right:

```
\pagestyle{myheadings}
\markright{John Smith\hfill On page styles\hfill}
```

An issue to look out for is that the major sectioning commands (`\part`, `\chapter` or `\maketitle`) specify a `\thispagestyle{plain}`. So, if you wish to suppress all styles by inserting a `\pagestyle{empty}` at the beginning of your document, then the style command at each section will override your initial rule, for those pages only. To achieve the intended result one can follow the new section commands with `\thispagestyle{empty}`. The `\part` command, however, cannot be fixed this way, because it sets the page style, but also advances to the next page, so that \thispagestyle{} cannot be applied to that page. Another approach is to simply write `\usepackage{nopageno}` in the preamble. This package will make `\pagestyle{plain}` have the same effect as `\pagestyle{empty}`, effectively suppressing page numbering when it is used.

Customising with `fancyhdr`

To get better control over the headers, one can use the package `fancyhdr` written by Piet van Oostrum. It provides several commands that allow you to customize the header and footer lines of your document. For a more complete guide, the author of the package produced this documentation [2].

The tricky problem when customizing headers and footers is to get things like running section and chapter names in there. LaTeX accomplishes this with a two-stage approach. In the header and footer definition, you use the commands `\rightmark` and `\leftmark` to represent the current section and chapter heading, respectively. The values of these two commands are overwritten whenever a chapter or section command is processed. For ultimate flexibility, the `\chapter` command and its friends do not redefine `\rightmark` and `\leftmark` themselves. They call yet another command (`\chaptermark`, `\sectionmark`, or `\subsectionmark`) that is responsible for redefining `\rightmark` and `\leftmark`.

If you want to change the look of the chapter name in the header line, you need only "renew" the `\chaptermark` command.

To begin, add the following lines to your preamble:

```
\usepackage{fancyhdr}
\setlength{\headheight}{15.2pt}
\pagestyle{fancy}
```

The second line will prevent LaTeX from giving a warning. Both the header and footer comprise three elements each according to its horizontal position (left, centre or right). To set their values, the following commands are available:

`\lhead[`*lh-even*`]{`*lh-odd*`}` `\chead[`*ch-even*`]{`*ch-odd*`}` `\rhead[`*rh-even*`]{`*rh-odd*`}`

`\lfoot[`*lf-even*`]{`*lf-odd*`}` `\cfoot[`*cf-even*`]{`*cf-odd*`}` `\rfoot[`*rf-even*`]{`*rf-odd*`}`

Hopefully, the behaviour of the above commands is fairly intuitive: if it has *head* in it, it affects the head etc, and obviously, *l*, *c* and *r* means left, centre and right respectively. Documents can be either one- or two-sided. Articles are by default one-sided, books are two-sided. Two-sided documents differentiate the left (even) and right (odd) pages, whereas one-sided do not.

Watch out: if you give long text in two different "parts" only in the footer or only in the header, you might see overlapping text, be careful. There are special commands you can use as arguments:

`\thepage`	number of the current page
`\leftmark`	current chapter name printed like "CHAPTER 3. THIS IS THE CHAPTER TITLE"
`\rightmark`	current section name printed like "1.6. THIS IS THE SECTION TITLE"
`\chaptername`	the name *chapter* in the current language. If this is English, it will display "Chapter"
`\thechapter`	current chapter number
`\thesection`	current section number

Note that `\leftmark` and `\rightmark` convert the names to uppercase, whichever was the formatting of the text. If you want them to print the actual name of the chapter without converting it to uppercase use the following command:

```
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{#1}{}}
```

now `\leftmark` and `\rightmark` will just print the name of the chapter and section, without number and without affecting the formatting. Note that these redefinitions must be inserted *after* the first call of

`\pagestyle{fancy}`. The standard book formatting of the `\chaptermark` is:

```
\renewcommand{\chaptermark}[1]{\markboth{\MakeUppercase{\chaptername\
\thechapter.\ #1}}{}}
```

Moreover, with the following commands you can define the thickness of the decorative lines on both the header and the footer:

```
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
```

The first line for the header, the second for the footer. Setting it to zero means that there will be no line.

An example:

```
\fancyhf{}

\lhead{Andrew Roberts}
\rhead{\today}
\rfoot{\thepage}
```

It is often necessary to clear any defaults or a previous style definition, and the first line of the above example will do this. The commands are an alternative interface to customising the headers/footers that fancyhdr offers, and so by not passing anything to them, it assumes that you want it all blank.

The result of these commands will put my name at the top left, todays date at the top right, and the current page number at the bottom right of the page. Even if the document class was two-sided, because no optional text has been supplied for the even pages, the style will be used for all pages.

This approach has a serious bad point: some pages like the title or the beginning of each chapter have no header or footer, but with the code we have shown *every* page will get the same output. There is a way to solve this problem: you can use the *fancyplain* style. If you do so, you can use the command `\fancyplain{...}{...}` inside `\lhead{...}` etc.

When LaTeX wants to create a page with an empty style, it will insert the first argument of `fancyplain`, in all the other cases it will use the second argument. So, an improved version of the previous code would be:

```
\pagestyle{fancyplain}

\fancyhf{}

\lhead{\fancyplain{}{Andrew Roberts}}
\rhead{\fancyplain{}{\today}}
\rfoot{\fancyplain{}{\thepage}}
```

It has the same behavior of the previous code, but you will get empty header and footer in the title and at the beginning of chapters.

For two-sided, it's common to mirror the style of opposite pages, you tend to think in terms of *inner* and *outer*. So, the same example as above for two-sided is:

```
\lhead[Andrew Roberts]{}
\rhead[]{Andrew Roberts}
\lhead[]{\today}
\rhead[\today]{}
\lfoot[\thepage]{}
```

```
\rfoot[]{\thepage}
```

This is effectively saying my name is top outer, todays date is top inner, and current page number is bottom outer. You can use the `fancyplain` command within them for two-sided documents, too.

As an example, here is the complete code of a basic configuration you could use for a real document:

```
\usepackage{fancyhdr}
\setlength{\headheight}{15pt}

\pagestyle{fancyplain}
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}

\lhead{\fancyplain{}{\thepage}}
\chead{}
\rhead{\fancyplain{}{\textit{\leftmark}}}
\lfoot{}
\cfoot{}
\rfoot{}
```

*NB. If you want to make the **article class two-sided**, use* **`\documentclass[twoside]{article}`**.

Another approach with `fancyhdr`

If you want to get different style for even and odd pages, there is another possible way, still using `fancyhdr`. Start again with:

```
\fancyhf{}
```

it will just delete the current heading/footer configuration, so you can make your own. Now you can create what you want using the command `\fancyhead` for header and `\fancyfoot` for footer. They work in the same way, so we'll explain only the first one. The syntax is:

```
\fancyhead[selectors]{output you want}
```

The selectors are the following:

E even page

O odd page

L left side

C centered

R right side

so **CE** will refer to the center of the even pages and **RO** to the right side of the odd pages. Whether it is header or footer, depends if you are using `\fancyhead` or `\fancyfoot`. You can use multiple selectors separated by a comma. Here is an example:

```
\fancyhead[CE]{Author's Name}
\fancyhead[CO]{\today}
\fancyfoot[LE,RO]{\thepage}
```

it will print author's name on the center of the header of the even pages, the date of the current day on the center of the odd pages and the current page number on the left side of even pages *and* on the right size of the odd pages.

Finally, in order to have the pages at the beginning of any chapter really plain, you could redefine the *plain* style, for example to have a really plain page when you want. The command to use is `\fancypagestyle{plain}{...}` and the argument can contain all the commands explained before. An example is the following:

```
\fancypagestyle{plain}{ %
\fancyhf{} % remove everything
\renewcommand{\headrulewidth}{0pt} % remove lines as well
\renewcommand{\footrulewidth}{0pt}}
```

Finally, here is the complete code of a possible style you could use for a two-sided document:

```
\usepackage{fancyhdr}
\setlength{\headheight}{15pt}

\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{#1}{}}

\fancyhf{}
\fancyhead[LE,RO]{\thepage}
\fancyhead[RE]{\textit{\nouppercase{\leftmark}}}
\fancyhead[LO]{\textit{\nouppercase{\rightmark}}}

\fancypagestyle{plain}{ %
\fancyhf{} % remove everything
\renewcommand{\headrulewidth}{0pt} % remove lines as well
\renewcommand{\footrulewidth}{0pt}}
```

Page *n* of *m*

Some people like to put the current page number in context with the whole document. LaTeX only provides access to the current page number. However, you can use the `lastpage` package to find the total number of pages, like this:

```
\usepackage{lastpage}
...
\cfoot{\thepage\ of \pageref{LastPage}}
```

Note the capital letters. Also, add a backslash after `\thepage` to ensure adequate space between the page number and 'of'. And recall, when using references, that you have to run LaTeX an extra time to resolve the cross-references.

Multi-column pages

It is common to see articles and conference proceedings formatted with two columns of text. However, such publishers will usually provide you with their own document class, which automatically implements this format, without you having to do anything. It is very easy to format your page in this way. If you are using a standard Latex document class, then you can simply pass the optional argument *twocolumn* to the document class: `\documentclass[twocolumn]{article}` which will give the desired effect.

While this simple addition will do the job 9 out of 10 times, it is widely acknowledged that there are many limitations of this approach, and that the `multicol` package is much more useful for handling multiple columns. It

has several advantages:

- Can support up to ten columns.
- Implements a *multicols* environment, therefore, it is possible to mix the number of columns within a document.
- Additionally, the environment can be nested inside other environments, such as `figure`.
- `Multicol` outputs *balanced* columns, whereby the columns on the final page will be of roughly equal length.
- Vertical rules between columns can be customised.
- Column environments can be easily customised locally or globally.

Floats are not fully supported by this environment. It can only cope if you use the starred forms of the float commands (e.g., `\begin{figure*}`) which makes the float span all columns. This is not hugely problematic, since floats of the same width as a column may be too small, and you would probably want to span them anyway.

To create a typical two-column layout:

```
\begin{multicols}{2}
  lots of text
\end{multicols}
```

The parameter `\columnseprule` holds the width of the vertical rules. By default, the lines are omitted as this parameter is set to a length of 0pt. Do the following before the beginning of the environment:

```
\setlength{\columnseprule}{1pt}
```

This will draw a thin line of 1pt in width. A thick line would not look very pleasing, however, you are free to put in any length of your choosing. Also, to change the horizontal space in between columns (the default is set at 10pt, which is quite narrow) then you need to change the `\columnsep` parameter:

```
\setlength{\columnsep}{20pt}
```

Manual page formatting

There may be instances, especially in very long documents, such as books, that Latex will not get all page breaks looking as good as it could. It may, therefore, be necessary to manually tweak the page formatting. Of course, you should only do this at the very final stage of producing your document, once all the content is complete. Latex offers the following:

`\newline`	Breaks the line at the point of the command.
`\\`	Breaks the line at the point of the command, it's a shorter version of the previous command but it does exactly the same
`\\*`	Breaks the line at the point of the command and additionally prohibits a page break after the forced line break
`\linebreak[number]`	Breaks the line at the point of the command. The *number* you provide as an argument represents the priority of the command in a range from 0 (it will be easily ignored) to 4 (do it anyway). LaTeX will try to produce the best line breaks possible, meeting its high standards. If it cannot, it will decide whether including the linebreak or not according to the priority you have provided.
`\newpage`	Ends the current page and starts a new one.
`\pagebreak[number]`	Breaks the current page at the point of the command. The optional *number* argument sets the priority in a scale from 0 to 4.
`\nopagebreak[number]`	Stops the page being broken at the point of the command. The optional *number* argument sets the priority in a scale from 0 to 4.
`\clearpage`	Ends the current page and causes any floats encountered in the input, but yet to appear, to be printed.

Widows and orphans

In professional books, it's not desirable to have single lines at the beginning or end of a page. In typesetting such situations are called 'widows' and 'orphans'. Normally it is possible that widows and orphans appear in LaTeX documents. You can try to deal with them using manual page formatting, but there's also an automatic solution.

LaTeX has a parameter for 'penalty' for widows and orphans ('club lines' in LaTeX terminology). With the greater penalty LaTeX will try more to avoid widows and orphans. You can try to increase these penalties by putting following commands in your document preamble:

```
\widowpenalty=300
\clubpenalty=300
```

If this does not help, you can try increasing these values even more, to a maximum of 10000. However, it is not recommended to set this value too high, as setting it to 10000 forbids LaTeX from doing this altogether, which might result in strange behavior.

It also helps to have rubber band values for the space between paragraphs:

```
\setlength{\parskip}{3ex plus 2ex minus 2ex}
```

Summary

This tutorial is relatively short, largely due to the fact that the whole LaTeX ethos is to concentrate on the content, and let LaTeX (and/or other typographers who have developed suitable document classes) decide on the best presentation. The next step to achieve greater control of page layout is to set about designing your own class. Unfortunately, that is not a straightforward task, and is often best left to the professionals!

This page uses material from Andy Roberts' Getting to grips with Latex [3] *with permission from the author.*

References

[1] http://www.ctex.org/documents/packages/layout/titlesec.pdf
[2] http://www.ctan.org/tex-archive/macros/latex/contrib/fancyhdr/fancyhdr.pdf
[3] http://www.andy-roberts.net/misc/latex/index.html

Formatting

This section will guide you through the various text, paragraph, and page formatting techniques. *Formatting* tends to refer to most things to do with appearance, so it makes the list of possible topics quite eclectic: text style, font, size; paragraph alignment, interline spacing, indents; special paragraph types; footnotes, margin notes, etc.

A lot of the formatting techniques are required to differentiate certain elements from the rest of the text. It is often necessary to add emphasis to key words or phrases. Footnotes are useful for providing extra information or clarification without interrupting the main flow of text. So, for these reasons, formatting is very important. However, it is also very easy to abuse, and a document that has been over-done can look and read worse than one with none at all.

Text formatting

Hyphenation

LaTeX hyphenates words whenever necessary. If the hyphenation algorithm does not find the correct hyphenation points, you can remedy the situation by using the following commands to tell TeX about the exception. The command

```
\hyphenation{word list}
```

causes the words listed in the argument to be hyphenated only at the points marked by "-". The argument of the command should only contain words built from normal letters, or rather characters that are considered to be normal letters by LaTeX. It is known that the hyphenation algorithm does not find all correct American English hyphenation points for several words. A log of known exceptions is published periodically in the *TUGboat* journal. (See a 2008 list: http://www.tug.org/TUGboat/Articles/tb29-2/tb92hyf.pdf)

The hyphenation hints are stored for the language that is active when the hyphenation command occurs. This means that if you place a hyphenation command into the preamble of your document it will influence the English language hyphenation. If you place the command after the `\begin{document }` and you are using some package for national language support like babel, then the hyphenation hints will be active in the language activated through babel. The example below will allow "hyphenation" to be hyphenated as well as "Hyphenation", and it prevents "FORTRAN", "Fortran" and "fortran" from being hyphenated at all. No special characters or symbols are allowed in the argument. Example:

```
\hyphenation{FORTRAN Hy-phen-a-tion}
```

The command `\-` inserts a discretionary hyphen into a word. This also becomes the only point where hyphenation is allowed in this word. This command is especially useful for words containing special characters (e.g., accented characters), because LaTeX does not automatically hyphenate words containing special characters.

```
\begin{minipage}{2in} I think this is: su\-per\-cal\-%
i\-frag\-i\-lis\-tic\-ex\-pi\-% al\-i\-do\-cious
\end{minipage}
```

I think this is: supercalifragi-
listicexpialidocious

This can be quite cumbersome if one has many words that contain a dash like electromagnetic-endioscopy. One alternative to this is using the `\hyp` command of the hyphenat package. This command typesets a hyphen and allows full automatic hyphenation of the other words forming the compound word. One would thus write

```
electromagnetic\hyp{}endioscopy
```

Several words can be kept together on one line with the command

```
\mbox{text}
```

It causes its argument to be kept together under all circumstances. Example:

```
My phone number will change soon. It will be \mbox{0116 291 2319}.
```

`\fbox` is similar to `\mbox` , but in addition there will be a visible box drawn around the content.

To avoid hyphenation altogether, the penalty for hyphenation can be set to an extreme value:

```
\hyphenpenalty=100000
```

You can change the degree to which LaTeX will hyphenate by changing the value of `\tolerance=1000` and `\hyphenpenalty=1000` . You'll have to experiment with the values to achieve the desired effect. A document which has a low tolerance value will cause LaTeX not to tolerate uneven spacing between words, hyphenating words more frequently than in documents with higher tolerances.

Quote-marks

Latex treats left and right quotes as different entities. For single quotes, ` (on American keyboards, this symbol is found on the tilde key (adjacent to the number 1 key on most) gives a left quote mark, and ' is the right. For double quotes, simply double the symbols, and Latex will interpret them accordingly. (Although, you can use the " for right double quotes if you wish). On British keyboards, ' ` ' is left of the ' 1 ' key and shares the key with ' ¬ ', and sometimes '¦' or '|'. The apostrophe (' ' ') key is to the right of the colon/semicolon key and shares it with the ' @ ' symbol.

`To `quote' in Latex`	To ‘quote’ in Latex.
`To ``quote'' in Latex`	To “quote” in Latex.
`To ``quote" in Latex`	To “quote” in Latex.
`To ,,quote'' in Latex`	To „quote” in Latex.
``Please press the `x' key.''	“Please press the ‘x’ key.”
,,Proszę, naciśnij klawisz <<x>>''.	„Proszę, naciśnij klawisz «x».”

The right quote is also used for apostrophe in Latex without trouble.

For left bottom quote and European quoting style you need to use `T1` font encoding enabled by:

```
\usepackage[T1]{fontenc}
```

The package csquotes offers a multi-lingual solution to quotations, with integration to citation mechanisms offered by BibTeX. This package allows one for example to switch languages and quotation styles according to babel language selections.

Diacritics and accents

Diacritics may be added to letters by placing special escaped metacharacters before the letter that requires the diacritic. For a list of diacritic metacharacters, see LaTeX/Accents.

Space between Words and Sentences

To get a straight right margin in the output, LaTeX inserts varying amounts of space between the words. By default, It also inserts slightly more space at the end of a sentence. However, the extra space added at the end of sentences is generally considered typographically old-fashioned in English language printing. (The practice is found in nineteenth century design and in twentieth century typewriter styles.) Most modern typesetters treat the end of sentence space the same as the interword space. (See for example, Bringhurst's *Elements of Typographic Style*.) The additional space after periods can be disabled with the command

```
\frenchspacing
```

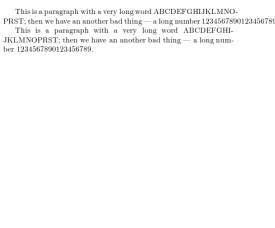
which tells LaTeX not to insert more space after a period than after ordinary character. Frenchspacing can be turned off later in your document via the `\nonfrenchspacing` command.

If an author wishes to use the wider end-of-sentence spacing, care must be exercised so that punctuation marks are not misinterpreted as ends of sentences. TeX assumes that sentences end with periods, question marks or exclamation marks. Although if a period follows an uppercase letter, this is not taken as a sentence ending, since periods after uppercase letters normally occur in abbreviations. Any exception from these assumptions has to be specified by the author. A backslash in front of a space generates a space that will not be enlarged. A tilde '~ ' character generates a space that cannot be enlarged and additionally prohibits a line break. The command `\@` in front of a period specifies that this period terminates a sentence even when it follows an uppercase letter. (If you are using `\frenchspacing` , then none of these exceptions need be specified.)

Margin misalignment and interword spacing

Some very long words, numbers or URLs may not be hyphenated properly and move far beyond the side margin. One solution for this problem is to use `sloppypar` environment, which tells LaTeX to adjust word spacing less strictly. As a result, some spaces between words may be a bit too large, but long words will be placed properly.

```
This is a paragraph with a very long word
ABCDEFGHIJKLMNOPRST; then we have another bad thing
--- a long number 1234567890123456789.
\begin{sloppypar} This is a paragraph with a very
long word ABCDEFGHIJKLMNOPRST; then we have an
another bad thing --- a long number
1234567890123456789. \end{sloppypar}
```


This is a paragraph with a very long word ABCDEFGHIJKLMNOPRST; then we have another bad thing — a long number 1234567890123456789. This is a paragraph with a very long word ABCDEFGHIJKLMNOPRST; then we have an another bad thing — a long number 1234567890123456789.

Ligatures

Some letter combinations are typeset not just by setting the different letters one after the other, but by actually using special symbols (like "ff"), called ligatures. Ligatures can be prohibited by inserting `{}` or, if this does not work, `{\kern0pt}` between the two letters in question. This might be necessary with words built from two words. Here is an example:

```
\Large Not shelfful\\ but shelf{}ful
```

Not shelfful
but shelfful

Some tools are unable to perform search in documents that contain ligatures (a search for `"finally"` wouldn't find the string `"finally"`). If one desires, for greater accessibility, to disable ligatures altogether in the whole document, the `\DisableLigatures` from the microtype package [1] can be used:

```
\usepackage{microtype}
\DisableLigatures{encoding = *, family = *}
```

Note that this will also disable ligatures such as `--` → –, `---` → —, etc.

If you are using XeLaTeX and OpenType fonts, the fontspec package allows for standard ligatures to be turned off as well as fancy swash ligatures to be turned on.

Slash marks

The normal typesetting of the `/` character in LaTeX does not allow following characters to be "broken" on to new lines, which often create "overfull" errors in output (where letters push off the margin). Words that use slash marks, such as "input/output" should be typeset as "`input\slash output`", which allow the line to "break" after the slash mark (if needed). The use of the `/` character in LaTeX should be restricted to units, such as "`mm/year`", which should not be broken over multiple lines.

Emphasizing Text

In order to add some emphasis to a word or phrase, the simplest way is to use the `\emph{text}` command.

```
I want to \emph{emphasize} a word.
```

I want to *emphasize* a word.

Fonts

See also: LaTeX/Fonts.

In LaTeX, there are many ways to specify and control font, and this section is only intended to serve as a brief overview of the topic.

Font Styles

There are three main font families: roman (e.g., Times), sans serif (e.g., Arial) and monospace (e.g., Courier). You can also specify styles such as italic and bold.

The following table lists the commands you will need to access the typical font styles:

LaTeX command	Equivalent to	Output style	Remarks
`\textnormal{...}`	`{\normalfont ...}`	document font family	this is the default or normal font
`\emph{...}`	`{\em ...}`	*emphasis*	typically italics
`\textrm{...}`	`{\rmfamily ...}`	roman font family	
`\textsf{...}`	`{\sffamily ...}`	sans serif font family	
`\texttt{...}`	`{\ttfamily ...}`	teletypefont family	this is a fixed-width or monospace font
`\textup{...}`	`{\upshape ...}`	upright shape	the same as the normal typeface
`\textit{...}`	`{\itshape ...}`	*italic shape*	
`\textsl{...}`	`{\slshape ...}`	*slanted shape*	a skewed version of the normal typeface (similar to, but slightly different from, italics)
`\textsc{...}`	`{\scshape ...}`	Small Capitals	

`\uppercase{...}`		uppercase (all caps)	Also `\lowercase`. There are some caveats, though; see here [2].
`\textbf{...}`	`{\bfseries ...}`	**bold**	
`\textmd{...}`	`{\mdseries ...}`	medium weight	a font weight in between normal and bold

The commands in column two are not entirely equivalent to the commands in column one: They do not correct spacing after the selected font style has ended. The commands in column one are therefore in general recommended.

You may have noticed the absence of underline. Although this is available via the `\underline{...}` command, text underlined in this way will not break properly. This functionality has to be added with the `ulem` (underline emphasis) package. Stick `\usepackage{ulem}` in your preamble. By default, this overrides the `\emph` command with the underline rather than the italic style. It is unlikely that you wish this to be the desired effect, so it is better to stop `ulem` taking over `\emph` and simply call the underline command as and when it is needed.

- To restore the usual `em` formatting, add `\normalem` straight after the document environment begins. Alternatively, use `\usepackage[normalem]{ulem}` .
- To underline, use `\uline{...}` .
- To add a wavy underline, use `\uwave{...}` .
- And for a strike-out `\sout{...}` .

Sizing text

To apply different font sizes, simply follow the commands on this table:

Command	Output
`\tiny`	sample text
`\scriptsize`	sample text
`\footnotesize`	sample text
`\small`	sample text
`\normalsize`	sample text
`\large`	sample text
`\Large`	sample text
`\LARGE`	sample text
`\huge`	sample text
`\Huge`	sample text

Note that the font size definitions are set by the document class. Depending on the document style the actual font size may differ from that listed above. And not every document class has unique sizes for all 10 size commands.

Absolute Point Sizes, [10pt] being default

size	standard classes, *proc*			AMS classes, *memoir*			*slides*	*beamer*		
	[10pt]	[11pt]	[12pt]	[10pt]	[11pt]	[12pt]		[10pt]	[11pt]	[12pt]
`\tiny`	6.80565	7.33325	7.33325	7.33325	7.97224	8.50012	17.27505	5.31258	6.37509	6.37509
`\scriptsize`	7.97224	8.50012	8.50012	7.97224	8.50012	9.24994	20.73755	7.43760	8.50012	8.50012
`\footnotesize`	8.50012	9.24994	10.00002	8.50012	9.24994	10.00002	20.73755	8.50012	9.24994	10.00002
`\small`	9.24994	10.00002	10.95003	9.24994	10.00002	10.95003	20.73755	9.24994	10.00002	10.95003
`\normalsize`	10.00002	10.95003	11.74988	10.00002	10.95003	11.74988	24.88382	10.00002	10.95003	11.74988
`\large`	11.74988	11.74988	14.09984	10.95003	11.74988	14.09984	29.86258	11.74988	11.74988	14.09984

\Large	14.09984	14.09984	15.84985	11.74988	14.09984	15.84985	35.82510	14.09984	14.09984	16.24988
\LARGE	15.84985	15.84985	19.02350	14.09984	15.84985	19.02350	43.00012	16.24988	16.24988	19.50362
\huge	19.02350	19.02350	22.82086	15.84985	19.02350	22.82086	51.60014	19.50362	19.50362	23.39682
\Huge	22.82086	22.82086	22.82086	19.02350	22.82086	22.82086	51.60014	23.39682	23.39682	23.39682

As a technical note, points in TeX follow the standard American point size in which 1 pt is approximately 0.3513_6 mm. The standard point size used in most modern computer programs (known as the *desktop publishing point* or *PostScript point*) has 1 pt equal to approximately 0.352_7 mm while the standard European point size (known as the *Didot point*) had 1 pt equal to approximately 0.37597151 mm. (See: w:Point_(typography).)

Even if you can easily change the output of your fonts using those commands, you're better off not using explicit commands like this, because they work in opposition to the basic idea of LaTeX, which is to separate the logical and visual markup of your document. This means that if you use the same font changing command in several places in order to typeset a special kind of information, you should use **\newcommand** to define a "logical wrapper command" for the font changing command.

```
\newcommand{\oops}[1]{\textbf{#1}} Do not \oops{enter} this
room, it's occupied by \oops{machines} of unknown origin and
purpose.
```

Do not **enter** this room, it's occupied by **machines** of unknown origin and purpose.

This approach has the advantage that you can decide at some later stage that you want to use some visual representation of danger other than **\textbf** , without having to wade through your document, identifying all the occurrences of **\textbf** and then figuring out for each one whether it was used for pointing out danger or for some other reason.

Text mode superscript and subscript

To superscript text in text-mode, you can use the **\textsuperscript{}** command. This allows you to, for instance, typeset 6th as 6th:

```
Michelangelo was born on March 6\textsuperscript{th}, 1475.
```

A very common use of subscripts within the text environment is to typeset chemical formulae. For this purposes, a highly recommended package is mhchem [3]. This package is easy to use, and works with your text fonts (rather than math fonts). To insert a chemical formula, use **\ce{}** with the text-equivalent formula, for example:

```
% In your preamble, add:
\usepackage[version=3]{mhchem} ... % In your
document: Ammonium sulphate is \ce{(NH4)2SO4}.
```

Ammonium sulphate is $(NH_4)_2SO_4$.

Subscripting in text-mode is not supported by LaTeX alone; however, several packages allow the use of the **\textsubscript{}** command. For instance, bpchem [4], KOMA-Script2 [5], and fixltx2e [6] all support this command. Of these, fixltx2e [6] is perhaps the most universal option since it is distributed with LaTeX and requires no additional packages to be implemented.

```
% In your preamble, add:
\usepackage{fixltx2e} ... % In your
document: It is found that
height\textsubscript{apple tree} is
different than
height\textsubscript{orange tree}.
```

It is found that $\text{height}_{\text{apple tree}}$ is different than $\text{height}_{\text{orange tree}}$.

If you do not load a package that supports `\textsubscript{}` , the math mode must be used. This is easily accomplished in running text by bracketing your text with the `$` symbol. In math mode subscripting is done using the underscore: `_{}` .

For example, the formula for water is written as:

```
H$_2$O is the formula for water
```

H_2O is the formula for water

See also the above mentioned package mhchem for chemical symbols and formulas.

Note that in math mode text will appear in a font suitable for mathematical variables. In math mode, to generate roman text, for example, one would use the `\mathrm` command:

```
This is $\mathrm{normal\ roman\
and}_\mathrm{subscript\ roman}$ text
```

This is normal roman and$_{\mathrm{subscript\ roman}}$ text

Note the use of \<space> to insert a space in math mode.

Similarly, you can superscript using:

```
This is $\mathrm{normal\ roman\
and}^\mathrm{superscript\ roman}$ text
```

This is normal roman and$^{\mathrm{superscript\ roman}}$ text

Text figures ("old style" numerals)

Many typographers prefer to use titling figures, sometimes called lining figures, when numerals are interspersed with full caps, when they appear in tables, and when they appear in equations, using text figures elsewhere. LaTeX allows this usage through the `\oldstylenums{}` command:

```
\oldstylenums{1234567890}
```

Some fonts do not have text figures built in; the `textcomp` package attempts to remedy this by effectively generating text figures from the currently-selected font. Put `\usepackage{textcomp}` in your preamble. `textcomp` also allows you to use decimal points, properly formatted dollar signs, etc. within `\oldstylenums{}` .

One common use for text figures is in section, paragraph, and page numbers. These can be set to use text figures by placing some code in your preamble:

```
\usepackage{textcomp}

% Enclose everything in an \AtBeginDocument{}
\AtBeginDocument{%
  % Make \section{} use text figures
  \let\myTheSection\thesection
  \renewcommand{\thesection}{\oldstylenums{\myTheSection}}

  % Make \paragraph{} use text figures
  \let\myTheParagraph\theparagraph
  \renewcommand{\theparagraph}{\oldstylenums{\myTheParagraph}}

  % Make the page numbers in text figures
  \let\myThePage\thepage
  \renewcommand{\thepage}{\oldstylenums{\myThePage}}
}
```

Should you use additional sectioning or paragraphing commands, you may adapt the previous code listing to include them as well.

NOTE: A subsequent use of the `\pagenumbering` command, e.g., `\pagenumbering{arabic}`, will reset the `\thepage` command back to the original. Thus, if you use the `\pagenumbering` command in your document, be sure to reinstate your `\myThePage` definition from the code above:

```
...
\tableofcontents
\pagenumbering{roman}
\chapter{Preface}
...
\chapter{Introduction}
...
\pagenumbering{arabic}
\renewcommand{\thepage}{\oldstylenums{\myThePage}} % without this, the
\thepage command will not be in oldstyle (e.g., in your Table of
Contents}
\Chapter{Foo}
...
```

Symbols and special characters

Dashes and Hyphens

LaTeX knows four kinds of dashes: a hyphen (-), en dash (–), em dash (—), or a minus sign (−). You can access three of them with different numbers of consecutive dashes. The fourth sign is actually not a dash at all—it is the mathematical minus sign:

```
Hyphen: daughter-in-law, X-rated\\ En dash: pages
13--67\\ Em dash: yes---or no? \\ Minus sign: $0$, $1$
and $-1$
```

daughter-in-law, X-rated
pages 13–67
yes—or no?
0, 1 and −1

The names for these dashes are: '-'(-) hyphen , '--'(–) en-dash , '---'(—) em-dash and ' — '(–) minus sign. They have different purposes:

Input	Output	Purpose
-	-	inter-word
--	–	page range, 1–10
---	—	punctuation dash — like this

Use `\hyp{}` macro from `hyphenat` package instead of hyphen if you want LaTeX to break compound words between lines.

Euro € currency symbol

When writing about money these days, you need the euro sign. You have several choices. If the fonts you are using have a euro symbol and you want to use that one, first you have to load the *textcomp* package in the preamble:`\usepackage{textcomp}` then you can insert the euro symbol with the command `\texteuro`. If you want to use the official version of the euro symbol, then you have to use `eurosym`, load it with the `official` option in the preamble: `\usepackage[official]{eurosym}` then you can insert it with the `\euro` command. Finally, if you want a euro symbol that matches with the current font style (e.g., bold, italics, etc.) but your current font does not provide it, you can use the `eurosym` package again but with a different option:

`\usepackage[gen]{eurosym}` again you can insert the euro symbol with `\euro`

Ellipsis (…)

A sequence of three dots is known as an *ellipsis*, which is commonly used to indicate omitted text. On a typewriter, a comma or a period takes the same amount of space as any other letter. In book printing, these characters occupy only a little space and are set very close to the preceding letter. Therefore, you cannot enter 'ellipsis' by just typing three dots, as the spacing would be wrong. Instead, there is a special command for these dots. It is called `\ldots` :

```
Not like this ... but like this:\\ New York, Tokyo,
Budapest, \ldots
```

Not like this ... but like this:
New York, Tokyo, Budapest, . . .

Alternatively, you can use the `\textellipsis` command which allows the spacing between the dots to vary.

Ready-made strings

There are some very simple LaTeX commands for typesetting special text strings:

Command	Example	Description
`\today`	May 31, 2006	Current date
`\TeX`	TEX	Your favorite typesetter
`\LaTeX`	LATEX	The Name of the Game
`\LaTeXe`	LATEX 2ε	The current incarnation

Other symbols

LaTeX has *lots* of symbols at its disposal. The majority of them are within the mathematical domain, and later chapters will cover how to get access to them. For the more common text symbols, use the following commands:

Command	Symbol	Command	Symbol
`\%`	%	`\#`	#
`\$`	$	`\&`	&
`\{`	{	`\}`	}
`\_`	_	`\S`	§
`\P`	¶	`\dag`	†
`\ddag`	‡	`\textbackslash`	\
`\textbar`	|	`\textless`	<
`\textgreater`	>	`\textemdash`	—
`\textendash`	–	`\textregistered`	®
`\texttrademark`	™	`\textquestiondown`	¿
`\textexclamdown`	¡	`\textcircled{a}`	ⓐ
`\textsuperscript{a}`	a	`\copyright`	©
`\pounds`	£		

Not mentioned in above table, tilde (~) is used in LaTeX code to produce non-breakable space. To get printed tilde sign, either make it verbatim text or write `\~{}` . And a visible space ␣ can be created with `\textvisiblespace` .

Of course, these are rather boring. For some more interesting symbols, the Postscript ZipfDingbats font is available thanks to the `pifont` package. Hopefully you are beginning to notice now that when you want to use a package, you need to add the declaration to your preamble; in this instance: `\usepackage{pifont}` . Next, the command `\ding{number}` , will print the specified symbol. Here is a table of the available symbols:

32		33	✁	34	✂	35	✃	36	✄	37	☎	38	✆	39	✇
40	✈	41	✉	42	☛	43	☞	44	✌	45	✍	46	✎	47	✏
48	✐	49	✑	50	✒	51	✓	52	✔	53	✕	54	✖	55	✗
56	✘	57	✙	58	✚	59	✛	60	✜	61	✝	62	✞	63	✝
64	✠	65	✡	66	✢	67	✣	68	✤	69	✥	70	✦	71	✧
72	★	73	☆	74	✪	75	✫	76	✬	77	✭	78	✮	79	✯
80	✰	81	✱	82	✲	83	✳	84	✴	85	✵	86	✶	87	✷
88	✸	89	✹	90	✺	91	✻	92	✼	93	✽	94	✾	95	✿
96	❀	97	❁	98	❂	99	❃	100	❄	101	❅	102	❆	103	❇
104	❈	105	❉	106	❊	107	❋	108	●	109	❍	110	■	111	❑
112	❒	113	❑	114	❒	115	▲	116	▼	117	◆	118	❖	119	◗
120	|	121	❙	122	❚	123	❛	124	❜	125	❝	126	❞		
		161	❡	162	❢	163	❣	164	❤	165	❥	166	❦	167	❧
168	♣	169	♦	170	♥	171	♠	172	①	173	②	174	③	175	④
176	⑤	177	⑥	178	⑦	179	⑧	180	⑨	181	⑩	182	❶	183	❷
184	❸	185	❹	186	❺	187	❻	188	❼	189	❽	190	❾	191	❿
192	①	193	②	194	③	195	④	196	⑤	197	⑥	198	⑦	199	⑧
200	⑨	201	⑩	202	❶	203	❷	204	❸	205	❹	206	❺	207	❻
208	❼	209	❽	210	❾	211	❿	212	➔	213	→	214	↔	215	↕
216	↘	217	↣	218	↗	219	↠	220	➜	221	→	222	⟶	223	➞
224	➠	225	➡	226	➢	227	➢	228	➤	229	➥	230	➦	231	➧
232	➨	233	➩	234	➪	235	➫	236	➬	237	➭	238	➮	239	➯
		241	➱	242	➲	243	➳	244	➴	245	➵	246	➶	247	➷
248	➸	249	➹	250	➺	251	➻	252	➼	253	➽	254	➾		

.

Paragraph Formatting

Altering the paragraph formatting is not often required, especially in academic writing. However, it is useful to know, and applications tend to be for formatting text in floats, or other more exotic documents.

Paragraph Alignment

Paragraphs in Latex are usually fully justified (i.e., flush with both the left and right margins). For whatever reason, should you wish to alter the justification of a paragraph, there are three environments at hand, and also Latex command equivalents.

Alignment	Environment	Command
Left justified	`flushleft`	`\raggedright`
Right justified	`flushright`	`\raggedleft`
Center	`center`	`\centering`

All text between the **`\begin`** and **`\end`** of the specified environment will be justified appropriately. The commands listed are for use within other environments. For example, `p` (paragraph) columns in `tabular` .

Paragraph Indents

By default, the first paragraph after a heading follows the standard Anglo-American publishers' practice of no indentation. The size of subsequent paragraph indents are determined by a parameter called `\parindent` . The default length that this constant holds is set by the document class that you use. It is possible to override using the `\setlength` command.

```
\setlength{\parindent}{1cm}
```

This will set paragraph indents to 1cm.

Be careful, however, if you decide to set the indent to zero, then it means you will need a vertical space between paragraphs in order to make them clear. The space between paragraphs is held in `\parskip` , which could be altered in a similar fashion as above. However, this parameter is used elsewhere too, such as in lists, which means you run the risk of making various parts of your document look very untidy by changing this setting. If you want to use the style of having no indentation with a space between paragraphs, use the `parskip` package, which does this for you, while making adjustments to the spacing of lists and other structures which use paragraph spacing, so they don't get too far apart. Add this to the preamble:

```
\usepackage{parskip}
```

To indent subsequent lines of a paragraph, use the TeX command `\hangindent` . (While the default behaviour is to apply the hanging indent after the first line, this may be changed with the `\hangafter` command.) An example follows.

```
\hangindent=0.7cm This paragraph has an extra indentation at the left.
```

The TeX commands `\leftskip` and `\rightskip` add additional space to the left and right sides of each line, allowing the formatting for subsequent paragraphs to differ from the overall document margins. This space is in addition to the indentation added by `\parindent` and `\hangindent` .

To change the indentation of the last line in a paragraph, use the TeX command `\parfillskip` .

White-space in LaTeX can also be made flexible (what Lamport calls "rubber" lengths). This means that values such as `\parskip` can have a default dimension plus an amount of expansion minus an amount of contraction. This is useful on pages in complex documents where not every page may be an exact number of fixed-height lines long, so some give-and-take in vertical space is useful. You specify this in a `\setlength` command like this:

```
\setlength{\parskip}{1cm plus4mm minus3mm}
```

Line Spacing

To change line spacing in the whole document use the command `\linespread` covered in LaTeX/Customizing_LaTeX#Spacing.

To change line spacing in specific environments do the following:

1. Add `\usepackage{setspace}` to the document preamble.
2. This then provides the following environments to use within your document:
   - `doublespace` - all lines are double spaced.
   - `onehalfspace` - line spacing set to one-and-half spacing.
   - `singlespace` - normal line spacing.

After declaring the package in the preamble the use of the command `\singlespacing` , `\doublespacing` , or `\onehalfspacing` will specify the line spacing for all sections and paragraphs until another command is used.

See the section on customizing lists below for information on how to change the line spacing in lists.

Special Paragraphs

`Verbatim` Text

There are several ways to introduce text that won't be interpreted by the compiler. If you use the `verbatim` environment, everything input between the *begin* and *end* commands are processed as if by a typewriter. All spaces and new lines are reproduced as given, and the text is displayed in an appropriate fixed-width font. Any LaTeX command will be ignored and handled as plain text. This is ideal for typesetting program source code. Here is an example:

```
\begin{verbatim} The verbatim environment simply
reproduces every character you input, including all s p
a c e s! \end{verbatim}
```

```
The verbatim environment
  simply reproduces every
 character you input,
including all  s p a c e s!
```

Note: once in the `verbatim` environment, the only command that will be recognized is `\end{verbatim}`. Any others will be output. The font size in the verbatim environment can be adjusted by placing a font size command before `\begin{verbatim}`. If this is an issue, then you can use the `alltt` package instead, providing an environment with the same name:

```
\begin{alltt} Verbatim extended with the ability to
use normal commands. Therefore, it is possible to
\emph{emphasize} words in this environment, for
example. \end{alltt}
```

```
Verbatim extended with the ability
to use normal commands.  Therefore, it
is possible to emphasize words in
this environment, for example.
```

Remember to add `\usepackage{alltt}` to your preamble to use it though! Within the `alltt` environment, you can use the command `\normalfont` to get back the normal font. To write equations within the `alltt` enviroment, you can use `\(` and `\)` to enclose them, instead of the usual`$`.

When using `\textbf{}` inside the `alltt` enviroment, note that the standard font has no bold TT font. Txtfonts has bold fonts: just add `\renewcommand{\ttdefault}{txtt}` after `\usepackage{alltt}`.

If you just want to introduce a short verbatim phrase, you don't need to use the whole environment, but you have the `\verb` command:

```
\verb+my text+
```

The first character following `\verb` is the delimiter: here we have used "+", but you can use any character you like but * and space; `\verb` will print verbatim all the text after it until it finds the next delimiter. For example, the code:

```
\verb|\textbf{Hi mate!}|
```

will print `\textbf{Hi mate!}`, ignoring the effect `\textbf` should have on text.

For more control over formatting, however, you can try the `fancyvrb` package, which provides a `Verbatim` environment (note the capital letter) which lets you draw a rule round the verbatim text, change the font size, and even have typographic effects inside the `Verbatim` environment. It can also be used in conjunction with the `fancybox` package and it can add reference line numbers (useful for chunks of data or programming), and it can even include entire external files.

Typesetting URLs

One of either the `hyperref` or `url` packages provides the `\url` command, which properly typesets URLs, for example:

```
Go to \url{http://www.uni.edu/~myname/best-website-ever.html} for my
website.
```

will show this URL exactly as typed (similar to the `\verb` command), but the `\url` command also performs a hyphenless break at punctuation characters (only in pdflatex, not in plain latex + dvips). It was designed for Web URLs, so it understands their syntax and will never break mid-way through an unpunctuated word, only at slashes and full stops. Bear in mind, however, that spaces are forbidden in URLs, so using spaces in `\url` arguments will fail, as will using other non-URL-valid characters.

When using this command through the `hyperref` package, the URL is "clickable" in the PDF document, whereas it is not linked to the web when using only the `url` package. Also when using the `hyperref` package, to remove the border placed around a URL, insert `pdfborder = {0 0 0 0}` inside the `\hypersetup{}` .

`Listing` Environment

This is also an extension of the verbatim environment provided by the `moreverb` package. The extra functionality it provides is that it can add line numbers along side the text. The command: `\begin{listing}[step]{first line}` . The mandatory *first line* argument is for specifying which line the numbering shall commence. The optional *step* is the step between numbered lines (the default is 1, which means every line will be numbered).

To use this environment, remember to add `\usepackage{moreverb}` to the document preamble.

Multi-line comments

As we have seen, the only way LaTeX allows you to add comments is by using the special character `%` , that will comment out all the rest of the line after itself. This approach is really time-consuming if you want to insert long comments or just comment out a part of your document that you want to improve later. Using the verbatim package, to be loaded in the preamble as usual:

```
\usepackage{verbatim}
```

(you can also use the comment package instead) you can use an environment called comment that will comment out everything within itself. Here is an example:

```
This is another \begin{comment} rather stupid, but helpful
\end{comment} example for embedding comments in your
document.
```

This is another example for embedding comments in your document.

Note that this won't work inside complex environments, like math for example. You may be wondering, why should I load a package called verbatim to have the possibility to add comments? The answer is straightforward: commented text is interpreted by the compiler just like verbatim text, the only difference is that verbatim text is introduced within the document, while the comment is just dropped.

Alternatively, you can define a `\comment{}` command, by adding the following to the document's preamble:

```
\newcommand{\comment}[1]{}
```

Then, to comment out text, simply do something like this:

```
\comment{ This is a long comment and can extend over multiple lines, etc. }
```

Quoting text

LaTeX provides several environments for quoting text; they have small differences and they are aimed for different types of quotations. All of them are indented on either margin, and you will need to add your own quotation marks if you want them. The provided environments are:

`quote`

for a short quotation, or a series of small quotes, separated by blank lines.

`quotation`

for use with longer quotations, of more than one paragraph, because it indents the first line of each paragraph.

`verse`

is for quotations where line breaks are important, such as poetry. Once in, new stanzas are created with a blank line, and new lines within a stanza are indicated using the newline command, `\\` . If a line takes up more than one line on the page, then all subsequent lines are indented until explicitly separated with `\\` .

Abstracts

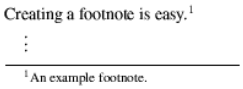
In scientific publications it is customary to start with an abstract which gives the reader a quick overview of what to expect. LaTeX provides the `abstract` environment for this purpose. It is available in `article` and `report` document classes; it's not available in the `book` , but it's quite simple to create your own if you really need it.

Footnotes

Footnotes are a very useful way of providing extra information to the reader. Usually, it is non-essential information which can be placed at the bottom of the page. This keeps the main body of text concise.

The footnote facility is easy to use. The command you need is: `\footnote{text}` . Do not leave a space between the command and the word where you wish the footnote marker to appear, otherwise Latex will process that space and will leave the output not looking as intended.

```
Creating a footnote is easy.\footnote{An example
footnote.}
```



Latex will obviously take care of typesetting the footnote at the bottom of the page. Each footnote is numbered sequentially - a process that, as you should have guessed by now, is automatically done for you.

It is possible to customize the footnote marking. By default, they are numbered sequentially (Arabic). However, without going too much into the mechanics of Latex at this point, it is possible to change this using the following command (which needs to be placed at the beginning of the document, or at least before the first footnote command is issued).

`\renewcommand{\thefootnote}{\arabic{footnote}}`	Arabic numerals, e.g., 1, 2, 3...
`\renewcommand{\thefootnote}{\roman{footnote}}`	Roman numerals (lowercase), e.g., i, ii, iii...
`\renewcommand{\thefootnote}{\Roman{footnote}}`	Roman numerals (uppercase), e.g., I, II, III...
`\renewcommand{\thefootnote}{\alph{footnote}}`	Alphabetic (lowercase), e.g., a, b, c...
`\renewcommand{\thefootnote}{\Alph{footnote}}`	Alphabetic (uppercase), e.g., A, B, C...
`\renewcommand{\thefootnote}{\fnsymbol{footnote}}`	A sequence of nine symbols (try it and see!)

To make a footnote without number mark use this declaration:

```
\let\thefootnote\relax\footnote{There is no number in this footnote}
```

The package footmisc [7] offers many possibilities for customizing the appearance of footnotes. It can be used, for example, to use a different font within footnotes.

Common problems and workarounds

- Footnotes unfortunately don't work with tables, as it is considered a bad practice. You can overcome this limitation with several techniques: you can use `\footnotemark[123]` in the table, and `\footnotetext[123]{HelloWorld!}` somewhere on the page. Or, you can add `\usepackage{footnote}` and `\makesavenoteenv{tabular}` to the preamble, and put your *table* environment in a `\begin{savenotes}` environment. Note that the latter does not work with the packages *color* or *colortbl*. See this FAQ page [8] for other approaches.
- Footnotes also don't work inside minipage environment (In fact, several environments break footnote support. the `\makesavenoteenv{environmentname}` command of the footnote package might fix most). The minipage includes its own footnotes, independent of the document's. The package mpfnmark [9] allows greater flexibility in managing these two sets of footnotes.
- If the text within the footnote is very long, LaTeX may split the footnote over several pages. You can prevent LaTeX from doing so by increasing the penalty for such an operation. To do this, insert the following line into the preamble of your document:

  ```
  \interfootnotelinepenalty=10000
  ```

- To make multiple references to the same footnote, you can use the following syntax:

  ```
  Text that has a footnote\footnote{This is the footnote} looks like this. Later text referring to same footnote\footnotemark[\value{footnote}] uses the other command.
  ```

  If you need hyperref support, use instead:

  ```
  Text that has a footnote\footnote{This is the footnote}\addtocounter{footnote}{-1}\addtocounter{Hfootnote}{-1} looks like this. Later text referring to same footnote\footnotemark uses the other command.
  ```

  Note that these approaches don't work if there are other footnotes between the first reference and any of the other "duplicates".

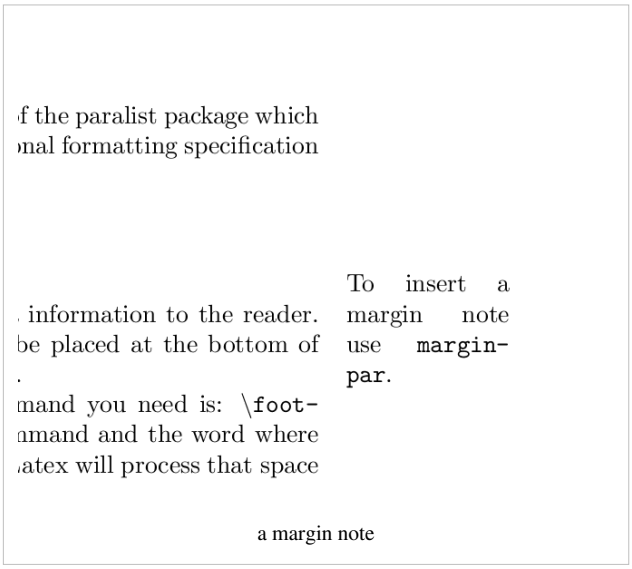
- If the footnote is intended to be added to the title of a chapter, a section, or similar, two methods can be used:

  `Write \section[title] {title\footnote{i'm a footnote referred to the section}}` where "title" is the title of the section.

  Use the footmisc package, with package option *stable*, and simply add the footnote to the section title.

Margin Notes

Margin Notes are useful during the editorial process, to exchange comments among authors. To insert a margin note use `\marginpar{margin text}` . For one-sided layout (simplex), the text will be placed in the right margin, starting from the line where it is defined. For two-sided layout (duplex), it will be placed in the outside margin and for two-column layout it will be placed in the nearest margin.



a margin note

To swap the default side, use `\reversemarginpar` and margin notes will then be placed on the opposite side, which would be the inside margin for two-sided layout.

If the text of your marginpar depends on which margin it is put in (say it includes an arrow pointing at the text or refers to a direction as in "as seen to the left..."), you can use `\marginpar[left text]{right text}` to specify the variants.

To insert a margin note in an area that `\marginpar` can't handle, such as footnotes or equation environments, use the package `marginnote`.

Another option for adding colored margin notes in a fancy way provides the package `todonotes` by using `\todo{todo note}` . It makes use of the package `pgf` used for designing and drawing with a huge tool database.

Also see the package `mparhack`.

Summary

Phew! What a busy tutorial! A lot of material was covered here, mainly because formatting is such a broad topic. Latex is so flexible that we actually only skimmed the surface, as you can have much more control over the presentation of your document if you wish. Having said that, one of the purposes of Latex is to take away the stress of having to deal with the physical presentation yourself, so you need not get too carried away!

This page uses material from Andy Roberts' Getting to grips with Latex [3] *with permission from the author.*

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/microtype/
[2] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=casechange
[3] http://www.ctan.org/tex-archive/macros/latex/contrib/mhchem/
[4] http://www.ctan.org/tex-archive/macros/latex/contrib/bpchem/
[5] http://www.ctan.org/tex-archive/macros/latex/contrib/koma-script/
[6] http://tug.ctan.org/pkg/fixltx2e
[7] http://www.ctan.org/tex-archive/help/Catalogue/entries/footmisc.html
[8] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=footintab
[9] http://www.cs.brown.edu/system/software/latex/doc/mpfnmark.pdf

Fonts

In order to select a font other than the default typeface in Latex environment, it is necessary to include some commands in the preamble of the document.

For example:

```
\usepackage[T1]{fontenc}
\usepackage[light,math]{iwona}
```

Important: note that this only works for fonts that are already prepared for use with LaTeX. If what you have is a ttf font or similar, you will have to convert it and make it available to LaTeX. See the external links section below for some useful resources.

Example

Below is an example found at the Google discussion group latexlovers. The example demonstrates how to select different fonts in a simple document.

```
\documentclass{book}

\begin{document}

 % using default font  (\familydefault = \rmdefault = Computer Modern
Roman)
 Lorem ipsum dolor sit amet, consectitur adipiscing elit.

 \renewcommand*\rmdefault{ppl}\normalfont\upshape
 Lorem ipsum dolor sit amet, consectitur adipiscing elit. % using
Palatino font

 \renewcommand*\rmdefault{iwona}\normalfont\upshape
 Lorem ipsum dolor sit amet, consectitur adipiscing elit. % using Iwona
 font

\end{document}
```

XeTeX

If you use the XeTeX or LuaTeX engine and the fontspec [1] package, you'll be able to use any font installed in the system effortlessly. XeTeX also allows using OpenType technology of modern fonts like specifying alternate glyphs and optical size variants. XeTeX also uses Unicode by default, which might be helpful for font issues.

To use the fonts, simply load the fontspec package and set the font:

```
\documentclass{article}

\usepackage{fontspec}
\setmainfont{Arial}

\begin{document}
Lorem ipsum...
```

```
\end{document}
```

Then compile the document with XeLateX or LuaLaTeX. Note that you can only generate .pdf files, and that you need a sufficiently new TeX distribution (TeX Live 2009 should work for XeTeX and Tex Live 2010 for LuaTeX). Also you should NOT load the inputenc or fontenc package. To make support both pdflatex and XeTeX you can use the `\ifxetex` macro from the ifxetex [2] package.

```
\documentclass{article}
\usepackage{ifxetex}

\ifxetex
  \usepackage{fontspec}
  \usepackage{xunicode}
  \defaultfontfeatures{Mapping=tex-text} % To support LaTeX quoting
style
  \setromanfont{Hoefler Text}
\else
  \usepackage[utf8]{inputenc}
  \usepackage[T1]{fontenc}
\fi

\begin{document}
Lorem ipsum...
\end{document}
```

Useful websites

- The Latex Font Catalogue [3]
- LaTeX font commands [4]
- How to change fonts in Latex [5]
- Understanding the world of TEX fonts and mastering the basics of fontinst [6]
- Font installation the shallow way [7] *"For one-off projects, you can cut corners with font installation* (i.e. fontinst) *and end up with a more manageable set of files and a cleaner TEX installation. This article shows how and why"*

TrueType (ttf) fonts

- Step-by-step guide to manually install a ttf-font for PdfTeX [8]
- A bash script for installing a LaTeX font family [9] (MikTeX [10] / TeXLive [10])
- LaTeX And TrueType Font [11]
- True Type Fonts with LaTeX under Linux + MiKTeX 2.5 [12]
- Unicode Truetype font installer for LaTeX under Windows + MikTeX [13]
- Using TrueType fonts with TeX (LaTeX) and pdfTeX (pdfLaTeX) [14] (for MikTeX)

References

[1] http://www.ctan.org/tex-archive/help/Catalogue/entries/fontspec.html
[2] http://www.ctan.org/tex-archive/macros/generic/ifxetex/
[3] http://www.tug.dk/FontCatalogue/
[4] http://www.cl.cam.ac.uk/~rf10/pstex/latexcommands.htm
[5] http://www.ee.iitb.ac.in/~trivedi/LatexHelp/latexfont.htm
[6] ftp://tug.ctan.org/tex-archive/fonts/utilities/fontinst/doc/talks/et99-font-tutorial.pdf
[7] http://www.tug.org/TUGboat/Articles/tb27-1/tb86kroonenberg-fonts.pdf
[8] http://c.caignaert.free.fr/Install-ttf-Font.pdf
[9] http://www.tex.ac.uk/ctan/support/installfont/installfont.pdf
[10] http://latex.josef-kleber.de/download/installfont-tl
[11] http://xpt.sourceforge.net/techdocs/language/latex/latex33-LaTeXAndTrueTypeFont
[12] http://fachschaft.physik.uni-greifswald.de/~stitch/ttf.html
[13] http://william.famille-blum.org/software/latexttf/index.html
[14] http://www.radamir.com/tex/ttf-tex.htm

List Structures

Convenient and predictable list formatting is one of the many advantages of using LaTeX. Many users of wysiwyg word processors are frequently frustrated by the software's clumsy attempts to figure out when you intend lists to begin and end. This is the price of auto-formatting. As a mark-up language, LaTeX gives you *far* more control over the structure and content of your list. With a little practice you will find that creating lists in LaTeX is actually a pleasure when compared to wrestling with your typical "high power" word processor.

List Structures

Lists often appear in documents, especially academic, as their purpose is often to present information in a clear and concise fashion. List structures in LaTeX are simply environments which essentially come in three flavors: `itemize` , `enumerate` and `description` .

All lists follow the basic format:

```
\begin{list_type}

  \item The first item
  \item The second item
  \item The third etc \ldots

\end{list_type}
```

All three of these types of lists can have multiple paragraphs per item: just type the additional paragraphs in the normal way, with a blank line between each. So long as they are still contained within the enclosing environment, they will automatically be indented to follow underneath their item.

Itemize

This environment is for your standard bulleted list of items.

```
\begin{itemize} \item The first item \item The second item \item
The third etc \ldots \end{itemize}
```

- The first item
- The second item
- The third etc …

Enumerate

The enumerate environment is for ordered lists, where by default, each item is numbered sequentially.

```
\begin{enumerate} \item The first item \item The second item
\item The third etc \ldots \end{enumerate}
```

1. The first item
2. The second item
3. The third etc …

Description

The description environment is slightly different. You can specify the item label by passing it as an optional argument (although optional, it would look odd if you didn't include it!). Ideal for a series of definitions, such as a glossary.

```
\begin{description} \item[First] The first item
\item[Second] The second item \item[Third] The third etc
\ldots \end{description}
```

First The first item

Second The second item

Third The third etc …

Sometimes you want a description where the text begins on a new line. This cannot easily be done with `\\` . The trick is to use `\hfill` .

```
\begin{description} \item[First] \hfill \\ The first item
\item[Second] \hfill \\ The second item \item[Third] \hfill \\
The third etc \ldots \end{description}
```

First
The first item

Second
The second item

Third
The third etc …

Compacted lists

As you may have noticed, in standard LaTeX document classes, the vertical spacing between items, and above and below the lists as a whole, is more than between paragraphs: it may look odd if the descriptions are too short. If you want tightly-packed lists, use the `mdwlist` package (included in the `mdwtools` bundle), which provides compact, "starred" versions of the previous environments, i.e. `itemize*` , `enumerate*` and `description*` . They work exactly in the same way, but the output is more compact. Other packages providing compacted lists are `paralist` and `enumitem`.

Alternatively, use the `memoir` class and with `\tightlists`.

Nested Lists

Latex will happily allow you to insert a list environment into an existing one (up to a depth of four -- if you need more than four, use the easylist package). Simply begin the appropriate environment at the desired point within the current list. Latex will sort out the layout and any numbering for you.

```
\begin{enumerate} \item The first item \begin{enumerate} \item
Nested item 1 \item Nested item 2 \end{enumerate} \item The
second item \item The third etc \ldots \end{enumerate}
```

1. The first item
   (a) Nested item 1
   (b) Nested item 2
2. The second item
3. The third etc …

Customizing Lists

Customizing LaTeX is outside the beginners' domain. While not necessarily difficult in itself, because beginners are already overwhelmed with the array of commands and environments, moving on to more advanced topics runs the risk of confusion.

However, since the tutorial is on formatting, I shall still include a brief guide on customizing lists. Feel free to skip!

Customizing Line Spacing in Lists

Inside lists you can redefine some length/dimension variables of latex, for example using:

```
\begin{itemize}  \setlength{\itemsep}{1pt}  \setlength{\parskip}{0pt}
\setlength{\parsep}{0pt} \item first item \item second item \end{itemize}
```

Alternatively, to create a unified look in your document you can define your own enumerate environment:

```
\newenvironment{my_enumerate}  {\begin{enumerate}  \setlength{\itemsep}{1pt}
\setlength{\parskip}{0pt} \setlength{\parsep}{0pt}} {\end{enumerate}}
```

Customizing Enumerated Lists

The thing people want to change most often with Enumerated lists are the counters. A quick solution to this problem is provided by the `enumerate` package of David Carlisle[1] , or the more sophisticated package `enumitem` by Javier Bezos[2] .

To go any further and do it yourself instead, a brief introduction to LaTeX *counters* is required. For anything that LaTeX automatically numbers, such as section headers, figures, and itemized lists, there is a counter associated with it that controls the numbering.

There are four individual counters that are associated with itemized lists, each one represents the four possible levels of nesting, which are called: `enumi` , `enumii` , `enumiii` , `enumiv` . In order to reset any of these counters in the middle of an enumeration simply use `\setcounter` . The counter is incremented by `\item` before it is printed. For example to reset `enumi` use:

```
\begin{enumerate}
  \setcounter{enumi}{4}
  \item fifth element
\end{enumerate}
```

which prints as:

```
5. fifth element
```

Each counter also has a default format that dictates how it is displayed whenever LaTeX needs to print it. Such formats are specified using internal LaTeX commands:

Command	Example
\arabic	1, 2, 3 ...
\alph	a, b, c ...
\Alph	A, B, C ...
\roman	i, ii, iii ...
\Roman	I, II, III ...
\fnsymbol	Aimed at footnotes (see below), but prints a sequence of symbols.

Each counter entity holds various bits of information about itself. To get to the numbered element, simply use **\the** followed immediately (i.e., no space) by the name of the counter, e.g., **\theenumi** . This is often referred to as the *representation* of a counter.

Now, that's most of the technicalities out of the way. To make changes to the formatting of a given level:

```
\renewcommand{\representation}{\format_command{counter}}
```

Admittedly, the generic version is not that clear, so a couple of examples will clarify:

```
%Redefine the first level
\renewcommand{\theenumi}{\Roman{enumi}}
\renewcommand{\labelenumi}{\theenumi}

%Redefine the second level
\renewcommand{\theenumii}{\Alph{enumii}}
\renewcommand{\labelenumii}{\theenumii}
```

The method used above first explicitly changes the format used by the counter. However, the element that controls the label needs to be updated to reflect the change, which is what the second line does. Another way to achieve this result is this:

```
\renewcommand{\labelenumi}{\Roman{enumi}}
```

This simply redefines the appearance of the label, which is fine, providing that you do not intend to cross-reference to a specific item within the list, in which case the reference will be printed in the previous format. This issue does not arise in the first example.

Note that you can also add other symbols, such as parentheses and periods, before and after the counter. For instance, to create a list indexed by lower case letters with parentheses before and after the letter, you might enter the following:

```
\renewcommand{\labelenumi}{(\alph{enumi})}
```

Customizing Itemised Lists

Itemized lists are not as complex as they do not need to count. Therefore, to customize, you simply change the labels. It can be done manually for each entry with `\item[new symbol]` , eg `\item[$\star$]` .

The itemize labels are accessed via `\labelitemi` , `\labelitemii` , `\labelitemiii` , `\labelitemiv` , for the four respective levels.

```
\renewcommand{\labelitemi}{\textgreater}
```

The above example would set the labels for the first level to a greater than (>) symbol. Of course, the text symbols available in Latex are not very exciting. Why not use one of the ZapfDingbat symbols, as described in the Symbols section. Or use a mathematical symbol:

```
\renewcommand{\labelitemi}{$\star$}
```

Itemized list with tightly set items, that is with no vertical space between two consecutive items, can be created as follows.

```
\begin{itemize}
  \setlength{\itemsep}{0cm}%
  \setlength{\parskip}{0cm}%
  \item Item opening the list
  \item Item tightly following
\end{itemize}
```

Details of Customizing Lists

Note that it is necessary that the `\renewcommand` appears after the `\begin{document}` instruction so the changes made are taken into account. This is needed for both enumerated and itemized lists.

Inline lists

Inline lists are a special case as they require the use of the `paralist` package which provides the `inparaenum` environment (with an optional formatting specification in square brackets):

```
... \usepackage{paralist} \begin{document}
\textbf{\itshape Inline lists}, which are
sequential in nature, just like enumerated lists,
but are \begin{inparaenum}[\itshape a\upshape)]
\item formatted within their paragraph; \item
usually labelled with letters; and \item usually
have the final item prefixed with `and' or `or',
\end{inparaenum} like this example. ...
```

Inline lists, which are sequential in nature, just like enumerated lists, but are *a)* formatted within their paragraph; *b)* usually labelled with letters; and *c)* usually have the final item prefixed with 'and' or 'or', like this example.

To change the styles of the counter, tokens A, a, I, i, and 1 can be used in the optional argument to produce the counter with one of the styles `\Alph` , `\alph` , `\Roman` , `\roman` and `\arabic` . For example:

```
\begin{inparaenum}[(i)]
```

produces the labels (i), (ii), (iii) ...

Other packages providing inline lists are `shortlst` and `enumitem`.

[1] (http://mirrors.fe.up.pt/pub/CTAN/macros/latex/required/tools/enumerate.pdf)The enumerate package, David Carlisle 1999
[2] (http://mirrors.fe.up.pt/pub/CTAN/macros/latex/contrib/enumitem/enumitem.pdf)The enumitem package, Javier Bezos 2011

Tables

In academic writing, tables are a common feature, often for summarising results from research. It is therefore a skill that needs mastering in order to produce quality papers.

However, if there is one area about LaTeX that is the least intuitive, then this is it. Basic tables are not too taxing, but you will quickly notice that anything more advanced can take a fair bit of construction. So, we start slowly and build up from there.

Workaround: You might save lots of time by converting tables from Excel or OpenOffice spreadsheets with the help of open source plugins, see e.g. http://calc2latex.sourceforge.net/ [1] for OpenOffice spreedsheets and http://www.ctan.org/tex-archive/support/excel2latex/ [2] for Microsoft Office Excel. For people used to matlab, there is also plugin called matrix2latex [3]

The `tabular` environment

The `tabular` environment can be used to typeset tables with optional horizontal and vertical lines. LaTeX determines the width of the columns automatically.

The first line of the environment has the form: `\begin{tabular}[pos]{table spec}`

the *table spec* argument tells LaTeX the alignment to be used in each column and the vertical lines to insert.

The number of columns does not need to be specified as it is inferred by looking at the number of arguments provided. It is also possible to add vertical lines between the columns here. The following symbols are available to describe the table columns (some of them require that the package *array* has been loaded):

`l`	left-justified column
`c`	centered column
`r`	right-justified column
`p{`*`width`*`}`	paragraph column with text vertically aligned at the top
`m{`*`width`*`}`	paragraph column with text vertically aligned in the middle (requires array package)
`b{`*`width`*`}`	paragraph column with text vertically aligned at the bottom (requires array package)
`|`	vertical line
`||`	double vertical line

By default, if the text in a column is too wide for the page, LaTeX won't automatically wrap it. Using `p{`*`width`*`}` you can define a special type of column which will wrap-around the text as in a normal paragraph. You can pass the width using any unit supported by LaTeX, such as pt and cm, or *command lengths*, such as `\textwidth`.You can find a complete list in appendix Useful Measurement Macros.

The optional parameter *pos* can be used to specify the vertical position of the table relative to the baseline of the surrounding text. In most cases, you will not need this option. It becomes relevant only if your table is not in a paragraph of its own. You can use the following letters:

`b`	bottom
`c`	center (default)
`t`	top

In the first line you have pointed out how many columns you want, their alignment and the vertical lines to separate them. Once in the environment, you have to introduce the text you want, separating between cells and introducing new lines. The commands you have to use are the following:

`&`	column separator
`\\`	start new row (additional space may be specified after `\\` using square brackets, such as `\\[6pt]`)
`\hline`	horizontal line
`\newline`	start a new line within a cell
`\cline{`*i*`-`*j*`}`	partial horizontal line beginning in column *i* and ending in column *j*

Note, any white space inserted between these commands is purely down to ones' preferences. I personally add spaces between to make it easier to read.

Basic examples

This example shows how to create a simple table in LaTeX. It is a three-by-three table, but without any lines.

```
\begin{tabular}{ l c r } 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

Expanding upon that by including some vertical lines:

```
\begin{tabular}{ l | c || r | } 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

To add horizontal lines to the very top and bottom edges of the table:

```
\begin{tabular}{ l | c || r | } \hline 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 &
8 & 9 \\ \hline \end{tabular}
```

1	2	3
4	5	6
7	8	9

And finally, to add lines between all rows, as well as centering (notice the use of the center environment - of course, the result of this is not obvious from the preview on this web page):

```
\begin{center} \begin{tabular}{ l | c || r | } \hline 1 & 2 & 3 \\
\hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \\ \hline \end{tabular}
\end{center}
```

1	2	3
4	5	6
7	8	9

```
\begin{tabular}{|r|l|} \hline 7C0 & hexadecimal \\
3700 & octal \\ \cline{2-2} 11111000000 & binary \\
\hline \hline 1984 & decimal \\ \hline \end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

Column specification using >{\cmd} and <{\cmd}

Using the array package, the column specification can be altered. This is done in the argument of the tabular environment using `>{\command}` for commands executed right *before* each column element and `<{\command}` for commands to be executed right *after* each column element. As an example: to get a column in math mode enter: `\begin{tabular}{>{$}c<{$}}`. Another example is changing the font: `\begin{tabular}{>{\small}c}` to print the column in a small font.

The argument of the `>` and `<` specifications must be correctly balanced when it comes to `{` and `}` characters. This means that `>{\bfseries}` is valid, while `>{\textbf}` will not work and `>{\textbf{}` is not valid. If there is the need to use the text of the table as an argument (for instance, using the `\textbf` to produce bold text), one should use the `\bgroup` and `\egroup` commands: `>{\textbf\bgroup}c<{\egroup}` produces the intended effect. This works only for some basic LaTeX commands. For other commands, such as `\underline` to underline text, it is necessary to temporarily store the column text in a box using `lrbox`. First, you must define such a box with `\newsavebox{\boxname}` and then you can define:

```
>{\begin{lrbox}{\boxname}}% l% <{\end{lrbox}% \underline{\unhbox\boxname}}% }
```

This stores the text in a box and afterwards, takes the text out of the box with `\unhbox` (this destroys the box, if the box is needed again one should use `\unhcopy` instead) and passing it to `\underline`. (For LaTeX2e, you may want to use `\usebox{\boxname}` instead of `\unhbox\boxname`.)

This same trick done with `\raisebox` instead of `\underline` can force all lines in a table to have equal height, instead of the natural varying height that can occur when e.g. math terms or superscripts occur in the text.

Here is an example showing the use of both `p{...}` and `>{\centering}` :

```
\begin{tabular}{>{\centering}p{3.5cm}>{\centering}p{3.5cm}} Geometry & Algebra
\tabularnewline \hline Points & Addition \tabularnewline Spheres &
Multiplication \end{tabular}
```

Note the use of `\tabularnewline` instead of `\\` to avoid a `Misplaced \noalign` error.

Text wrapping in tables

LaTeX's algorithms for formatting tables have a few shortcomings. One is that it will not automatically wrap text in cells, even if it overruns the width of the page. For columns that you know will contain a certain amount of text, then it is recommended that you use the *p* attribute and specify the desired width of the column (although it may take some trial-and-error to get the result you want). Use the *m* attribute to have the lines aligned toward the middle of the box and the *b* attribute to align along the bottom of the box.

Here is a practical example. The following code creates two tables with the same code; the only difference is that the last column of the second one has a defined width of 5 centimeters, while in the first one we didn't specify any width. Compiling this code:

```
\documentclass{article} \usepackage[english]{babel} \begin{document} Without
specifying width for last column: \begin{center} \begin{tabular}{ | l | l | l
| l |} \hline Day & Min Temp & Max Temp & Summary \\ \hline Monday & 11C & 22C
& A clear day with lots of sunshine. However, the strong breeze will bring
down the temperatures. \\ \hline Tuesday & 9C & 19C & Cloudy with rain, across
many northern regions. Clear spells across most of Scotland and Northern
Ireland, but rain reaching the far northwest. \\ \hline Wednesday & 10C & 21C
& Rain will still linger for the morning. Conditions will improve by early
afternoon and continue throughout the evening. \\ \hline \end{tabular}
\end{center} With width specified: \begin{center} \begin{tabular}{ | l | l | l
| p{5cm} |} \hline Day & Min Temp & Max Temp & Summary \\ \hline Monday & 11C
```

```
& 22C & A clear day with lots of sunshine. However, the strong breeze will
bring down the temperatures. \\ \hline Tuesday & 9C & 19C & Cloudy with rain,
across many northern regions. Clear spells across most of Scotland and
Northern Ireland, but rain reaching the far northwest. \\ \hline Wednesday &
10C & 21C & Rain will still linger for the morning. Conditions will improve by
early afternoon and continue throughout the evening. \\ \hline \end{tabular}
\end{center} \end{document}
```

You get the following output:

Without specifying width for last column:

Day	Min Temp	Max Temp	Summary
Monday	11C	22C	A clear day with lots of sunshine. However, the strong breeze w
Tuesday	9C	19C	Cloudy with rain, across many northern regions. Clear spells ac
Wednesday	10C	21C	Rain will still linger for the morning. Conditions will improve b

With width specified:

Day	Min Temp	Max Temp	Summary
Monday	11C	22C	A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures.
Tuesday	9C	19C	Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest.
Wednesday	10C	21C	Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening.

Note that the first table is cropped: The output is wider than the page width.

Text justification in tables

On rare occasions, it might be necessary to stretch every row in a table to the natural width of its longest line, for instance when one has the same text in two languages and wishes to present these next to each other with lines synching up. A tabular environment helps control where lines should break, but cannot justify the text, which leads to ragged right edges. The `eqparbox` package provides the command `\eqmakebox` which is like `\makebox` but instead of a *width* argument, it takes a tag. During compilation it bookkeeps which `\eqmakebox` with a certain tag contains the widest text and can stretch all `\eqmakebox`es with the same tag to that width. Combined with the `array` package, one can define a column specifier that justifies the text in all lines: (See the documentation of the `eqparbox` package for more details.)

```
\newsavebox{\tstretchbox}                                    \newcolumntype{S}[1]{%
>{\begin{lrbox}{\tstretchbox}}%                    l%              <{\end{lrbox}%
\eqmakebox[#1][s]{\unhcopy\tstretchbox}}% }
```

Other environments inside tables

If you use some LaTeX environments inside table cells, like `verbatim` or `enumerate`

```
\begin{tabular}{| c | c |} \hline \begin{verbatim} code \end{verbatim} & description \\ \hline \end{tabular}
```

you might encounter errors similar to

```
! LaTeX Error: Something's wrong--perhaps a missing \item.
```

To solve this problem, change column specifier to "paragraph" (`p`, `m` or `b`).

```
\begin{tabular}{| m{5cm} | c |}
```

Defining multiple columns

It is possible to define many identical columns at once using the `*{num}{str}` syntax.

This is particularly useful when your table has many columns.

Here is a table with six centered columns flanked by a single column on each side:

```
\begin{tabular}{l*{6}{c}r} Team & P & W & D & L & F & A & Pts \\ \hline
Manchester United & 6 & 4 & 0 & 2 & 10 & 5 & 12 \\ Celtic & 6 & 3 & 0 & 3 & 8 & 9 & 9 \\ Benfica & 6 & 2 & 1 & 3 & 7 & 8 & 7 \\ FC Copenhagen & 6 & 2 & 1 & 2 & 5 & 8 & 7 \\ \end{tabular}
```

Team	P	W	D	L	F	A	Pts
Manchester United	6	4	0	2	10	5	12
Celtic	6	3	0	3	8	9	9
Benfica	6	2	1	3	7	8	7
FC Copenhagen	6	2	1	2	5	8	7

@-expressions

The column separator can be specified with the `@{...}` construct.

It typically takes some text as its argument, and when appended to a column, it will automatically insert that text into each cell in that column before the actual data for that cell. This command kills the inter-column space and replaces it with whatever is between the curly braces. To add space, use `@{\hspace{width}}`.

Admittedly, this is not that clear, and so will require a few examples to clarify. Sometimes, it is desirable in scientific tables to have the numbers aligned on the decimal point. This can be achieved by doing the following:

```
\begin{tabular}{r@{.}l} 3 & 14159 \\ 16 & 2 \\ 123 & 456 \\
\end{tabular}
```

3.14159
16.2
123.456

Note that the approach outlined above won't work well if the column header is longer than any of the numbers. To center the column on the decimal separator, use the `dcolumn` package, which provides a new column specifier for floating point data.

The space suppressing qualities of the @-expression actually make it quite useful for manipulating the horizontal spacing between columns. Given a basic table, and varying the column descriptions:

```
\begin{tabular}{|l|l|} \hline stuff & stuff \\ \hline stuff & stuff \\ \hline
\end{tabular}
```

```
{|l|l|}
```

stuff	stuff
stuff	stuff

```
{|@{}l|l@{}|}
```

stuff	stuff
stuff	stuff

```
{|@{}l@{}|l@{}|}
```

stuff	stuff
stuff	stuff

```
{|@{}l@{}|@{}l@{}|}
```

stuff	stuff
stuff	stuff

Spanning

To complete this tutorial, we take a quick look at how to generate slightly more complex tables. Unsurprisingly, the commands necessary have to be embedded within the table data itself.

Rows spanning multiple columns

The command for this looks like this: `\multicolumn{num_cols}{alignment}{contents}`. *num_cols* is the number of subsequent columns to merge; *alignment* is, either l, c, r or to have text wrapping specifiy a width `p{5.0cm}` . And *contents* is simply the actual data you want to be contained within that cell. A simple example:

```
\begin{tabular}{|l|l|} \hline \multicolumn{2}{|c|}{Team sheet}
\\ \hline GK & Paul Robinson \\ LB & Lucus Radebe \\ DC &
Michael Duberry \\ DC & Dominic Matteo \\ RB & Didier Domi \\
MC & David Batty \\ MC & Eirik Bakke \\ MC & Jody Morris \\ FW
& Jamie McMaster \\ ST & Alan Smith \\ ST & Mark Viduka \\
\hline \end{tabular}
```

Team sheet	
GK	Paul Robinson
LB	Lucus Radebe
DC	Michael Duberry
DC	Dominic Matteo
RB	Didier Domi
MC	David Batty
MC	Eirik Bakke
MC	Jody Morris
FW	Jamie McMaster
ST	Alan Smith
ST	Mark Viduka

Columns spanning multiple rows

The first thing you need to do is add `\usepackage{multirow}` to the preamble[4] . This then provides the command needed for spanning rows: `\multirow{num_rows}{width}{contents}`. The arguments are pretty simple to deduce (`*` for the *width* means the content's natural width).

```
... \usepackage{multirow} ... \begin{tabular}{|l|l|l|}
\hline \multicolumn{3}{|c|}{Team sheet} \\ \hline
Goalkeeper & GK & Paul Robinson \\ \hline
\multirow{4}{*}{Defenders} & LB & Lucus Radebe \\ & DC
& Michael Duberry \\ & DC & Dominic Matteo \\ & RB &
Didier Domi \\ \hline \multirow{3}{*}{Midfielders} &
MC & David Batty \\ & MC & Eirik Bakke \\ & MC & Jody
Morris \\ \hline Forward & FW & Jamie McMaster \\
\hline \multirow{2}{*}{Strikers} & ST & Alan Smith \\
& ST & Mark Viduka \\ \hline \end{tabular}
```

Team sheet		
Goalkeeper	GK	Paul Robinson
Defenders	LB	Lucus Radebe
	DC	Michael Duberry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

The main thing to note when using `\multirow` is that a blank entry must be inserted for each appropriate cell in each subsequent row to be spanned.

If there is no data for a cell, just don't type anything, but you still need the "&" separating it from the next column's data. The astute reader will already have deduced that for a table of n columns, there must always be $n-1$ ampersands in each row. The exception to this is when `\multicolumn` and `\multirow` are used to create cells which span multiple columns or rows.

Spanning in both directions simultaneously

Here is a nontrivial example how to use spanning in both directions simultaneously and have the borders of the cells drawn correctly:

```
\usepackage{multirow} \begin{tabular}{cc|c|c|c|c|l}
\cline{3-6} & & \multicolumn{4}{|c|}{Primes} \\
\cline{3-6} & & 2 & 3 & 5 & 7 \\ \cline{1-6}
\multicolumn{1}{|c|}{\multirow{2}{*}{Powers}} &
\multicolumn{1}{|c|}{504} & 3 & 2 & 0 & 1 & \\
\cline{2-6} \multicolumn{1}{|c|}{} &
\multicolumn{1}{|c|}{540} & 2 & 3 & 1 & 0 & \\
\cline{1-6}
\multicolumn{1}{|c|}{\multirow{2}{*}{Powers}} &
\multicolumn{1}{|c|}{gcd} & 2 & 2 & 0 & 0 & min \\
\cline{2-6} \multicolumn{1}{|c|}{} &
\multicolumn{1}{|c|}{lcm} & 3 & 3 & 1 & 1 & max \\
\cline{1-6} \end{tabular}
```

		Primes				
		2	3	5	7	
Powers	504	3	2	0	1	
	540	2	3	1	0	
Powers	gcd	2	2	0	0	min
	lcm	3	3	1	1	max

The command `\multicolumn{1}{|c|}{...}` is just used to draw vertical borders both on the left and on the right of the cell. Even when combined with `\multirow{2}{*}{...}`, it still draws vertical borders that only span the first row. To compensate for that, we add `\multicolumn{1}{|c|}{...}` in the following rows spanned by the multirow. Note that we cannot just use `\hline` to draw horizontal lines, since we do not want the line to be drawn over the text that spans several rows. Instead we use the command `\cline{2-6}` and opt out the first column that contains the text "Powers".

Here is another example exploiting the same ideas to make the familiar and popular "2x2" or double dichotomy:

```
\begin{tabular}{r|c|c|} \multicolumn{1}{r|}{} &
\multicolumn{1}{c|}{noninteractive} &
\multicolumn{1}{c|}{interactive} \\ \cline{2-3}
massively multiple & Library & University \\
\cline{2-3} one-to-one & Book & Tutor \\
\cline{2-3} \end{tabular}
```

	noninteractive	interactive
massively multiple	Library	University
one-to-one	Book	Tutor

Resize tables

The command `\resizebox{width}{height}{object}` can be used with `tabular` to specify the height and width of a table. The following example shows how to resize a table to 8cm width while maintaining the original width/height ratio.

```
\resizebox{8cm}{!} { \begin{tabular}... \end{tabular} }
```

Alternatively you can use `\scalebox{ratio}{object}` in the same way but with ratios rather than fixed sizes:

```
\scalebox{0.7}{ \begin{tabular}... \end{tabular} }
```

Both `\resizebox` and `\scalebox` require the graphicx package.

To tweak the space between columns (LaTeX will by default choose very tight columns), one can alter the column separation: `\setlength{\tabcolsep}{5pt}`. The default value is 6pt.

Sideways tables

Tables can also be put on their side within a document using the `rotating` package and the `sidewaystable` environments in place of the table environment. (NOTE: most DVI viewers do not support displaying rotated text. Convert your document to a PDF to see the result. Most, if not all, PDF viewers do support rotated text.)

```
\usepackage{rotating} \begin{sidewaystable} \begin{tabular}... \end{tabular}
\end{sidewaystable}
```

When it is desirable to place the rotated table at the exact location where it appears in the source (.tex) file, `rotfloat` package may be used. Then one can use `\begin{sidewaystable}[H]` just like for normal tables. The 'H' option can not be used without this package.

Alternate Row Colors in Tables

The `xcolor` package provides the necessary commands to produce tables with alternate row colors, when loaded with the `table` option. The command `\rowcolors{<starting row>}{<odd color>}{<even color>}` has to be specified right before the `tabular` environment starts.

```
\documentclass{article}
\usepackage[table]{xcolor} \begin{document}
\begin{center} \rowcolors{1}{green}{pink}
\begin{tabular}{lll} odd & odd & odd \\ even &
even & even\\ odd & odd & odd \\ even & even &
even\\ \end{tabular} \end{center}
\end{document}
```

odd	odd	odd
even	even	even
odd	odd	odd
even	even	even

The command `\hiderowcolors` is available to deactivate highlighting of a specified row. Highlighting can be reactivated within the table via the `\showrowcolors` command.

Colors of individual Cells

As above this uses the `xcolor` package.

```
% Include this somewhere in your document \usepackage[table]{xcolor} % Enter
this in the cell you wish to color a light grey. % NB: the word 'gray' here
denotes the grayscale color scheme, not the color grey. `0.9' denotes how dark
the grey is. \cellcolor[gray]{0.9} % The following will color the cell red.
\cellcolor{red}
```

Partial Vertical Lines

Adding a partial vertical line to an individual cell:

```
\begin{tabular}{ l c r } \hline 1 & 2 & 3 \\ \hline 4 & 5 &
\multicolumn{1}{r|}{6} \\ \hline 7 & 8 & 9 \\ \hline \end{tabular}
```

1	2	3
4	5	6
7	8	9

Removing part of a vertical line in a particular cell:

```
\begin{tabular}{ | l | c | r | } \hline 1 & 2 & 3 \\ \hline 4 & 5 &
\multicolumn{1}{r}{6} \\ \hline 7 & 8 & 9 \\ \hline \end{tabular}
```

1	2	3
4	5	6
7	8	9

The `table` environment - captioning etc

The `tabular` environment doesn't cover all that you need to do with tables. For example, you might want a caption for your table. For this and other reasons, you should typically place your `tabular` environment inside a `table` environment:

```
\begin{table} \caption{Performance at peak F-measure} \begin{tabular}{| r | r || c | c | c |} ... \end{tabular} \end{table}
```

Why do the two different environments exist? Think of it this way: The `tabular` environment is concerned with arranging elements in a tabular grid, while the `table` environment represents the table more conceptually. This explains why it isn't `tabular` but `table` that provides for captioning (because the caption isn't displayed in the grid-like layout).

A `table` environment has a lot of similarities with a `figure` environment, in the way the "floating" is handled etc. For instance you can specify its placement in the page with the option `[placement]`, the valid values are any combination of (order is not important):

`h`	where the table is declared (**h**ere)
`t`	at the **t**op of the page
`b`	at the **b**ottom of the page
`p`	on a dedicated **p**age of floats
`!`	override the default float restrictions. E.g., the maximum size allowed of a `b` float is normally quite small; if you want a large one, you need this `!` parameter as well.

The default is `[tbp]`. If you want to place a table in the place where it's declared, do not just write `[h]`; if the table cannot fit (because the text is near the bottom of the page, say) it will float to a dedicated page of floats (as if it were a `p` float) which can be some distance away in the document. A good rule of thumb is to always use `htbp` until the document is finished, at which stage the final float parameters can be fine-tuned.

The `table` environment is also useful when you want to have a list of tables at the beginning or end of your document with the command `\listoftables`; it enables making cross-references to the table with:

```
You may refer to table~\ref{my_table} for an example. ... \begin{table} \begin{tabular} ... \end{tabular} \caption{An example of table} \label{my_table} \end{table}
```

The `tabular*` environment - controlling table width

This is basically a slight extension on the original tabular version, although it requires an extra argument (before the column descriptions) to specify the preferred width of the table.

```
\begin{tabular*}{0.75\textwidth}{ | c | c | c | r | } \hline label 1 & label 2 & label 3 & label 4 \\ \hline item 1 & item 2 & item 3 & item 4 \\ \hline \end{tabular*}
```

label 1	label 2	label 3	label 4	
item 1	item 2	item 3	item 4	

However, that may not look quite as intended. The columns are still at their natural width (just wide enough to fit their contents) while the rows are as wide as the table width specified. If you do not like this default, you must also explicitly insert extra column space. LaTeX has *rubber lengths*, which, unlike others, are not fixed. LaTeX can dynamically decide how long the lengths should be. So, an example of this is the following.

```
\begin{tabular*}{0.75\textwidth}{@{\extracolsep{\fill}} | c | c | c | r | } \hline label 1 & label 2 & label 3 & label 4 \\ \hline item 1 & item 2 & item
```

```
3 & item 4 \\ \hline \end{tabular*}
```

You will notice the `@{...}` construct added at the beginning of the column description. Within it is the `\extracolsep` command, which requires a width. A fixed width could have been used. However, by using a rubber length, such as `\fill`, the columns are automatically spaced evenly.

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

The `tabularx` package - simple column stretching

This package provides a table environment called `tabularx` which is similar to the `tabular*` environment, except that it has a new column specifier `X` (in uppercase). The column(s) specified with this specifier will be stretched to make the table as wide as specified, greatly simplifying the creation of tables.

```
\usepackage{tabularx} ...
\begin{tabularx}{\textwidth}{
|X|X|X|X| } \hline label 1 & label
2 & label 3 & label 4 \\ \hline
item 1 & item 2 & item 3 & item 4
\\ \hline \end{tabularx}
```

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

The content provided for the boxes is treated as for a `p` column, except that the width is calculated automatically. If you use the package `array`, you may also apply any `>{\cmd}` or `<{\cmd}` command to achieve specific behavior (like `\centering`, or `\raggedright\arraybackslash`) as described previously.

Another option is the use of `\newcolumntype` in order to get selected columns formatted in a different way. It defines a new column specifier, e.g. `R` (in uppercase). In this example, the second and fourth column is adjusted in a different way (`\raggedleft`):

```
\usepackage{tabularx} ...
\newcolumntype{R}{>{\raggedleft\arraybackslash}X}%
\begin{tabularx}{\textwidth}{ |l|R|l|R| } \hline
label 1 & label 2 & label 3 & label 4 \\ \hline
item 1 & item 2 & item 3 & item 4 \\ \hline
\end{tabularx}
```

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

Tabularx with rows spanning multiple columns using `\multicolumn`. The two central columns are posing as one by using the `X@{}` option. Note that the `\multicolumn` width (which in this example is 2) should equal the (in this example 1+1) width of the spanned columns:

```
ize="6.97"> \usepackage{tabularx} ...
tabularx}{1\textwidth}{|>{\setlength\hsize{1\hsize}\centering}X|>{\setlength\hsize{1\hsize}\raggedl
length\hsize{1\hsize}\raggedright}X|>{\setlength\hsize{1\hsize}\centering}X|} \hline Label 1 &
olumn{2}{>{\centering\setlength\hsize{2\hsize}}X|}{Label 2} & Label 3\tabularnewline \hline 123 & 1
23 \tabularnewline \hline 123 & 123 & 456 & 123 \tabularnewline \hline \end{tabularx} </font>
```

Vertically centered images

Inserting images into a table row will align it at the top of the cell. By using the `array` package this problem can be solved. Defining a new columntype will keep the image vertically centered.

```
\newcolumntype{V}{>{\centering\arraybackslash} m{.4\linewidth} }
```

Or use a parbox to center the image.

```
\parbox[c]{1em}{\includegraphics{image.png}}
```

A raisebox works as well, also allowing to manually fine-tune the alignment with its first parameter.

```
\raisebox{-.5\height}{\includegraphics{image.png}}
```

Professional tables

Many professionally typeset books and journals feature simple tables, which have appropriate spacing above and below lines, and almost *never* use vertical rules. Many examples of LaTeX tables (including this Wikibook) showcase the use of vertical rules (using "`|`"), and double-rules (using "`\hline\hline`" or "`||`"), which are regarded as unnecessary and distracting in a professionally published form. The booktabs [5] package is useful for easily providing this professionalism in LaTeX tables, and the documentation [6] also provides guidelines on what constitutes a "good" table.

In brief, the package uses `\toprule` for the uppermost rule (or line), `\midrule` for the rules appearing in the middle of the table (such as under the header), and `\bottomrule` for the lowermost rule. This ensures that the rule weight and spacing are acceptable. In addition, `\cmidrule` can be used for mid-rules that span specified columns. The following example contrasts the use of booktabs and two equivalent normal LaTeX implementations (the second example requires `\usepackage{array}` or `\usepackage{dcolumn}`, and the third example requires `\usepackage{booktabs}` in the preamble).

Normal LaTeX

```
\begin{tabular}{llr}
\hline
\multicolumn{2}{c}{Item}
\\ \cline{1-2} Animal &
Description & Price (\$)
\\ \hline Gnat & per gram
& 13.65 \\ & each & 0.01
\\ Gnu & stuffed & 92.50
\\ Emu & stuffed & 33.33
\\ Armadillo & frozen &
8.99 \\ \hline
\end{tabular}
```

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

Using `array`

```
\usepackage{booktabs} %or
\usepackage{dcolumn} ...
\begin{tabular}{llr}
\firsthline
\multicolumn{2}{c}{Item}
\\ \cline{1-2} Animal &
Description & Price (\$)
\\ \hline Gnat & per gram
& 13.65 \\ & each & 0.01
\\ Gnu & stuffed & 92.50
\\ Emu & stuffed & 33.33
\\ Armadillo & frozen &
8.99 \\ \lasthline
\end{tabular}
```

Using `booktabs`

```
\usepackage{booktabs} ...
\begin{tabular}{llr}
\toprule
\multicolumn{2}{c}{Item}
\\ \cmidrule(r){1-2}
Animal & Description &
Price (\$) \\ \midrule
Gnat & per gram & 13.65
\\ & each & 0.01 \\ Gnu &
stuffed & 92.50 \\ Emu &
stuffed & 33.33 \\
Armadillo & frozen & 8.99
\\ \bottomrule
\end{tabular}
```

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

Usually the need arises for footnotes under a table (and not at the bottom of the page), with a caption properly spaced above the table. These are addressed by the ctable [7] package. It provides the option of a short caption given to be inserted in the list of tables, instead of the actual caption (which may be quite long and inappropriate for the list of tables). The ctable package uses the booktabs package.

Adding rule spacing above or below `\hline` and `\cline` commands

An alternative way to adjust the rule spacing is to add `\noalign{\smallskip}` before or after the `\hline` and `\cline{i-j}` commands:

Normal LaTeX

```
\begin{tabular}{llr} \hline\noalign{\smallskip} \multicolumn{2}{c}{Item} \\
\cline{1-2}\noalign{\smallskip} Animal & Description & Price (\$) \\
\noalign{\smallskip}\hline\noalign{\smallskip} Gnat & per gram & 13.65 \\ & each & 0.01 \\ Gnu & stuffed & 92.50 \\ Emu & stuffed & 33.33 \\ Armadillo & frozen & 8.99 \\ \noalign{\smallskip}\hline \end{tabular}
```

You may also specify the skip after a line explicitly using glue after the line terminator

```
\begin{tabular}{|l|l|} \hline Mineral & Color \\[1cm] Ruby & red \\ Sapphire & blue \\ \hline \end{tabular}
```

Tables with different font size

A table can be globally switched to a different font size by simply adding the desired size command (here: `\footnotesize`) after the `\begin{table}...` statement:

```
\begin{table}[h]\footnotesize \caption{Performance at peak F-measure}
\begin{tabular}{| r | r || c | c | c |} ... \end{tabular} \end{table}
```

The table caption font size is not affected.

To control the caption font size, see Caption Styles.

Table with legend

To add a legend to a table the caption [8] package can be used. With the caption package a `\caption*{...}` statement can be added besides the normal `\caption{...}`.

Example

```
\begin{table} \begin{tabular}{| r | r || c | c | c |} ... \end{tabular}
\caption{A normal caption} \caption*{ A legend, even a table can be used
\begin{tabular}{l l} item 1 & explanation 1 \\ \end{tabular} } \end{table}
```

The normal caption is needed for labels and references.

Need more complicated features?

Have a look at one of the following packages:

- `hhline` [9]: do whatever you want with horizontal lines
- `array` [10]: gives you more freedom on how to define columns
- `colortbl` [11]: make your table more colorful
- `supertabular` [12]: for tables that need to stretch over several pages
- `longtable` [13]: similar to `supertab`.
  - Note: footnotes do not work properly in a normal tabular environment. If you replace it with a longtable environment, footnotes work properly
- `xtab` [14]: Yet another package for tables that need to span many pages
- `tabulary` [15]: modified `tabular*` allowing width of columns set for equal heights
- `arydshln` [16]: creates dashed horizontal and vertical lines
- `ctable` [17]: allows for footnotes under table and properly spaced caption above (incorporates booktabs package)
- `slashbox` [18]: create 2D tables with the first cell containing a description for both axes
- `dcolumn` [19]: decimal point alignment of numeric cells
- `rccol` [20]: advanced decimal point alignment of numeric cells with rounding
- `spreadtab` [21]: spread sheets allowing the use of formulae

Summary

This concludes discussion of basic tables. Experimentation quickly leads to mastery. The table syntax in LaTeX can look rather messy, and seeing new examples can look confusing. But hopefully, enough has been covered here so that a user can create any table needed for your papers. Unsurprisingly, LaTeX has plenty more up its sleeve, so expect a follow up tutorial covering more advanced features in the near future.

References

[1] http://calc2latex.sourceforge.net/
[2] http://www.ctan.org/tex-archive/support/excel2latex/
[3] http://www.mathworks.com/matlabcentral/fileexchange/4894-matrix2latex
[4] Package multirow on CTAN (http://www.ctan.org/tex-archive/macros/latex/contrib/multirow/)
[5] http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/
[6] http://mirrors.ctan.org/macros/latex/contrib/booktabs/booktabs.pdf
[7] http://www.ctan.org/tex-archive/macros/latex/contrib/ctable/
[8] http://www.ctan.org/tex-archive/macros/latex/contrib/caption/
[9] http://tug.ctan.org/pkg/hhline
[10] http://tug.ctan.org/pkg/array
[11] http://tug.ctan.org/pkg/colortbl
[12] http://tug.ctan.org/pkg/supertabular
[13] http://tug.ctan.org/pkg/longtable
[14] http://tug.ctan.org/pkg/xtab
[15] http://tug.ctan.org/pkg/tabulary
[16] http://tug.ctan.org/pkg/arydshln
[17] http://tug.ctan.org/pkg/ctable
[18] http://tug.ctan.org/pkg/slashbox
[19] http://tug.ctan.org/pkg/dcolumn
[20] http://tug.ctan.org/pkg/rccol
[21] http://www.ctan.org/pkg/spreadtab

Bibliography Management

For any academic/research writing, incorporating references into a document is an important task. Fortunately, LaTeX has a variety of features that make dealing with references much simpler, including built-in support for citing references. However, a much more powerful and flexible solution is achieved thanks to an auxiliary tool called BibTeX [3] (which comes bundled as standard with LaTeX).

BibTeX provides for the storage of all references in an external, flat-file database. This database can be linked to any LaTeX document, and citations made to any reference that is contained within the file. This is often more convenient than embedding them at the end of every document written. There is now a centralized bibliography source that can be linked to as many documents as desired (write once, read many!). Of course, bibliographies can be split over as many files as one wishes, so there can be a file containing references concerning *General Relativity* and another about *Quantum Mechanics.* When writing about *Quantum Gravity* (QG), which tries to bridge the gap between these two theories, both of these files can be linked into the document, in addition to references specific to QG.

Embedded system

If you are writing only one or two documents and aren't planning on writing more on the same subject for a long time, maybe you don't want to waste time creating a database of references you are never going to use. In this case you should consider using the basic and simple bibliography support that is embedded within LaTeX.

LaTeX provides an environment called `thebibliography` that you have to use where you want the bibliography; that usually means at the very end of your document, just before the `\end{document}` command. Here is a practical example:

```
\begin{thebibliography}{9} \bibitem{lamport94} Leslie Lamport, \emph{\LaTeX: A Document Preparation System}. Addison Wesley, Massachusetts, 2nd Edition, 1994. \end{thebibliography}
```

OK, so what is going on here? The first thing to notice is the establishment of the environment. `thebibliography` is a keyword that LaTeX recognizes as everything between the begin and end tags as being data for the bibliography. The optional argument, which I supplied after the begin statement, is telling LaTeX how wide the item label will be when printed. Note however, that it is not a literal parameter, i.e the number 9 in this case, but a text width. Therefore, I am effectively telling LaTeX that I will only need reference labels of one character in width, which means no more than nine references in total. If you want more than ten, then input a two-digit number, such as '99' which permits fewer than 100 references.

Next is the actual reference entry itself. This is prefixed with the `\bibitem{`*`cite_key`*`}` command. The *cite_key* should be a unique identifier for that particular reference, and is often some sort of mnemonic consisting of any sequence of letters, numbers and punctuation symbols (although not a comma). I often use the surname of the first author, followed by the last two digits of the year (hence *lamport94*). If that author has produced more than one reference for a given year, then I add letters after, 'a', 'b', etc. But, you should do whatever works for you. Everything after the key is the reference itself. You need to type it as you want it to be presented. I have put the different parts of the reference, such as author, title, etc., on different lines for readability. These linebreaks are ignored by LaTeX. I wanted the title to be in italics, so I used the `\emph{}` command to achieve this.

Citations

To actually cite a given document is very easy. Go to the point where you want the citation to appear, and use the following: `\cite{cite_key}`, where the *cite_key* is that of the bibitem you wish to cite. When LaTeX processes the document, the citation will be cross-referenced with the bibitems and replaced with the appropriate number citation. The advantage here, once again, is that LaTeX looks after the numbering for you. If it were totally manual, then adding or removing a reference would be a real chore, as you would have to re-number all the citations by hand.

```
Instead of WYSIWYG editors, typesetting systems like \TeX{} or \LaTeX{}
\cite{lamport94} can be used.
```

Referring More Specific

Sometimes you want to refer to a certain page, figure or theorem in a text book. For that you can use the arguments to the `\cite` command:

```
\cite[p.~215]{citation01}
```

The argument, "p. 215", will show up inside the same brackets. Note the tilde in [p.~215], which replaces the end-of-sentence spacing with a non-breakable inter-word space. There are two reasons: end-of-sentence spacing is too wide, and "p." should not be separated from the page number.

Multiple Citations

When a sequence of multiple citations are needed, you should use a single `\cite{}` command. The citations are then separated by commas. Note that you must not use spaces between the citations. Here's an example:

```
\cite{citation01,citation02,citation03}
```

The result will then be shown as citations inside the same brackets.

No Cite

If you only want a reference to appear in the bibliography, but not where it is referenced in the main text, then the `\nocite{}` command can be used, for example:

```
Lamport showed in 1995 something... \nocite{lamport95}.
```

A special version of the command, `\nocite{*}`, includes all entries from the database, whether they are referenced in the document or not.

Natbib

Using the standard LaTeX bibliography support, you will see that each reference is numbered and each citation corresponds to the numbers. The numeric style of citation is quite common in scientific writing. In other disciplines, the author-year style, e.g., (Roberts, 2003), such as *Harvard* is preferred, and is in fact becoming increasingly common within scientific publications. A discussion about which is best will not occur here, but a possible way to get such an output is by the `natbib` package. In fact, it can supersede LaTeX's own citation commands, as Natbib allows the user to easily switch between Harvard or numeric.

The first job is to add the following to your preamble in order to get LaTeX to use the Natbib package:

```
\usepackage{natbib}
```

Natbib commands

Citation command	Output
`\citet{goossens93}`	Goossens et al. (1993)
`\citep{goossens93}`	(Goossens et al., 1993)
`\citet*{goossens93}`	Goossens, Mittlebach, and Samarin (1993)
`\citep*{goossens93}`	(Goossens, Mittlebach, and Samarin, 1993)
`\citeauthor{goossens93}`	Goossens et al.
`\citeauthor*{goossens93}`	Goossens, Mittlebach, and Samarin
`\citeyear{goossens93}`	1993
`\citeyearpar{goossens93}`	(1993)
`\citealt{goossens93}`	Goossens et al. 1993
`\citealp{goossens93}`	Goossens et al., 1993

Also, you need to change the bibliography style file to be used, so edit the appropriate line at the bottom of the file so that it reads: `\bibliographystyle{plainnat}`. Once done, it is basically a matter of altering the existing `\cite` commands to display the type of citation you want.

The main commands simply add a *t* for 'textual' or *p* for 'parenthesized', to the basic `\cite` command. You will also notice how Natbib by default will compress references with three or more authors to the more concise *1st surname et al* version. By adding an asterisk (*), you can override this default and list all authors associated with that citation. There are some other specialized commands that Natbib supports, listed in the table here. Keep in mind that for instance abbrvnat does not support \citet* and will automatically choose between all authors and et al..

The final area that I wish to cover about Natbib is customizing its citation style. There is a command called `\bibpunct` that can be used to override the defaults and change certain settings. For example, I have put the following in the preamble:

```
\bibpunct{(}{)}{;}{a}{,}{,}
```

The command requires six mandatory parameters.

Natbib-compatible styles

Style	Source	Description
plainnat	Provided	natbib-compatible version of plain
abbrvnat	Provided	natbib-compatible version of abbrv
unsrtnat	Provided	natbib-compatible version of unsrt
apsrev	REVTeX 4 home page [1]	natbib-compatible style for Physical Review journals
rmpaps	REVTeX 4 home page [1]	natbib-compatible style for Review of Modern Physics journals
IEEEtranN	TeX Catalogue entry [2]	natbib-compatible style for IEEE publications
achemso	TeX Catalogue entry [3]	natbib-compatible style for American Chemical Society journals
rsc	TeX Catalogue entry [4]	natbib-compatible style for Royal Society of Chemistry journals

1. The symbol for the opening bracket.
2. The symbol for the closing bracket.
3. The symbol that appears between multiple citations.

4. This argument takes a letter:
   - *n* - numerical style.
   - *s* - numerical superscript style.
   - *any other letter* - author-year style.
5. The punctuation to appear between the author and the year (in parenthetical case only).
6. The punctuation used between years, in multiple citations when there is a common author. e.g., (Chomsky 1956, 1957). If you want an extra space, then you need `{,~}`.

So as you can see, this package is quite flexible, especially as you can easily switch between different citation styles by changing a single parameter. Do have a look at the Natbib manual [5], it's a short document and you can learn even more about how to use it.

BibTeX

I have previously introduced the idea of embedding references at the end of the document, and then using the `\cite` command to cite them within the text. In this tutorial, I want to do a little better than this method, as it's not as flexible as it could be. Which is why I wish to concentrate on using BibTeX.

A BibTeX database is stored as a *.bib* file. It is a plain text file, and so can be viewed and edited easily. The structure of the file is also quite simple. An example of a BibTeX entry:

```
@article{greenwade93,
    author  = "George D. Greenwade",
    title   = "The {C}omprehensive {T}ex {A}rchive {N}etwork ({CTAN})",
    year    = "1993",
    journal = "TUGBoat",
    volume  = "14",
    number  = "3",
    pages   = "342--351"
}
```

Each entry begins with the declaration of the reference type, in the form of `@type`. BibTeX knows of practically all types you can think of, common ones are: *book*, *article*, and for papers presented at conferences, there is *inproceedings*. In this example, I have referred to an article within a journal.

After the type, you must have a left curly brace '`{`' to signify the beginning of the reference attributes. The first one follows immediately after the brace, which is the citation key. This key must be unique for all entries in your bibliography. It is this identifier that you will use within your document to cross-reference it to this entry. It is up to you as to how you wish to label each reference, but there is a loose standard in which you use the author's surname, followed by the year of publication. This is the scheme that I use in this tutorial.

Next, it should be clear that what follows are the relevant fields and data for that particular reference. The field names on the left are BibTeX keywords. They are followed by an equals sign (=) where the value for that field is then placed. BibTeX expects you to explicitly label the beginning and end of each value. I personally use quotation marks ("), however, you also have the option of using curly braces ('{', '}'). But as you will soon see, curly braces have other roles, within attributes, so I prefer not to use them for this job as they can get more confusing. A notable exception is when you want to use characters with umlauts (ü, ö, etc), since their notation is in the format `\"{o}`, and the quotation mark will close the one opening the field, causing an error in the parsing of the reference.

Remember that each attribute must be followed by a comma to delimit one from another. You do not need to add a comma to the last attribute, since the closing brace will tell BibTeX that there are no more attributes for this entry, although you won't get an error if you do.

It can take a while to learn what the reference types are, and what fields each type has available (and which ones are required or optional, etc). So, look at this entry type reference [6] and also this field reference [7] for descriptions of all the fields. It may be worth bookmarking or printing these pages so that they are easily at hand when you need them.

Authors

BibTeX can be quite clever with names of authors. It can accept names in *forename surname* or *surname, forename*. I personally use the former, but remember that the order you input them (or any data within an entry for that matter) is customizable and so you can get BibTeX to manipulate the input and then output it however you like. If you use the *forename surname* method, then you must be careful with a few special names, where there are compound surnames, for example "John von Neumann". In this form, BibTeX assumes that the last word is the surname, and everything before is the forename, plus any middle names. You must therefore manually tell BibTeX to keep the 'von' and 'Neumann' together. This is achieved easily using curly braces. So the final result would be "John {von Neumann}". This is easily avoided with the *surname, forename*, since you have a comma to separate the surname from the forename.

Secondly, there is the issue of how to tell BibTeX when a reference has more than one author. This is very simply done by putting the keyword *and* in between every author. As we can see from another example:

```
@book{goossens93,
    author    = "Michel Goossens and Frank Mittlebach and Alexander Samarin",
    title     = "The LaTeX Companion",
    year      = "1993",
    publisher = "Addison-Wesley",
    address   = "Reading, Massachusetts"
}
```

This book has three authors, and each is separated as described. Of course, when BibTeX processes and outputs this, there will only be an 'and' between the penultimate and last authors, but within the .bib file, it needs the *and*'s so that it can keep track of the individual authors.

Standard templates

@article

An article from a magazine or a journal.

- Required fields: author, title, journal, year.
- Optional fields: volume, number, pages, month, note.

```
@article{Xarticle,
    author    = "",
    title     = "",
    journal   = "",
    %volume   = "",
    %number   = "",
    %pages    = "",
    year      = "XXXX",
    %month    = "",
    %note     = "",
}
```

@book

A published book

- Required fields: author/editor, title, publisher, year.
- Optional fields: volume/number, series, address, edition, month, note.

```
@book{Xbook,
    author    = "",
    title     = "",
    publisher = "",
    %volume   = "",
    %number   = "",
    %series   = "",
    %address  = "",
    %edition  = "",
    year      = "XXXX",
    %month    = "",
    %note     = "",
}
```

@booklet

A bound work without a named publisher or sponsor.

- Required fields: title.
- Optional fields: author, howpublished, address, month, year, note.

```
@booklet{Xbooklet,
    %author   = "",
    title     = "",
    %howpublished   = "",
    %address  = "",
    year      = "XXXX",
    %month    = "",
    %note     = "",
}
```

@conference

Equal to inproceedings

- Required fields: author, title, booktitle, year.
- Optional fields: editor, volume/number, series, pages, address, month, organization, publisher, note.

```
@conference{Xconference,
    author    = "",
    title     = "",
    booktitle = "",
    %editor   = "",
    %volume   = "",
    %number   = "",
    %series   = "",
    %pages    = "",
    %address  = "",
```

```
    year      = "XXXX",
    %month    = "",
    %publisher= "",
    %note     = "",
}
```

@inbook

A section of a book *without* its own title.

- Required fields: author/editor, booktitle, chapter and/or pages, publisher, year.
- Optional fields: volume/number, series, type, address, edition, month, note.

@incollection

A section of a book having its own title.

- Required fields: author, title, booktitle, publisher, year.
- Optional fields: editor, volume/number, series, type, chapter, pages, address, edition, month, note.

@inproceedings

An article in a conference proceedings.

- Required fields: author, title, booktitle, year.
- Optional fields: editor, volume/number, series, pages, address, month, organization, publisher, note.

@manual

Technical manual

- Required fields: title.
- Optional fields: author, organization, address, edition, month, year, note.

@mastersthesis

Master's thesis

- Required fields: author, title, school, year.
- Optional fields: type (eg. "diploma thesis"), address, month, note.

```
@mastersthesis{Xthesis,
    author    = "",
    title     = "",
    school    = "",
    %type     = "diploma thesis",
    %address  = "",
    year      = "XXXX",
    %month    = "",
    %note     = "",
}
```

@misc

Template useful for other kinds of publication

- Required fields: none
- Optional fields: author, title, howpublished, month, year, note.

```
@misc{Xmisc,
    %author    = "",
    %title     = "",
```

```
    %howpublished = "",
    %year     = "XXXX",
    %month    = "",
    %note     = "",
}
```

@phdthesis

Ph.D. thesis

- Required fields: author, title, year, school.
- Optional fields: address, month, keywords, note.

@proceedings

The proceedings of a conference.

- Required fields: title, year.
- Optional fields: editor, volume/number, series, address, month, organization, publisher, note.

@techreport

Technical report from educational, commercial or standardization institution.

- Required fields: author, title, institution, year.
- Optional fields: type, number, address, month, note.

```
@techreport{Xtreport,
    author    = "",
    title     = "",
    institution = "",
    %type     = "",
    %number   = "",
    %address  = "",
    year      = "XXXX",
    %month    = "",
    %note     = "",
}
```

@unpublished

An unpublished article, book, thesis, etc.

- Required fields: author, title, note.
- Optional fields: month, year.

Not standard templates

@patent

@collection

@electronic

Preserving capital letters

In the event that BibTeX has been set to not preserve all capitalization within titles, problems can occur, especially if you are referring to proper nouns, or acronyms. To tell BibTeX to keep them, use the good old curly braces around the letter in question, (or letters, if it's an acronym) and all will be well!

```
title = "The {LaTeX} Companion",
```

However, **avoid** putting the whole title in curly braces, as it will look odd if different capitalization format is used:

```
title = "{The LaTeX Companion}",
```

A few additional examples

Below you will find a few additional examples of bibliography entries. The first one covers the case of multiple authors in the Surname, Firstname format, and the second one deals with the incollection case.

```
@article{AbedonHymanThomas2003,
  author = "Abedon, S. T. and Hyman, P. and Thomas, C.",
  year = "2003",
  title = "Experimental examination of bacteriophage latent-period evolution as a response to bacterial availability",
  journal = "Applied and Environmental Microbiology",
  volume = "69",
  pages = "7499--7506"
}

@incollection{Abedon1994,
  author = "Abedon, S. T.",
  title = "Lysis and the interaction between free phages and infected cells",
  pages = "397--405",
  booktitle = "Molecular biology of bacteriophage T4",
  editor = "Karam, Jim D. Karam and Drake, John W. and Kreuzer, Kenneth N. and Mosig, Gisela
            and Hall, Dwight and Eiserling, Frederick A. and Black, Lindsay W. and Kutter, Elizabeth
            and Carlson, Karin and Miller, Eric S. and Spicer, Eleanor",
  publisher = "ASM Press, Washington DC",
  year = "1994"
}
```

If you have to cite a website you can use @misc, for example:

```
@misc{website:fermentas-lambda,
      author = "Fermentas Inc.",
      title = "Phage Lambda: description \& restriction map",
      month = "November",
      year = 2008,
      url = "http://www.fermentas.com/techinfo/nucleicacids/maplambda.htm"
}
```

The note field comes in handy if you need to add unstructured information, for example that the corresponding issue of the journal has yet to appear:

```
@article{blackholes,
      author="Bunny, R.",
      title="Black Holes and Their Relation to Hiding Eggs",
      journal="Theoretical Easter Physics",
      publisher="Eggs Ltd.",
      year="2010",
      note="(to appear)"
```

```
}
```

Getting current LaTeX document to use your .bib file

At the end of your LaTeX file (that is, after the content, but before `\end{document}`, you need to place the following commands:

```
\bibliographystyle{plain}  \bibliography{sample1,sample2,...,samplen}  % Note the lack of whitespace between the commas and the next bib file.
```

Bibliography styles are files recognized by BibTeX that tell it how to format the information stored in the `.bib` file when processed for output. And so the first command listed above is declaring which style file to use. The style file in this instance is `plain.bst` (which comes as standard with BibTeX). You do not need to add the .bst extension when using this command, as it is assumed. Despite its name, the plain style does a pretty good job (look at the output of this tutorial to see what I mean).

The second command is the one that actually specifies the `.bib` file you wish to use. The ones I created for this tutorial were called `sample1.bib`, `sample2.bib`, . . ., `samplen.bib`, but once again, you don't include the file extension. At the moment, the `.bib` file is in the same directory as the LaTeX document too. However, if your .bib file was elsewhere (which makes sense if you intend to maintain a centralized database of references for all your research), you need to specify the path as well, e.g `\bibliography{/some/where/sample}`.

Now that LaTeX and BibTeX know where to look for the appropriate files, actually citing the references is fairly trivial. The `\cite{`*`ref_key`*`}` is the command you need, making sure that the *ref_key* corresponds exactly to one of the entries in the .bib file. If you wish to cite more that one reference at the same time, do the following: `\cite{`*`ref_key1, ref_key2, ..., ref_keyN`*`}`.

Why won't LaTeX generate any output?

The addition of BibTeX adds extra complexity for the processing of the source to the desired output. This is largely hidden to the user, but because of all the complexity of the referencing of citations from your source LaTeX file to the database entries in another file, you actually need multiple passes to accomplish the task. This means you have to run LaTeX a number of times, where each pass, it will perform a particular task until it has managed to resolve all the citation references. Here's what you need to type (into command line):

1. `latex latex_source_code.tex`
2. `bibtex latex_source_code.aux`
3. `latex latex_source_code.tex`
4. `latex latex_source_code.tex`

(Extensions are optional, if you put them note that the bibtex command takes the AUX file as input.)

After the first LaTeX run, you will see errors such as:

```
LaTeX Warning: Citation `lamport94' on page 1 undefined on input line 21.
...
LaTeX Warning: There were undefined references.
```

The next step is to run bibtex on that same LaTeX source (and not on the actual .bib file) to then define all the references within that document. You should see output like the following:

```
This is BibTeX, Version 0.99c (Web2C 7.3.1)
The top-level auxiliary file: latex_source_code.aux
The style file: plain.bst
Database file #1: sample.bib
```

The third step, which is invoking LaTeX for the second time will see more errors like "`LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.`". Don't be alarmed, it's almost complete. As you can guess, all you have to do is follow its instructions, and run LaTeX for the third time, and the document will be output as expected, without further problems.

If you want a pdf output instead of a dvi output you can use `pdflatex` instead of `latex` as follows:

1. `pdflatex latex_source_code.tex`
2. `bibtex latex_source_code.aux`
3. `pdflatex latex_source_code.tex`
4. `pdflatex latex_source_code.tex`

(Extensions are optional, if you put them note that the bibtex command takes the AUX file as input.)

Note that if you are editing your source in vim and attempt to use command mode and the current file shortcut (%) to process the document like this:

1. `:! pdflatex %`
2. `:! bibtex %`

You will get an error similar to this:

1. `I couldn't open file name 'current_file.tex.aux'`

It appears that the file extension is included by default when the current file command (%) is executed. To process your document from within vim, you must explicitly name the file without the file extension for bibtex to work, as is shown below:

1. `:! pdflatex %`
2. `:! bibtex latex_source_code` (without file extension, it looks for the AUX file as mentioned above)
3. `:! pdflatex %`
4. `:! pdflatex %`

However, it is much easier to install the Vim-LaTeX plugin from here [8]. This allows you to simply type \ll when not in insert mode, and all the appropriate commands are automatically executed to compile the document. Vim-LaTeX even detects how many times it has to run pdflatex, and whether or not it has to run bibtex. This is just one of the many nice features of Vim-LaTeX, you can read the excellent Beginner's Tutorial [9] for more about the many clever shortcuts Vim-LaTeX provides.

Another option exists if you are running Unix/Linux or any other platform where you have make [10]. Then you can simply create a Makefile and use vim's make command or use make in shell. The Makefile would then look like this:

```
latex_source_code.pdf: latex_source_code.tex latex_source_code.bib
    pdflatex latex_source_code.tex
    bibtex latex_source_code
    pdflatex latex_source_code.tex
    pdflatex latex_source_code.tex
```

Bibliography styles

Below you can see three styles available with LaTeX:

Instead of WYSIWYG editors, typesetting systems like TeX[1] or LaTeX [2] can be used.

References

[1] Paul W. Abrahams, Kathryn A. Hargreaves, and Karl Berry. *TEXfor the Impatient*, 2003.

[2] Leslie Lamport. *LATEX: A Document Preparation System*. Addison Wesley, second edition, 1994.

plain

Instead of WYSIWYG editors, typesetting systems like TeX[1] or LaTeX [2] can be used.

References

[1] P. W. Abrahams, K. A. Hargreaves, and K. Berry. *TEXfor the Impatient*, 2003.

[2] L. Lamport. *LATEX: A Document Preparation System*. Addison Wesley, second edition, 1994.

abbrv

Instead of WYSIWYG editors, typesetting systems like TeX[AHB03] or LaTeX [Lam94] can be used.

References

[AHB03] Paul W. Abrahams, Kathryn A. Hargreaves, and Karl Berry. *TEXfor the Impatient*, 2003.

[Lam94] Leslie Lamport. *LATEX: A Document Preparation System*. Addison Wesley, second edition, 1994.

alpha

To number the references in order of appearance, rather than alphabetical order use ieeetr

```
\bibliographystyle{ieeetr}
```

Web page http://www.cs.stir.ac.uk/~kjt/software/latex/showbst.html contains more examples.

Including URLs in bibliography

As you can see, there is no field for URLs. One possibility is to include Internet addresses in `howpublished` field of `@misc` or `note` field of `@techreport`, `@article`, `@book`:

```
howpublished = "\url{http://www.example.com}"
```

Note the usage of `\url` command to ensure proper appearance of URLs.

Another way is to use special field `url` and make bibliography style recognise it.

```
 url = "http://www.example.com"
```

You need to use `\usepackage{url}` in the first case or `\usepackage{hyperref}` in the second case.

Styles provided by Natbib (see below) handle this field, other styles can be modified using urlbst [11] program. Modifications of three standard styles (plain, abbrv and alpha) are provided with urlbst.

If you need more help about URLs in bibliography, visit FAQ of UK List of TeX [12].

Customizing bibliography appearance

One of the main advantages of BibTeX, especially for people who write many research papers, is the ability to customize your bibliography to suit the requirements of a given publication. You will notice how different publications tend to have their own style of formatting references, to which authors must adhere if they want their manuscripts published. In fact, established journals and conference organizers often will have created their own bibliography style (.bst file) for those users of BibTeX, to do all the hard work for you.

It can achieve this because of the nature of the .bib database, where all the information about your references is stored in a structured format, but nothing about style. This is a common theme in LaTeX in general, where it tries as much as possible to keep content and presentation separate.

A bibliography style file (`.bst`) will tell LaTeX how to format each attribute, what order to put them in, what punctuation to use in between particular attributes etc. Unfortunately, creating such a style by hand is not a trivial task. Which is why `Makebst` (also known as *custom-bib*) is the tool we need.

`Makebst` can be used to automatically generate a .bst file based on your needs. It is very simple, and actually asks you a series of questions about your preferences. Once complete, it will then output the appropriate style file for you to use.

It should be installed with the LaTeX distribution (otherwise, you can download it [13]) and it's very simple to initiate. At the command line, type:

```
latex makebst
```

LaTeX will find the relevant file and the questioning process will begin. You will have to answer quite a few (although, note that the default answers are pretty sensible), which means it would be impractical to go through an example in this tutorial. However, it is fairly straight-forward. And if you require further guidance, then there is a comprehensive manual [14] available. I'd recommend experimenting with it and seeing what the results are when applied to a LaTeX document.

If you are using a custom built .bst file, it is important that LaTeX can find it! So, make sure it's in the same directory as the LaTeX source file, *unless* you are using one of the standard style files (such as *plain* or *plainnat*, that come bundled with LaTeX - these will be automatically found in the directories that they are installed. Also, make sure the name of the `.bst` file you want to use is reflected in the `\bibliographystyle{style}` command (but don't include the `.bst` extension!).

Localizing bibliography appearance

When writing documents in languages other than English, you may find it desirable to adapt the appearance of your bibliography to the document language. This concerns words such as *editors*, *and*, or *in* as well as a proper typographic layout. The `babelbib` package [15] can be used here. For example, to layout the bibliography in German, add the following to the header:

```
\usepackage[fixlanguage]{babelbib} \selectbiblanguage{german}
```

Alternatively, you can layout each bibliography entry according to the language of the cited document:

```
\usepackage{babelbib}
```

The language of an entry is specified as an additional field in the BibTeX entry:

```
@article{mueller08,
  % ...
  language = {german}
}
```

For `babelbib` to take effect, a bibliography style supported by it - one of `babplain`, `babplai3`, `babalpha`, `babunsrt`, `bababbrv`, and `bababbr3` - must be used:

```
\bibliographystyle{babplain} \bibliography{sample}
```

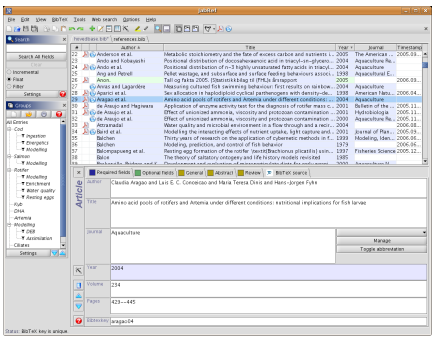
Getting Bibliographic data

Many online databases provide bibliographic data in BibTeX-Format, making it easy to build your own database. For example, Google Scholar [16] offers the option to return properly formatted output, but you must turn it on in the Preferences [17].

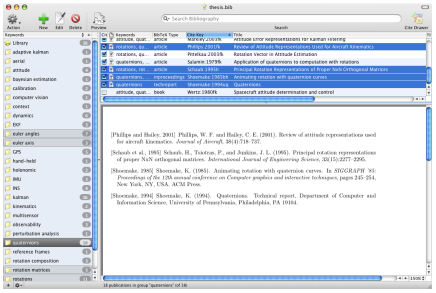
One should be alert to the fact that bibliographic databases are frequently the product of several generations of automatic processing, and so the resulting BibTex code is prone to a variety of minor errors, especially in older entries.

Helpful Tools

- Mendeley [18] Mendeley is cost-free academic software for managing PDFs which can manage a bibliography in Open Office and read BibTeX.
- Zotero [19] Zotero is a free and open reference manager working as a Firefox plugin capable of importing and exporting bib files.
- JabRef [20] is a Java program (under the GPL license) which lets you search many bibliographic databases such as Medline, Citeseer, IEEEXplore and arXiv and feed and manage your BibTeX local databases with your selected articles. Based on BiBTeX, JabRef can export in many other output formats such as html, MS Word or EndNote. It can be used online without being installed (http://jabref.sourceforge.net/jws/jabref.jnlp)
- Referencer [21] Referencer is a Gnome application to organise documents or references, and ultimately generate a BibTeX bibliography file.



JabRef



BibDesk

- Citavi [22] Commercial software (with size-limited free demo version) which even searches libraries for citations and keeps all your knowledge in a database. Export of the database to all kinds of formats is possible. Works together with MS Word and Open Office Writer. Moreover plug ins for browsers and Acrobat Reader exist to automatically include references to your project.
- bibliographer [23](broken link) Bibliographer is a BibTeX bibliography database editor which aims to be easy to use. Its features include linking files to your records with indexing and searching support. The interface is designed for the easy navigation of your bibliography, and double clicking a record will open the linked file.
- cb2Bib [24] The cb2Bib is a tool for rapidly extracting unformatted, or unstandardized bibliographic references from email alerts, journal Web pages, and PDF files.
- KBibTeX [25] KBibTeX is a BibTeX editor for KDE to edit bibliographies used with LaTeX. Features include comfortable input masks, starting web queries (e. g. Google or PubMed) and exporting to PDF, PostScript, RTF and XML/HTML. As KBibTeX is using KDE's KParts technology, KBibTeX can be embedded into Kile or Konqueror.
- KBib [26] Another BibTeX editor for KDE. It has similar capabilities, and slightly different UI. Features include BibTeX reference generation from PDF files, plain text, DOI, arXiv & PubMed IDs. Web queries to Google Scholar, PubMer, arXiv and a number of other services are also supported.
- Bibwiki [27] Bibwiki is a Specialpage for MediaWiki to manage BibTeX bibliographies. It offers a straightforward way to import and export bibliographic records.
- BibDesk [28] BibDesk is a bibliographic reference manager for Mac OS X. It features a very usable user interface and provides a number of features like smart folders based on keywords and live tex display.
- CiteULike [29] CiteULike is a free online service to organise academic papers. It can export citations in BibTeX format, and can "scrape" BibTeX data from many popular websites.
- Bibtex [30] Bibtex is a DokuWiki plugin that allows for the inclusion of bibtex formatted citations in DokuWiki pages and displays them in APA format. Note: This Plugins is vulnerable to an XSS attack -> http://www.dokuwiki.org/plugin:bibtex
- BibSonomy [31] — A free social bookmark and publication management system based on BibTeX.
- Synapsen [32] — Hypertextual Card Index / Reference Manager with special support for BiBTeX / biblatex, written in Java.

Summary

Although it can take a little time to get to grips with BibTeX, in the long term, it's an efficient way to handle your references. It's not uncommon to find .bib files on websites that people compile as a list of their own publications, or a survey of relevant works within a given topic, etc. Or in those huge, online bibliography databases, you often find BibTeX versions of publications, so it's a quick cut-and-paste into your own .bib file, and then no more hassle!

Having all your references in one place can be a big advantage. And having them in a structured form, that allows customizable output is another one. There are a variety of free utilities that can load your .bib files, and allow you to view them in a more efficient manner, as well as sort them and check for errors.

Bibliography in the table of contents

If you are writing a *book* or *report*, you'll likely insert your bibliography using something like:

```
\begin{thebibliography}{99} \bibitem{bib:one_book} some information
\bibitem{bib:one_article} other information \end{thebibliography}
```

This will create a chapter-like output showing properly all your references. Even though it looks like a chapter, it will not be handled like that so it will not appear on the Table of Contents at the beginning of the document. If you want your bibliography to be in the table of contents, just add the following two lines just before the *thebibliography* environment:

```
\clearpage \addcontentsline{toc}{chapter}{Bibliography}
```

(OR `\addcontentsline{toc}{section}{Bibliography}` if you're writing an *article*)

The first line just terminates the current paragraph and page. If you are writing a *book*, use `\cleardoublepage` to match the style used. The second line will add a line in the Table of Contents (first option, *toc*), it will be like the ones created by chapters (second option, *chapter*), and the third argument will be printed on the corresponding line in the Table of Contents; here *Bibliography* was chosen because it's the same text the *thebibliography* environment will automatically write when you use it, but you are free to write whatever you like. If you are using separate bib file, add these lines between `\bibliographystyle` and `\bibliography`.

If you use hyperref package, you should also use `\phantomsection` command to enable hyperlinking from the table of contents to bibliography.

```
\cleardoublepage \phantomsection \addcontentsline{toc}{chapter}{Bibliography}
```

This trick is particularly useful when you have to insert the bibliography in the Table of Contents, but it can work for anything. When LaTeX finds the code above, it will record the info as described and the current page number, inserting a new line in the Contents page.

Add the Bibliography to the Table of Contents as numbered item

If you instead want bibliography to be numbered section or chapter, you'll likely use this way:

```
\cleardoublepage % This is needed if the book class is used, to place
the anchor in the correct page,
                 % because the bibliography will start on its own page.
                 % Use \clearpage instead if the document class uses
the "oneside" argument
\renewcommand*{\refname}{} % This will define heading of bibliography
to be empty, so you can...
\section{Bibliography}     % ...place a normal section heading before
the bibliography entries.

\begin{thebibliography}{99}
...
\end{thebibliography}
```

Another even easier solution is to use `\section` inside of the `\renewcommand` block:

```
\renewcommand{\refname}{\section{Sources}} % Using "Sources" as the
title of the section
\begin{thebibliography}{99}
...
\end{thebibliography}
```

You may wish to use `\renewcommand*{\refname}{\vspace*{-12mm}}` to counteract the extra space the blank `\refname` inserts.

- Note: Use `\bibname` instead of `\refname` if you use the report class.

This page uses material from Andy Roberts' Getting to grips with Latex [3] *with permission from the author.*

References

[1] http://authors.aps.org/revtex4/
[2] http://www.ctan.org/tex-archive/help/Catalogue/entries/ieeetran.html
[3] http://www.ctan.org/tex-archive/help/Catalogue/entries/achemso.html
[4] http://www.ctan.org/tex-archive/help/Catalogue/entries/rsc.html
[5] http://www.ctex.org/documents/packages/bibref/natbib.pdf
[6] http://newton.ex.ac.uk/tex/pack/bibtex/btxdoc/node6.html
[7] http://newton.ex.ac.uk/tex/pack/bibtex/btxdoc/node7.html
[8] http://vim-latex.sourceforge.net/
[9] http://vim-latex.sourceforge.net/documentation/latex-suite-quickstart/
[10] http://en.wikipedia.org/wiki/Make_%28software%29
[11] http://purl.org/nxg/dist/urlbst
[12] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=citeURL
[13] http://www.mps.mpg.de/software/latex/localtex/localltx.html#makebst
[14] http://www.mps.mpg.de/software/latex/localtex/doc/merlin.pdf
[15] http://tug.ctan.org/tex-archive/biblio/bibtex/contrib/babelbib/
[16] http://scholar.google.com
[17] http://scholar.google.de/scholar_preferences?hl=en&lr=&output=search
[18] http://mendeley.com
[19] http://www.zotero.org/
[20] http://jabref.sourceforge.net/
[21] http://icculus.org/referencer/index.html
[22] http://www.citavi.ch
[23] http://bibliographer.homelinux.net/
[24] http://www.molspaces.com/cb2bib/
[25] http://www.unix-ag.uni-kl.de/~fischer/kbibtex/
[26] http://users.tpg.com.au/thachly/kbib/
[27] http://www.plaschg.net/bibwiki/
[28] http://bibdesk.sourceforge.net/
[29] http://www.citeulike.org/
[30] http://stat.genopole.cnrs.fr/~cambroise/doku.php?id=softwares:dokuwikibibtexplugin
[31] http://www.bibsonomy.org/
[32] http://www.verzetteln.de/synapsen/

Mathematics

Mathematics

One of the greatest motivating forces for Donald Knuth when he began developing the original TeX system was to create something that allowed simple construction of mathematical formulas, whilst looking professional when printed. The fact that he succeeded was most probably why TeX (and later on, LaTeX) became so popular within the scientific community. Typesetting mathematics is one of LaTeX's greatest strengths. It is also a large topic due to the existence of so much mathematical notation.

If your document requires only a few simple mathematical formulas, plain LaTeX has most of the tools that you will need. If you are writing a scientific document that contains numerous complicated formulas, the amsmath package[1] introduces several new commands that are more powerful and flexible than the ones provided by LaTeX. The mathtools package fixes some amsmath quirks and adds some useful settings, symbols, and environments to amsmath.[2] To use either package, include:

```
\usepackage{amsmath}
```

or

```
\usepackage{mathtools}
```

in the preamble of the document.

Mathematics environments

LaTeX needs to know beforehand that the subsequent text does in fact contain mathematical elements. This is because LaTeX typesets maths notation differently than normal text. Therefore, special environments have been declared for this purpose. They can be distinguished into two categories depending on how they are presented:

- *text* - text formulas are displayed in-line, that is, within the body of text where it is declared. e.g., I can say that $a + a = 2a$ within this sentence.
- *displayed* - displayed formulas are separate from the main text.

As maths require special environments, there are naturally the appropriate environment names you can use in the standard way. Unlike most other environments, however, there are some handy shorthands to declaring your formulas. The following table summarizes them:

Type	Environment	LaTeX shorthand	TeX shorthand
Text	`\begin{math}...\end{math}`	`\(...\)`	`$...$`
Displayed	`\begin{displaymath}...\end{displaymath}` or `\begin{equation*}...\end{equation*}` [3]	`\[...\]`	`$$...$$`

Note: Using the `$$...$$` should be avoided, as it may cause problems, particularly with the AMS-LaTeX macros. Furthermore, should a problem occur, the error messages may not be helpful.

Additionally, there is a second possible environment for the *displayed* type of formulas: equation. The difference between this and displaymath is that equation also adds sequential equation numbers by the side.

If you are typing text normally, you are said to be in *text mode*, while you are typing within one of those mathematical environments, you are said to be in *math mode*, that has some differences compared to the *text mode*:

1. Most spaces and line breaks do not have any significance, as all spaces are either derived logically from the mathematical expressions, or have to be specified with special commands such as **\quad**
2. Empty lines are not allowed. Only one paragraph per formula.
3. Each letter is considered to be the name of a variable and will be typeset as such. If you want to typeset normal text within a formula (normal upright font and normal spacing) then you have to enter the text using dedicated commands.

Inserting "Displayed" maths inside blocks of text

In order for some operators, such as **\lim** or **\sum** to be displayed correctly inside some math environments (read `$.....$`), it might be convenient to write the **\displaystyle** class inside the environment. Doing so might cause the line to be taller, but will cause exponents and indices to be displayed correctly for some math operators. For example, the $\sum$ will print a smaller Σ and $\displaystyle \sum$ will print a bigger one $\displaystyle \sum$, like in equation

(This only works with AMSMATH package).

Symbols

Mathematics has lots and lots of symbols! If there is one aspect of maths that is difficult in LaTeX it is trying to remember how to produce them. There are of course a set of symbols that can be accessed directly from the keyboard:

```
+ - = ! / ( ) [ ] < > | ' :
```

Beyond those listed above, distinct commands must be issued in order to display the desired symbols. And there are *a lot!* of Greek letters, set and relations symbols, arrows, binary operators, etc. For example:

```
\[ \forall x \in X, \quad \exists y \leq \epsilon \]
```

$$\forall x \in X, \quad \exists y \leq \epsilon$$

Fortunately, there's a tool that can greatly simplify the search for the command for a specific symbol. Look for "Detexify" in the external links section below. Another option would be to look in the "The Comprehensive LaTeX Symbol List" in the external links section below.

Greek letters

Greek letters are commonly used in mathematics, and they are very easy to type in *math mode*. You just have to type the name of the letter after a backslash: if the first letter is lowercase, you will get a lowercase Greek letter, if the first letter is uppercase (and only the first letter), then you will get an uppercase letter. Note that some uppercase Greek letters look like Latin ones, so they are not provided by LaTeX (e.g. uppercase *Alpha* and *Beta* are just "A" and "B" respectively). Lowercase epsilon, theta, phi, pi, rho, and sigma are provided in two different versions. The alternate, or *var*iant, version is created by adding "var" before the name of the letter:

```
\[ \alpha, \Alpha, \beta, \Beta, \gamma, \Gamma, \pi, \Pi,
\phi, \varphi, \Phi \]
```

$$\alpha, \mathrm{A}, \beta, \mathrm{B}, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \Phi$$

Scroll down to #List_of_Mathematical_Symbols for a complete list of Greek symbols.

Operators

An operator is a function that is written as a word: e.g. trigonometric functions (sin, cos, tan), logarithms and exponentials (log, exp). LaTeX has many of these defined as commands:

```
\[ \cos (2\theta) = \cos^2 \theta - \sin^2 \theta \]
```
$$\cos(2\theta) = \cos^2\theta - \sin^2\theta$$

For certain operators such as limits, the subscript is placed underneath the operator:

```
\[ \lim_{x \to \infty} \exp(-x) = 0 \]
```
$$\lim_{x \to \infty} \exp(-x) = 0$$

For the modular operator there are two commands: **\bmod** and **\pmod** :

```
\[ a \bmod b \]
```
$$a \bmod b$$

```
\[ x \equiv a \pmod b \]
```
$$x \equiv a \pmod b$$

To use operators which are not pre-defined, such as argmax, see custom operators

Powers and indices

Powers and indices are equivalent to superscripts and subscripts in normal text mode. The caret (^) character is used to raise something, and the underscore (_) is for lowering. If more than one expression is raised or lowered, they should be grouped using curly braces ({ and }).

```
\[ k_{n+1} = n^2 + k_n^2 - k_{n-1} \]
```
$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

An underscore (_) can be used with a vertical bar (|) to denote evaluation using subscript notation in mathematics:

```
\[ f(n) = n^5 + 4n^2 + 2 |_{n=17} \]
```
$$f(n) = n^5 + 4n^2 + 2|_{n=17}$$

Fractions and Binomials

A fraction is created using the **\frac**{numerator}{denominator } command. (for those who need their memories refreshed, that's the *top* and *bottom* respectively!). Likewise, the binomial coefficient (aka the Choose function) may be written using the **\binom** command[3] :

```
\[ \frac{n!}{k!(n-k)!} = \binom{n}{k} \]
```
$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

It is also possible to use the **\choose** command without the amsmath package:

```
\[ \frac{n!}{k!(n-k)!} = {n \choose k} \]
```
$$\frac{n!}{k!(n-k)!} = {n \choose k}$$

You can embed fractions within fractions:

```
\[ \frac{\frac{1}{x}+\frac{1}{y
```
Rendering missing!

{y-z} \] |render=
$$\frac{\frac{1}{x} + \frac{1}{y}}{y - z}$$
}}

Note that when appearing inside another fraction, or in inline text $\frac{a}{b}$, a fraction is noticeably smaller than in displayed mathematics. The **\tfrac** and **\dfrac** commands[3] force the use of the respective styles,

\textstyle and **\displaystyle** . Similarly, the **\tbinom** and **\dbinom** commands typeset the binomial coefficient.

Another way to write fractions is to use the **\over** command without the amsmath package:

```
\[ {n! \over k!(n-k)!} = {n \choose k} \]
```

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

For relatively simple fractions, it may be more aesthetically pleasing to use powers and indices:

```
\[ ^3/_7 \]
```

$$^3/_7$$

If you use them throughout the document, usage of xfrac package is recommended. This package provides **\sfrac** command to create slanted fractions. Usage:

```
Take \sfrac{1}{2} cup of sugar, \dots \[
3\times\sfrac{1}{2}=1\sfrac{1}{2} \] Take ${}^1/_2$
cup of sugar, \dots \[ 3\times{}^1/_2=1{}^1/_2 \]
```

Take ½ cup of sugar, …

$$3 \times {}^1/_2 = 1{}^1/_2$$

Take $^1/_2$ cup of sugar, …

$$3 \times {}^1/_2 = 1{}^1/_2$$

Alternatively, the nicefrac package provides the **\nicefrac** command, whose usage is similar to **\sfrac** .

Continued fractions

Continued fractions should be written using **\cfrac** command[3] :

```
\begin{equation} x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 +
\cfrac{1}{a_3 + a_4
```

Rendering missing!

} \end{equation} |render=

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + a_4}}}$$

Multiplication of two numbers

To make multiplication visually similar to a fraction, you can use nested array. Multiplication of numbers written one behalf other.

```
\begin{equation} \frac{ \begin{array}[b]{r} \left( x_1 x_2 \right)\\
\times \left( x'_1 x'_2 \right) \end{array} }{ \left( y_1y_2y_3y_4
\right) } \end{equation}
```

$$\frac{\begin{array}{r}(x_1x_2)\\ \times (x'_1x'_2)\end{array}}{(y_1y_2y_3y_4)}$$

Roots

The **\sqrt** command creates a square root surrounding an expression. It accepts an optional argument specified in square brackets ([and]) to change magnitude:

```
\[ \sqrt{\frac{a}{b
```

Rendering missing!

\] |render=

$$\sqrt{\frac{a}{b}}$$

```
\[ \sqrt[n]{1+x+x^2+x^3+\ldots} \]
```

$$\sqrt[n]{1+x+x^2+x^3+\ldots}$$

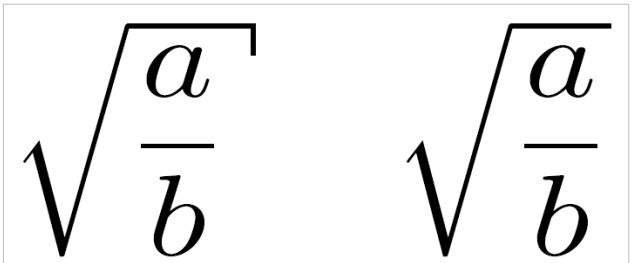
Some people prefer writing the square root "closing" it over its content. This method arguably makes it more clear just what is in the scope of the root sign. This habit is not normally used while writing with the computer, but if you still want to change the output of the square root, LaTeX gives you this possibility. Just add the following code in the preamble of your document:

```
% New definition of square root: % it renames \sqrt as \oldsqrt
\let\oldsqrt\sqrt % it defines the new \sqrt in terms of the old one
\def\sqrt{\mathpalette\DHLhksqrt} \def\DHLhksqrt#1#2{%
\setbox0=\hbox{$#1\oldsqrt{#2\,}$}\dimen0=\ht0 \advance\dimen0-0.2\ht0
\setbox2=\hbox{\vrule height\ht0 depth -\dimen0}% {\box0\lower0.4pt\box2
```

Rendering missing!

lrender=

}} This TeX code first renames the **\sqrt** command as **\oldsqrt** , then redefines **\sqrt** in terms of the old one, adding something more. The new square root can be seen in the picture on the right, compared to the old one. Unfortunately this code won't work if you want to use multiple roots: if you try to write $\sqrt[b]{a}$ as **\sqrt**[b]{a } after you used the code above, you'll just get a wrong output. In other words, you can redefine the square root this way only if you are not going to use multiple roots in the whole document.



The new style is on left, the old one on right

Sums and integrals

The **\sum** and **\int** commands insert the sum and integral symbols respectively, with limits specified using the caret (^) and underscore (_). The typical notation for sums is:

```
\[ \sum_{i=1}^{10} t_i \]
```

$$\sum_{i=1}^{10} t_i$$

The limits for the integrals follow the same notation. It's also important to represent the integration variables with an upright d, which in math mode is obtained through the \mathrm{} command, and with a small space separating it from the integrand, which is attained with the \, command.

```
\[ \int_0^\infty e^{-x}\,\mathrm{d}x \]
```

$$\int_0^\infty e^{-x}\,\mathrm{d}x$$

There are many other "big" commands which operate in a similar manner:

\sum	$\sum$	\prod	$\prod$	\coprod	$\coprod$
\bigoplus	$\bigoplus$	\bigotimes	$\bigotimes$	\bigodot	$\bigodot$
\bigcup	$\bigcup$	\bigcap	$\bigcap$	\biguplus	$\biguplus$
\bigsqcup	$\bigsqcup$	\bigvee	$\bigvee$	\bigwedge	$\bigwedge$
\int	$\int$	\oint	$\oint$	\iint[3]	$\iint$
\iiint[3]	$\iiint$	\iiiint[3]	$\iiiint$	\idotsint[3]	$\idotsint$

For more integral symbols, including those not included by default in the Computer Modern font, try the esint package.

The **\substack** command[3] allows the use of **** to write the limits over multiple lines:

```
\[ \sum_{\substack{ 0<i<m \\ 0<j<n
```

Rendering missing!

```
P(i,j)
```

\] |render= $\sum_{\substack{0<i<m \\ 0<j<n}} P(i,j)$ }}

If you want the limits of an integral to be specified above and below the symbol (like the sum), use the **\limits** command:

```
\[ \int\limits_a^b \]
```

$\int\limits_a^b$

However if you want this to apply to ALL integrals, it is preferable to specify the intlimits option when loading the amsmath package:

```
\usepackage[intlimits]{amsmath}
```

Subscripts and superscripts in other contexts as well as other parameters to amsmath package related to them are described in Advanced Mathematics chapter.

For bigger integrals, you may use personal declarations, or the bigints package [4] .

Brackets, braces and delimiters

How to use braces in multi line equations is described in the Advanced Mathematics chapter.

The use of delimiters such as brackets soon becomes important when dealing with anything but the most trivial equations. Without them, formulas can become ambiguous. Also, special types of mathematical structures, such as matrices, typically rely on delimiters to enclose them.

There are a variety of delimiters available for use in LaTeX:

```
\[ () \, [] \, \{\} \, || \, \|\| \, \langle\rangle \,
\lfloor\rfloor \, \lceil\rceil \]
```

$() \, [] \, \{\} \, || \, \|\| \, \langle\rangle \, \lfloor\rfloor \, \lceil\rceil$

Automatic sizing

Very often mathematical features will differ in size, in which case the delimiters surrounding the expression should vary accordingly. This can be done automatically using the **\left** and **\right** commands. Any of the previous delimiters may be used in combination with these:

```
\[ \left(\frac{x^2}{y^3}\right) \]
```

$$\left(\frac{x^2}{y^3}\right)$$

Curly braces are defined differently by using **\left\{** and **\right\}** ,

```
\[ \left\{\frac{x^2}{y^3}\right\} \]
```

$$\left\{\frac{x^2}{y^3}\right\}$$

If a delimiter on only one side of an expression is required, then an invisible delimiter on the other side may be denoted using a period (.).

```
\[ \left.\frac{x^3}{3}\right|_0^1 \]
```

$$\left.\frac{x^3}{3}\right|_0^1$$

Manual sizing

In certain cases, the sizing produced by the **\left** and **\right** commands may not be desirable, or you may simply want finer control over the delimiter sizes. In this case, the **\big** , **\Big** , **\bigg** and **\Bigg** modifier commands may be used:

```
\[ ( \big( \Big( \bigg( \Bigg( \]
```

$$(\big(\Big(\bigg(\Bigg($$

Matrices and arrays

A basic matrix may be created using the matrix environment[3] : in common with other table-like structures, entries are specified by row, with columns separated using an ampersand (`&`) and a new rows separated with a double backslash (****)

```
\[ \begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix} \]
```

$$\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$$

To specify alignment of columns in the table, use starred version[5] :

```
\[ \begin{matrix} -1 & 3 \\ 2 & -4 \end{matrix} =
\begin{matrix*}[r] -1 & 3 \\ 2 & -4 \end{matrix*} \]
```

$$\begin{matrix} -1 & 3 \\ 2 & -4 \end{matrix} = \begin{matrix*}[r] -1 & 3 \\ 2 & -4 \end{matrix*}$$

The alignment by default is c but it can be any column type valid in array environment.

However matrices are usually enclosed in delimiters of some kind, and while it is possible to use the `\left` and `\right` commands, there are various other predefined environments which automatically include delimiters:

Environment name	Surrounding delimiter	Notes
pmatrix[3]	( )	centers columns by default
pmatrix*[5]	( )	allows to specify alignment of columns in optional parameter
bmatrix[3]	[ ]	centers columns by default
bmatrix*[5]	[ ]	allows to specify alignment of columns in optional parameter
Bmatrix[3]	{ }	centers columns by default
Bmatrix*[5]	{ }	allows to specify alignment of columns in optional parameter
vmatrix[3]	| |	centers columns by default
vmatrix*[5]	| |	allows to specify alignment of columns in optional parameter
Vmatrix[3]	‖ ‖	centers columns by default
Vmatrix*[5]	‖ ‖	allows to specify alignment of colums in optional parameter

When writing down arbitrary sized matrices, it is common to use horizontal, vertical and diagonal triplets of dots (known as ellipses) to fill in certain columns and rows. These can be specified using the **\cdots** , **\vdots** and **\ddots** respectively:

```
\[ A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots
& a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2}
& \cdots & a_{m,n} \end{pmatrix} \]
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

In some cases you may want to have finer control of the alignment within each column, or want to insert lines between columns or rows. This can be achieved using the array environment, which is essentially a math-mode version of the `tabular` environment, which requires that the columns be pre-specified:

```
\[ \begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array} \]
```

$$\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}$$

You may see that the AMS matrix class of environments doesn't leave enough space when used together with fractions resulting in output similar to this:

$$M = \begin{bmatrix} \frac{5}{6} & \frac{1}{6} & 0 \\ \frac{5}{6} & 0 & \frac{1}{6} \\ 0 & \frac{5}{6} & \frac{1}{6} \end{bmatrix}$$

To counteract this problem, add additional leading space with the optional parameter to the **** command:

```
\[ M = \begin{bmatrix} \frac{5}{6} & \frac{1}{6} & 0 \\[0.3em]
\frac{5}{6} & 0 & \frac{1}{6} \\[0.3em] 0 & \frac{5}{6} &
\frac{1}{6} \end{bmatrix} \]
```

$$M = \begin{bmatrix} \frac{5}{6} & \frac{1}{6} & 0 \\ \frac{5}{6} & 0 & \frac{1}{6} \\ 0 & \frac{5}{6} & \frac{1}{6} \end{bmatrix}$$

If you need "border" or "indexes" on your matrix, plain TeX provides the macro **\bordermatrix**

```
\[ M = \bordermatrix{~ & x & y \cr A & 1 & 0 \cr B & 0 & 1
\cr} \]
```

$$M = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

Matrices in running text

To insert a small matrix, and not increase leading in the line containing it, use smallmatrix environment:

```
A matrix in text must be set smaller
$\bigl(\begin{smallmatrix} a&b\\ c&d
\end{smallmatrix} \bigr)$ to not increase leading
in a portion of text.
```

A matrix in text must be set smaller: $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ to not increase leading in a portion of text.

Adding text to equations

The math environment differs from the text environment in the representation of text. Here is an example of trying to represent text within the math environment:

```
\[ 50 apples \times 100 apples = lots of apples^2 \]
```

$$50apples \times 100apples = lotsofapples^2$$

There are two noticeable problems: there are no spaces between words or numbers, and the letters are italicized and more spaced out than normal. Both issues are simply artifacts of the maths mode, in that it treats it as a mathematical expression: spaces are ignored (LaTeX spaces mathematics according to its own rules), and each character is a separate element (so are not positioned as closely as normal text).

There are a number of ways that text can be added properly. The typical way is to wrap the text with the **\text**{ ... } command [3] (a similar command is **\mbox**{ ... }, though this causes problems with subscripts, and has a less descriptive name). Let's see what happens when the above equation code is adapted:

```
\[ 50 \text{apples} \times 100 \text{apples} =
\text{lots of apples}^2 \]
```

$$50\text{apples} \times 100\text{apples} = \text{lots of apples}^2$$

The text looks better. However, there are no gaps between the numbers and the words. Unfortunately, you are required to explicitly add these. There are many ways to add spaces between maths elements, but for the sake of simplicity you may literally add the space character in the affected **\text** (s) itself (just before the text.)

```
\[ 50 \text{ apples} \times 100 \text{ apples} =
\text{lots of apples}^2 \]
```

$$50\ \text{apples} \times 100\ \text{apples} = \text{lots of apples}^2$$

Formatted text

Using the **\text** is fine and gets the basic result. Yet, there is an alternative that offers a little more flexibility. You may recall the introduction of font formatting commands, such as **\textrm** , **\textit** , **\textbf** , etc. These commands format the argument accordingly, e.g., **\textbf**{bold text } gives **bold text**. These commands are equally valid within a maths environment to include text. The added benefit here is that you can have better control over the font formatting, rather than the standard text achieved with **\text** .

```
\[ 50 \textrm{ apples} \times 100 \textbf{ apples}
= \textit{lots of apples}^2 \]
```

$$50\ \text{apples} \times 100\ \textbf{apples} = \textit{lots of apples}^2$$

Formatting mathematics symbols

So we can format text, what about formatting mathematics? There are a set of formatting commands very similar to the font formatting ones just used, except they are aimed specifically for text in maths mode (requires amsfonts)

LaTeX command	Sample	Description	Common use
`\mathnormal{...}`	$ABCDEFabcdef123456$	the default math font	most mathematical notation
`\mathrm{...}`	ABCDEFabcdef123456	this is the default or normal font, unitalicised	units of measurement, one word functions
`\mathit{...}`	$ABCDEFabcdef123456$	italicised font	
`\mathbf{...}`	**ABCDEFabcdef123456**	bold font	vectors
`\mathsf{...}`	ABCDEFabcdef123456	Sans-serif	
`\mathtt{...}`	ABCDEFabcdef123456	Monospace (fixed-width) font	
`\mathcal{...}`	$\mathcal{ABCDEF}\dashv\sqcup\sqcap\{\infty\in\ni\triangle\triangledown,$	Calligraphy (uppercase only)	often used for sheaves/schemes and categories, used to denote cryptological concepts like an *alphabet of definition* ($\mathcal{A}$), *message space* ($\mathcal{M}$), *ciphertext space* ($\mathcal{C}$) and *key space* ($\mathcal{K}$); Kleene's $\mathcal{O}$; naming convention in description logic
`\mathfrak{...}`[6]	$\mathfrak{ABCDEFabcdef123456}$	Fraktur	Almost canonical font for Lie algebras, with superscript used to denote New Testament papyri, ideals in ring theory
`\mathbb{...}`[6]	$\mathbb{ABCDEF}\mathfrak{U}\nvdash\nvDash\nVdash\not\supset\not\subset\not\in$	Blackboard bold	Used to denote special sets (e.g. real numbers)
`\mathscr{...}`[7]		Script	

The maths formatting commands can be wrapped around the entire equation, and not just on the textual elements: they only format letters, numbers, and uppercase Greek, and the rest of the maths syntax is ignored.

To bold lowercase Greek or other symbols use the `\boldsymbol` command[3] ; this will only work if there exists a bold version of the symbol in the current font. As a last resort there is the `\pmb` command[3] (poor mans bold): this prints multiple versions of the character slightly offset against each other

```
\[ \boldsymbol{\beta} = (\beta_1,\beta_2,\ldots,\beta_n) \]
```

$$\boldsymbol{\beta} = (\beta_1,\beta_2,\ldots,\beta_n)$$

To change the size of the fonts in math mode, see Changing font size.

Accents

So what to do when you run out of symbols and fonts? Well the next step is to use accents:

`a'`	a'	`a''`	a''	`a''''`	a''''	`a'''''`	a'''''		
`\hat{a}`	$\hat{a}$	`\bar{a}`	$\bar{a}$	`\overline{aaa}`	$\overline{aaa}$	`\check{a}`	$\check{a}$	`\tilde{a}`	$\tilde{a}$
`\grave{a}`	$\grave{a}$	`\acute{a}`	$\acute{a}$	`\breve{a}`	$\breve{a}$	`\vec{a}`	$\vec{a}$		
`\dot{a}`	$\dot{a}$	`\ddot{a}`	$\ddot{a}$	`\ddddot{a}`[3]		`\dddddot{a}`[3]			
`\not{a}`	$\not{a}$	`\mathring{a}`		`\widehat{AAA}`	$\widehat{AAA}$	`\widetilde{AAA}`			

Plus and minus signs

Latex deals with the + and − signs in two possible ways. The most common is as a binary operator. When two maths elements appear either side of the sign, it is assumed to be a binary operator, and as such, allocates some space either side of the sign. The alternative way is a sign designation. This is when you state whether a mathematical quantity is either positive or negative. This is common for the latter, as in maths, such elements are assumed to be positive unless a − is prefixed to it. In this instance, you want the sign to appear close to the appropriate element to show their association. If you put a + or a − with nothing before it but you want it to be handled like a binary operator you can add an *invisible* character before the operator using `{ }`. This can be useful if you are writing multiple-line formulas, and a new line could start with a = or a +, for example, then you can fix some strange alignments adding the invisible character where necessary.

A plus-minus sign used for uncertainty is written as:

```
\[ \pm \]
```

$$\pm$$

Controlling horizontal spacing

LaTeX is obviously pretty good at typesetting maths—it was one of the chief aims of the core Tex system that LaTeX extends. However, it can't always be relied upon to accurately interpret formulas in the way you did. It has to make certain assumptions when there are ambiguous expressions. The result tends to be slightly incorrect horizontal spacing. In these events, the output is still satisfactory, yet, any perfectionists will no doubt wish to *fine-tune* their formulas to ensure spacing is correct. These are generally very subtle adjustments.

There are other occasions where LaTeX has done its job correctly, but you just want to add some space, maybe to add a comment of some kind. For example, in the following equation, it is preferable to ensure there is a decent amount of space between the maths and the text.

```
\[ f(n) = \left\{ \begin{array}{l l} n/2 & \quad
\text{if $n$ is even}\\ -(n+1)/2 & \quad \text{if $n$
is odd}\\ \end{array} \right. \]
```

$$f(n) = \begin{cases} n/2 & \quad \text{if } n \text{ is even} \\ -(n+1)/2 & \quad \text{if } n \text{ is odd} \end{array} \right.$$

This code produces errors with Miktex 2.9 and does not yield the results seen on the right. Use \textrm instead of just \text.

(Note that this particular example can be expressed in more elegant code by the cases construct provided by the amsmath package described in Advanced Mathematics chapter.)

LaTeX has defined two commands that can be used anywhere in documents (not just maths) to insert some horizontal space. They are **`\quad`** and **`\qquad`**

A **`\quad`** is a space equal to the current font size. So, if you are using an 11pt font, then the space provided by **`\quad`** will also be 11pt (horizontally, of course.) The **`\qquad`** gives twice that amount. As you can see from the code from the above example, **`\quad`** s were used to add some separation between the maths and the text.

OK, so back to the fine tuning as mentioned at the beginning of the document. A good example would be displaying the simple equation for the indefinite integral of *y* with respect to *x*:

$$\int y \, \mathrm{d}x$$

If you were to try this, you may write:

```
\[ \int y \mathrm{d}x \]
```

$$\int y \mathrm{d}x$$

However, this doesn't give the correct result. LaTeX doesn't respect the white-space left in the code to signify that the *y* and the d*x* are independent entities. Instead, it lumps them altogether. A **\quad** would clearly be overkill is this situation—what is needed are some small spaces to be utilized in this type of instance, and that's what LaTeX provides:

Command	Description	Size
\,	small space	3/18 of a quad
\:	medium space	4/18 of a quad
\;	large space	5/18 of a quad
\!	negative space	-3/18 of a quad

NB you can use more than one command in a sequence to achieve a greater space if necessary.

So, to rectify the current problem:

```
\[ \int y\, \mathrm{d}x \]
```

$$\int y \, \mathrm{d}x$$

```
\[ \int y\: \mathrm{d}x \]
```

$$\int y \, \mathrm{d}x$$

```
\[ \int y\; \mathrm{d}x \]
```

$$\int y \, \mathrm{d}x$$

The negative space may seem like an odd thing to use, however, it wouldn't be there if it didn't have *some* use! Take the following example:

```
\[ \left( \begin{array}{c} n \\ r \end{array} \right) =
\frac{n!}{r!(n-r)!} \]
```

$$\left(\begin{array}{c} n \\ r \end{array} \right) = \frac{n!}{r!(n-r)!}$$

The matrix-like expression for representing binomial coefficients is too padded. There is too much space between the brackets and the actual contents within. This can easily be corrected by adding a few negative spaces after the left bracket and before the right bracket.

```
\[ \left(\! \begin{array}{c} n \\ r \end{array} \!\right) =
\frac{n!}{r!(n-r)!} \]
```

$$\left(\! \begin{array}{c} n \\ r \end{array} \!\right) = \frac{n!}{r!(n-r)!}$$

In any case, adding some spaces manually should be avoided whenever possible: it makes the source code more complex and it's against the basic principles of a What You See is What You Mean approach. The best thing to do is to define some commands using all the spaces you want and then, when you use your command, you don't have to add any other space. Later, if you change your mind about the length of the horizontal space, you can easily change it modifying only the command you defined before. Let us use an example: you want the *d* of a *dx* in an integral to be in roman font and a small space away from the rest. If you want to type an integral like **\int** x **\;**

`\mathrm{d} x` , you can define a command like this:

```
\newcommand{\dd}{\; \mathrm{d
```

}} in the preamble of your document. We have chosen `\dd` just because it reminds the "d" it replaces and it is fast to type. Doing so, the code for your integral becomes `\int x \dd x` . Now, whenever you write an integral, you just have to use the `\dd` instead of the "d", and all your integrals will have the same style. If you change your mind, you just have to change the definition in the preamble, and all your integrals will be changed accordingly.

Advanced Mathematics: AMS Math package

The AMS (American Mathematical Society) mathematics package is a powerful package that creates a higher layer of abstraction over mathematical LaTeX language; if you use it it will make your life easier. Some commands amsmath introduces will make other plain LaTeX commands obsolete: in order to keep consistency in the final output you'd better use amsmath commands whenever possible. If you do so, you will get an elegant output without worrying about alignment and other details, keeping your source code readable. If you want to use it, you have to add this in the preamble:

```
\usepackage{amsmath}
```

Introducing text and dots in formulas

amsmath defines also the `\dots` command, that is a generalization of the existing `\ldots` . You can use `\dots` in both text and math mode and LaTeX will replace it with three dots "…" but it will decide according to the context whether to put it on the bottom (like `\ldots`) or centered (like `\cdots`).

Dots

LaTeX gives you several commands to insert dots in your formulas. This can be particularly useful if you have to type big matrices omitting elements. First of all, here are the main dots-related commands LaTeX provides:

Code	Output	Comment
`\dots`	$\dots$	generic dots, to be used in text (outside formulas as well). It automatically manages whitespaces before and after itself according to the context, it's a higher level command.
`\ldots`	$\ldots$	the output is similar to the previous one, but there is no automatic whitespace management; it works at a lower level.
`\cdots`	$\cdots$	These dots are centered relative to the height of a letter. There is also the binary multiplication operator, `\cdot`, mentioned below.
`\vdots`	$\vdots$	vertical dots
`\ddots`	$\ddots$	diagonal dots
`\iddots`		inverse diagonal dots (requires the mathdots package)
`\hdotsfor{n}`	$\cdots\cdots$	to be used in matrices, it creates a row of dots spanning *n* columns.

Instead of using `\ldots` and `\cdots` , you should use the semantically oriented commands. It makes it possible to adapt your document to different conventions on the fly, in case (for example) you have to submit it to a publisher who insists on following house tradition in this respect. The default treatment for the various kinds follows American Mathematical Society conventions.

Code	Output	Comment
`A_1,A_2,\dotsc,`	$A_1, A_2, \dotsc,$	for "dots with commas"
`A_1+\dotsb+A_N`	$A_1 + \dotsb + A_N$	for "dots with binary operators/relations"
`A_1 \dotsm A_N`	$A_1 \dotsm A_N$	for "multiplication dots"
`\int_a^b \dotsi`	$\int_a^b \dotsi$	for "dots with integrals"
`A_1\dotso A_N`	$A_1 \dotso A_N$	for "other dots" (none of the above)

List of Mathematical Symbols

All the pre-defined mathematical symbols from the \TeX\ package are listed below. More symbols are available from extra packages.

Relation Symbols

Symbol	Script	Symbol	Script	Symbol	Script	Symbol	Script	Symbol	Script
$\leq$	`\leq`	$\geq$	`\geq`	$\equiv$	`\equiv`	$\models$	`\models`	$\prec$	`\prec`
$\succ$	`\succ`	$\sim$	`\sim`	$\perp$	`\perp`	$\preceq$	`\preceq`	$\succeq$	`\succeq`
$\simeq$	`\simeq`	$\mid$	`\mid`	$\ll$	`\ll`	$\gg$	`\gg`	$\asymp$	`\asymp`
$\parallel$	`\parallel`	$\subset$	`\subset`	$\supset$	`\supset`	$\approx$	`\approx`	$\bowtie$	`\bowtie`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cong$	`\cong`	$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`
$\neq$	`\neq`	$\smile$	`\smile`	$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\doteqdot$	`\doteq`
$\frown$	`\frown`	$\in$	`\in`	$\ni$	`\ni`	$\notin$	`\notin`	$\propto$	`\propto`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$<$	`<`	$>$	`>`	$=$	`=`

Greek Letters

Symbol	Script
A and α	`\Alpha` and `\alpha`
B and β	`\Beta` and `\beta`
Γ and γ	`\Gamma` and `\gamma`
Δ and δ	`\Delta` and `\delta`
E, ϵ and ε	`\Epsilon`, `\epsilon` and `\varepsilon`
Z and ζ	`\Zeta` and `\zeta`
H and η	`\Eta` and `\eta`
Θ, θ and ϑ	`\Theta`, `\theta` and `\vartheta`
I and ι	`\Iota` and `\iota`
K and κ	`\Kappa` and `\kappa`
Λ and λ	`\Lambda` and `\lambda`
M and μ	`\Mu` and `\mu`
N and ν	`\Nu` and `\nu`

Ξ and ξ	`\Xi` and `\xi`
O and o	`\Omicron` and `\omicron`
Π, π and ϖ	`\Pi`, `\pi` and `\varpi`
P, ρ and ϱ	`\Rho`, `\rho` and `\varrho`
Σ, σ and ς	`\Sigma`, `\sigma` and `\varsigma`
T and τ	`\Tau` and `\tau`
Υ and υ	`\Upsilon` and `\upsilon`
Φ, ϕ, and φ	`\Phi`, `\phi` and `\varphi`
X and χ	`\Chi` and `\chi`
Ψ and ψ	`\Psi` and `\psi`
Ω and ω	`\Omega` and `\omega`

Binary Operations

Symbol	Script	Symbol	Script	Symbol	Script	Symbol	Script
$\pm$	`\pm`	$\cap$	`\cap`	$\diamond$	`\diamond`	$\oplus$	`\oplus`
$\mp$	`\mp`	$\cup$	`\cup`	$\triangle$	`\bigtriangleup`	$\ominus$	`\ominus`
$\times$	`\times`	$\uplus$	`\uplus`	$\triangledown$	`\bigtriangledown`	$\otimes$	`\otimes`
$\div$	`\div`	$\sqcap$	`\sqcap`	$\triangleleft$	`\triangleleft`	$\oslash$	`\oslash`
$*$	`\ast`	$\sqcup$	`\sqcup`	$\triangleright$	`\triangleright`	$\odot$	`\odot`
$\star$	`\star`	$\vee$	`\vee`	$\bigcirc$	`\bigcirc`	$\circ$	`\circ`
$\wedge$	`\wedge`	$\dagger$	`\dagger`	$\bullet$	`\bullet`	$\setminus$	`\setminus`
$\ddagger$	`\ddagger`	$\cdot$	`\cdot`	$\wr$	`\wr`	$\amalg$	`\amalg`

Set and/or Logic Notation

Symbol	Script
$\exists$	`\exists`
$\nexists$	`\nexists`
$\forall$	`\forall`
$\neg$	`\neg`
$\in$	`\in`
$\notin$	`\notin`
$\ni$	`\ni`
$\land$	`\land`
$\lor$	`\lor`
$\rightarrow$	`\rightarrow`
$\implies$	`\implies`
$\Rightarrow$	`\Rightarrow` (preferred for implication)
$\iff$	`\iff`
$\Leftrightarrow$	`\Leftrightarrow` (preferred for equivalence (iff))

$\top$	`\top`
$\bot$	`\bot`
$\emptyset$ and $\varnothing$	`\emptyset` and `\varnothing`

Delimiters

Symbol	Script
$\uparrow$	`\uparrow`
$\Uparrow$	`\Uparrow`
$\downarrow$	`\downarrow`
$\Downarrow$	`\Downarrow`
$\{$	`\{`
$\}$	`\}`
$\lceil$	`\lceil`
$\rceil$	`\rceil`
$\lfloor$	`\lfloor`
$\rfloor$	`\rfloor`
$\langle$	`\langle`
$\rangle$	`\rangle`
$/$	`/`
$\backslash$	`\backslash`
$\vert$	`\vert`
$\Vert$	`\Vert`

Other symbols

Symbol	Script
∂	`\partial`
∞	`\infty`
∇	`\nabla`
$\hbar$	`\hbar`
$\Box$	`\Box`
$\aleph$	`\aleph`
ℓ	`\ell`
$\imath$	`\imath`
$\jmath$	`\jmath`
$\Re$	`\Re`
$\Im$	`\Im`
$\wp$	`\wp`
$\eth$	`\eth`

Trigonometric Functions

Symbol	Script	Symbol	Script	Symbol	Script	Symbol	Script
$\sin$	`\sin`	$\cos$	`\cos`	$\tan$	`\tan`	$\cot$	`\cot`
$\arcsin$	`\arcsin`	$\arccos$	`\arccos`	$\arctan$	`\arctan`	arccot	`\arccot`
$\sinh$	`\sinh`	$\cosh$	`\cosh`	$\tanh$	`\tanh`	$\coth$	`\coth`
$\sec$	`\sec`	$\csc$	`\csc`				

Summary

As you begin to see, typesetting math can be tricky at times. However, because Latex provides so much control, you can get professional quality mathematics typesetting with relatively little effort (once you've had a bit of practice, of course!). It would be possible to keep going and going with math topics because it seems potentially limitless. However, with this tutorial, you should be able to get along sufficiently.

Notes

[1] http://www.ams.org/publications/authors/tex/amslatex
[2] http://www.tex.ac.uk/ctan/macros/latex/contrib/mh/mathtools.pdf
[3] requires the amsmath package
[4] http://hdl.handle.net/2268/6219
[5] requires the mathtools package
[6] requires `amsfonts` or `amssymb` packages
[7] require `mathrsfs` package

Further reading

- meta:Help:Displaying a formula: Wikimedia uses a subset of LaTeX commands.

External links

- LaTeX maths symbols (http://www.artofproblemsolving.com/Wiki/index.php/LaTeX:Symbols)
- detexify (http://detexify.kirelabs.org): applet for looking up LaTeX symbols by handwriting them
- `amsmath` documentation (ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf)
- LaTeX - The Student Room (http://www.thestudentroom.co.uk/wiki/LaTeX)
- The Comprehensive LaTeX Symbol List (http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-letter.pdf)

Advanced Mathematics

This page outlines some more advanced uses of mathematics markup using LaTeX. In particular it makes heavy use of the AMS-LaTeX packages supplied by the American Mathematical Society.

Equation numbering

The `equation` environment automatically numbers your equation.

```
\begin{equation} f(x)=(x+a)(x+b) \end{equation}
```

$$f(x) = (x+a)(x+b) \quad (1)$$

You can also use the **`\label`** and **`\ref`** commands to label and reference equations respectively. Or **`\eqref`** but this requires amsmath package, here is an example showing how to reference formulas using **`\ref`** (results in 1 for equation 1) and then using **`\eqref`** using the amsmath package (results in (1) for equation 1):

```
\begin{equation} \label{eq:someequation} 5^2 - 5 = 20
\end{equation} this references the equation
\ref{eq:someequation}.
```

$$5^2 - 5 = 20 \quad (1)$$
this references the equation 1.

```
\begin{equation} \label{eq:erl} a = bq + r \end{equation}
where \eqref{eq:erl} is true if $a$ and $b$ are integers
with $b \neq c$.
```

$$a = bq + r \quad (1)$$
where (1) is true if a and b are integers with $b \neq c$.

Further information is provided in the labels and cross-referencing chapter.

To have the enumeration follow from your section or subsection heading, you must use the amsmath package or use AMS class documents. Then enter

```
\numberwithin{equation}{section}
```

to the preamble to get enumeration at the section level or

```
\numberwithin{equation}{subsection}
```

to have the enumeration go to the subsection level.

```
\documentclass[12pt]{article} \usepackage{amsmath}
\numberwithin{equation}{subsection} \begin{document} \section{First
Section} \subsection{A subsection} \begin{equation} L' =
{L}{\sqrt{1-\frac{v^2}{c^2
```

Rendering missing!

}

```
\end{equation}
```

\end{document} \render= $L' = L\sqrt{1 - \frac{v^2}{c^2}} \quad (1.1.1)$}}

If the style you follow requires putting dots after ordinals (as it is required at least in Polish typography) the **`\numberwithin`**`{equation}{subsection}` command in preamble will result in the equation number in the above example to be rendered in this way: (1.1..1).

To remove the duplicate dot, add following command immediately after **`\numberwithin`**`{equation}{subsection}` :

```
\renewcommand{\theequation}{\thesection\arabic{equation
```

}}

For numbering scheme using `\numberwithin{equation}{subsection}` use:

```
\renewcommand{\theequation}{\thesubsection\arabic{equation
```

}} in the preamble of the document.

Note: Though it may look like the `\renewcommand` works by itself, it won't reset the equation number with each new section. It must be used together with manual equation number resetting after each new section beginning or with the much cleaner `\numberwithin` .

Subordinate equation numbering

To number subordinate equations in a numbered equation environment, place the part of document containing them in a subequations environment:

```
\begin{subequations} Maxwell's equations: \begin{align}
B'&=-\partial \times E,\\ E'&=\partial \times B - 4\pi j,
\end{align} \end{subequations}
```

Maxwell's equations:

$$B' = -\partial \times E, \tag{1.1a}$$
$$E' = \partial \times B - 4\pi j, \tag{1.1b}$$

Vertically aligning displayed mathematics

An often encountered problem with displayed environments (displaymath and equation) is the lack of any ability to span multiple lines. While it is possible to define lines individually, these will not be aligned.

Above and below

The `\overset` and `\underset` commands[1] typeset symbols above and below expressions. Without AmsTex the same result of `\overset` can be obtained with `\stackrel` . This can be particularly useful for creating new binary relations:

```
\[ A \overset{!}{=} B; A \stackrel{!}{=} B \]
```

$A \overset{!}{=} B;\ A \stackrel{!}{=} B$

or to show usage of L'Hôpital's rule:

```
\[ \lim_{x\to 0}{\frac{e^x-1}{2x
```

Rendering missing!

```
\overset{\left[\frac{0}{0}\right]}{\underset{\mathrm{H}}{=}}
\lim_{x\to 0}{\frac{e^x}{2}}={\frac{1}{2}}
```

\] |render= $\lim_{x\to 0} \frac{e^x - 1}{2x} \overset{\left[\frac{0}{0}\right]}{\underset{\mathrm{H}}{=}} \lim_{x\to 0} \frac{e^x}{2} = \frac{1}{2}$}}

It's convenient to define a new operator that will set the equal sign with H and provided fraction:

```
\newcommand{\Heq}[1]{\overset{\left[#1\right]}{\underset{\mathrm{H
```

{=}}} }} to reduce above example to:

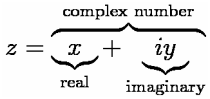
```
\[ \lim_{x\to 0}{\frac{e^x-1}{2x
```

```
\Heq{\frac{0}{0}}
\lim_{x\to 0}{\frac{e^x}{2}}={\frac{1}{2}}
```

\] }}

If the purpose is to make comments on particular parts of an equation, the **\overbrace** and **\underbrace** commands may be more useful, however they have a different syntax:

```
\[ z = \overbrace{ \underbrace{x}_\text{real} +
\underbrace{iy}_\text{imaginary} }^\text{complex number} \]
```



Sometimes the comments are longer than the formula being commented on, which can cause spacing problems. These can be removed using the **\mathclap** command[2] :

```
\[ y = a + f(\underbrace{b x}_{ \ge 0 \text{ by assumption
```

Rendering missing!

)

```
  = a + f(\underbrace{b x}_{
        \mathclap{\ge 0 \text{ by assumption}}})
```

$$y = a + f(\underbrace{\qquad bx \qquad}_{\geq 0 \text{ by assumption}}) = a + f(\underbrace{bx}_{\geq 0 \text{ by assumption}})$$

\] |render= }}

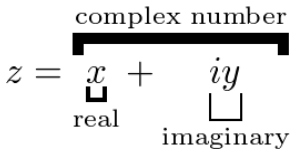
Alternatively, to use brackets instead of braces use **\underbracket** and **\overbracket** commands[2] :

```
\[ z = \overbracket[3pt]{ \underbracket{x}_{\text{real
```

Rendering missing!

+

```
    \underbracket[0.5pt][7pt]{iy}_{\text{imaginary}}
    }^{\text{complex number}}
```



\] |render= }}

The optional arguments set the rule thickness and bracket height respectively:

```
\underbracket[rule thickness][bracket height]{argument}_{text below}
```

The **\xleftarrow** and **\xrightarrow** commands[1] produce arrows which extend to the length of the text. Yet again, the syntax is different: the optional argument (using `[` & `]`) specifies the subscript, and the mandatory argument (using `{` & `}`) specifies the superscript (this can be left empty).

```
\[ A \xleftarrow{\text{this way
```

Rendering missing!

B

```
 \xrightarrow[\text{or that way}]{} C
```

\] |render= $A \xleftarrow[\text{this way}]{} B \xrightarrow[\text{or that way}]{} C$ }}

For more extensible arrows, you must use mathtools package:

```
\[ a \xleftrightarrow[under]{over} b\\ % A \xLeftarrow[under]{over} B\\
% B \xRightarrow[under]{over} C\\ % C \xLeftrightarrow[under]{over} D\\
% D \xhookleftarrow[under]{over} E\\ % E \xhookrightarrow[under]{over}
F\\ % F \xmapsto[under]{over} G\\ \]
```

$$a \xleftrightarrow[under]{over} b$$
$$A \xLeftarrow[under]{over} B$$
$$B \xRightarrow[under]{over} C$$
$$C \xLeftrightarrow[under]{over} D$$
$$D \xhookleftarrow[under]{over} E$$
$$E \xhookrightarrow[under]{over} F$$
$$F \xmapsto[under]{over} G$$

and for harpoons:

```
\[ H \xrightharpoondown[under]{over} I\\ % I
\xrightharpoonup[under]{over} J\\ % J \xleftharpoondown[under]{over}
K\\ % K \xleftharpoonup[under]{over} L\\ % L
\xrightleftharpoons[under]{over} M\\ % M
\xleftrightharpoons[under]{over} N \]
```

$$H \xrightharpoondown[under]{over} I$$
$$I \xrightharpoonup[under]{over} J$$
$$J \xleftharpoondown[under]{over} K$$
$$K \xleftharpoonup[under]{over} L$$
$$L \xrightleftharpoons[under]{over} M$$
$$M \xleftrightharpoons[under]{over} N$$

align and align*

The align and align* environments[1] are used for arranging equations of multiple lines. As with matrices and tables, `\\` specifies a line break, and `&` is used to indicate the point at which the lines should be aligned.

The align* environment is used like the displaymath or equation* environment:

```
\begin{align*} f(x) &= (x+a)(x+b) \\ &= x^2 + (a+b)x + ab
\end{align*}
```

$$\begin{aligned} f(x) &= (x+a)(x+b) \\ &= x^2 + (a+b)x + ab \end{aligned}$$

To force numbering on a specific line, use the `\tag{...}` command before the linebreak.

The align is similar, but automatically numbers each line like the equation environment. Individual lines may be referred to by placing a `\label{...}` before the linebreak. The `\nonumber` or `\notag` command can be used to suppress the number for a given line:

```
\begin{align} f(x) &= x^4 + 7x^3 + 2x^2 \nonumber \\
&\qquad {} + 10x + 12 \end{align}
```

$$\begin{aligned} f(x) &= x^4 + 7x^3 + 2x^2 \\ &\qquad + 10x + 12 \end{aligned} \tag{3}$$

Notice that we've added some indenting on the second line. Also, we need to insert the double braces {} before the + sign because otherwise latex won't create the correct spacing after the + sign. The reason for this is that without the braces, latex interprets the + sign as a unary operator, instead of the binary operator that it really is.

Braces spanning multiple lines

Additional `&`'s on a single line will specify multiple "equation columns", each of which is aligned. If you want a brace to continue across a new line, do the following:

```
\begin{align} f(x) &= \pi \left\{ x^4 + 7x^3 + 2x^2
\right.\nonumber\\ &\qquad \left. {} + 10x + 12
\right\} \end{align}
```

$$\begin{aligned} f(x) &= \pi \left\{ x^4 + 7x^3 + 2x^2 \right. \\ &\qquad \left. + 10x + 12 \right\} \end{aligned} \tag{4}$$

The sizes of the left and right braces are made equal and matching the typical size of the symbols between them by using `\left\{` and `\right\}`. But because these two commands occur on different lines, we need to balance them with the `\right.` and `\left.` commands. (Normally these aren't needed if the formula is on one line.)

Alternatively, you can control the size of the braces manually with the **\big** , **\Big** , **\bigg** , **\Bigg** commands.

To automatically match sizes of opening and closing braces in a tall equation use **\vphantom** command:

```
\begin{align} A &= \left(\int_t XXX \right.\nonumber\\
&\qquad \left.\vphantom{\int_t} YYY \dots \right)
\end{align}
```

$$A = \left(\int_t XXX \right.$$
$$\left. YYY \dots \right) \qquad (5)$$

The cases environment

The cases environment[1] allows the writing of piecewise functions:

```
\[ u(x) = \begin{cases} \exp{x} & \text{if } x \geq 0 \\ 1 &
\text{if } x < 0 \end{cases} \]
```

$$u(x) = \begin{cases} \exp x & \text{if } x \geq 0 \\ 1 & \text{if } x < 0 \end{cases}$$

Just like before, you don't have to take care of definition or alignment of columns, LaTeX will do it for you.

Unfortunately, it sets the internal math environment to text style, leading to such result:

$$a = \begin{cases} \int x\,dx \\ b^2 \end{cases}$$

To force display style for equations inside this construct, use dcases environment[2] :

```
\[ a = \begin{dcases} \int x\, dx\\ b^2 \end{dcases} \]
```

$$a = \begin{cases} \displaystyle\int x\,dx \\ b^2 \end{cases}$$

Often the second column consists mostly of normal text, to set it in normal roman font of the document use dcases* environment[2] :

```
\[ f(x) = \begin{dcases*} x & when $x$ is even\\ -x &
when $x$ is odd \end{dcases*} \]
```

$$f(x) = \begin{cases} x & \text{when } x \text{ is even} \\ -x & \text{when } x \text{ is odd} \end{cases}$$

Other environments

Although align and align* are the most useful, there are several other environments which may also be of interest:

Environment name	Description	Notes
eqnarray and eqnarray*	Similar to align and align*	Not recommended since spacing is inconsistent
multline and multline*[1]	First line left aligned, last line right aligned	Equation number aligned vertically with first line and not centered as with other other environments.
gather and gather*[1]	Consecutive equations without alignment	
flalign and flalign*[1]	Similar to align, but left aligns first equation column, and right aligns last column	
alignat and alignat*[1]	Takes an argument specifying number of columns. Allows to control explicitly the horizontal space between equations	You can calculate the number of columns by counting & characters in a line, adding 1 and dividing the result by 2

There are also few environments that don't form a math environment by themselves and can be used as building blocks for more elaborate structures:

Math environment name	Description	Notes
gathered[1]	Allows to gather few equations to be set under each other and assigned a single equation number	
split[1]	Similar to align*, but used inside another displayed mathematics environment	
aligned[1]	Similar to align, to be used inside another mathematics environment.	
alignedat[1]	Similar to alignat, and just as it, takes an additional argument specifying number of columns of equations to set.	

For example:

```
\begin{equation} \left.\begin{aligned}
B'&=-\partial \times E,\\ E'&=\partial
\times B - 4\pi j, \end{aligned} \right\}
\qquad \text{Maxwell's equations}
\end{equation}
```

$$\left.\begin{aligned} B' &= -\partial \times E, \\ E' &= \partial \times B - 4\pi j, \end{aligned} \right\} \qquad \text{Maxwell's equations} \tag{1.1}$$

```
\begin{alignat}{2} \sigma_1 &= x + y &\quad \sigma_2 &= \frac{x}{y} \\
\sigma_1' &= \frac{\partial x + y}{\partial x} & \sigma_2' &=
\frac{\partial \frac{x}{y}
{\partial x} \end{alignat} |render=
```

Rendering missing!

$$\sigma_1 = x + y \qquad \sigma_2 = \frac{x}{y} \tag{1}$$

$$\sigma_1' = \frac{\partial x + y}{\partial x} \qquad \sigma_2' = \frac{\partial \frac{x}{y}}{\partial x} \tag{2}$$

}}

Indented Equations

In order to indent an equation, you can set fleqn in the document class and then specify a certain value for `\mathindent` variable:

```
\documentclass[a4paper,fleqn]{report}
\usepackage{amsmath} \setlength{\mathindent}{1cm}
\begin{document} \noindent Euler's formula is given
below: \begin{equation*} e^{ix} = \cos{x} + i
\sin{x} \end{equation*} \noindent This is a very
important formula. \end{document}
```

Euler's formula is given below:

$$e^{ix} = \cos x + i \sin x$$

This is a very important formula.

Page breaks in math environments

To suggest LaTeX a page break inside one of amsmath environments you may use the **\displaybreak** command before line break. Just like with **\pagebreak** , **\displaybreak** can take an optional argument between 0 and 4 denoting the level of desirability of a page break in specific break. While 0 means "it is permissible to break here", 4 forces a break. No argument means the same as 4.

Alternatively, you may enable automatic page breaks in math environments with **\allowdisplaybreaks** . It can too have an optional argument denoting permissiveness of page breaks in equations. Similarly, 1 means "allow page breaks but avoid them" and 4 means "break whenever you want". You can prohibit a page break after a given line using ***** .

LaTeX will insert a page break into a long equation if it has additional text added using **\intertext{ }** without any additional commands though.

Specific usage may look like this:

```
\begin{align*} &\vdots\\ &=12+7 \int_0^2 \left(
-\frac{1}{4}\left(e^{-4t_1}+e^{4t_1-8}\right)
\right)\,dt_1\displaybreak[3]\\ &=
12-\frac{7}{4}\int_0^2 e^{-4t_1}+e^{4t_1-8}\,dt_1\\
&\vdots % \end{align*}
```

$$\vdots$$
$$= 12 + 7 \int_0^2 \left(-\frac{1}{4} \left(e^{-4t_1} + e^{4t_1 - 8} \right) \right) dt_1$$

1

$$= 12 - \frac{7}{4} \int_0^2 e^{-4t_1} + e^{4t_1 - 8} \, dt_1$$
$$\vdots$$

Boxed Equations

For a single equation, with the tag outside the box, use **\boxed{ }**:

```
\begin{equation} \boxed{x^2+y^2 = z^2}
\end{equation}
```

$$\boxed{x^2 + y^2 = z^2} \tag{1}$$

If you want the entire line or several equations to be boxed, use a minipage inside an **\fbox{ }**:

```
\fbox{ \addtolength{\linewidth}{-2\fboxsep}%
\addtolength{\linewidth}{-2\fboxrule}%
\begin{minipage}{\linewidth} \begin{equation}
x^2+y^2=z^2 \end{equation} \end{minipage} }
```

$$x^2 + y^2 = z^2 \tag{1}$$

There is also the mathtools **\Aboxed{ }** which is able to box across alignment marks

```
\begin{align*} \Aboxed{ f(x) & = \int h(x)\, dx} \\ & =
g(x) \end{align*}
```

$$\boxed{f(x) = \int h(x)\, dx}$$
$$= g(x)$$

Custom operators

Although many common operators are available in LaTeX, sometimes you will need to write your own, for example to typeset the argmax operator. The **\operatorname** and **\operatorname*** commands[1] display a custom operators, the * version sets the underscored option underneath like the **\lim** operator:

```
\[ \operatorname{arg\,max}_a f(a) =
\operatorname*{arg\,max}_b f(b) \]
```

$$\operatorname{arg\,max}_a f(a) = \operatorname*{arg\,max}_b f(b)$$

However if the operator is frequently used, it is preferable to keep within the LaTeX ideal of markup to define a new operator. The **\DeclareMathOperator** and **\DeclareMathOperator*** commands[1] are specified in the header of the document:

```
\DeclareMathOperator*{\argmax}{arg\,max}
```

This defines a new command which may be referred to in the body:

```
\[ \argmax_c f(c) \]
```

$$\operatorname*{arg\,max}_c f(c)$$

Advanced formatting

Limits

There are defaults for placement of subscripts and superscripts. For example, limits for the `lim` operator are usually placed below the symbol, like this:

```
\begin{equation} \lim_{a\to \infty} \tfrac{1}{a} \end{equation}
```

$$\lim_{a\to \infty} \tfrac{1}{a}$$

To override this behavior use the **\nolimits** operator:

```
\begin{equation} \lim\nolimits_{a\to \infty} \tfrac{1}{a}
\end{equation}
```

$$\lim_{a\to \infty} \tfrac{1}{a}$$

A `lim` in running text (inside `$...$`) will have its limits placed on the side, so that additional leading won't be required. To override this behavior use **\limits** command.

Similarly one can put subscripts under a symbol that usually have them on the side:

```
\begin{equation} \int_a^b x^2 \end{equation}
```

$$\int_a^b x^2$$

Limits below and under:

```
\begin{equation} \int\limits_a^b x^2 \end{equation}
```

$$\int\limits_a^b x^2$$

To change default placement in all instances of summation-type symbol to the side add nosumlimits option to amsmath package. To change placement for integral symbols add intlimits to options and nonamelimits to change the default for named operators like `det`, `min`, `lim`...

Custom prod like operator

To declare custom text instead of sum, lim or prod, you could put in your preamble (before the document begins) something like this:

\DeclareMathOperator*{\specialE}{\ddot{e}}

```
\DeclareMathOperator*{\specialE}{\ddot{e
```
Rendering missing!

... \begin{equation}

```
\specialE_x
```

\end{equation} |render= $TODO I am not able to find how to render example here...$ }}

Subscripts and superscripts

While you can place symbols in sub- or superscript in summation style symbols with the above introduced **\nolimits** :

```
\begin{equation} \sum\nolimits' C_n \end{equation}
```
$$\sum\nolimits' C_n$$

It's impossible to mix them with typical usage of such symbols:

```
\begin{equation} \sum_{n=1}\nolimits' C_n \end{equation}
```
$$\sum\nolimits'_{n=1} C_n$$

To add both prime and a limit to a symbol, one have to use **\sideset** command:

```
\begin{equation} \sideset{}{'}\sum_{n=1}C_n \end{equation}
```
$$\sideset{}{'}\sum_{n=1} C_n$$

It's very flexible, for example, to put letters in each corner of the symbol use this command:

```
\begin{equation} \sideset{_a^b}{_c^d}\sum \end{equation}
```
$$\sideset{_a^b}{_c^d}\sum$$

If you wish to place them on the corners of an arbitrary symbol, you should use **\fourIdx** from the fouridx package.

Multiline subscripts

To produce multiline subscript use **\substack** command:

```
\begin{equation} \prod_{\substack{ 1\le i \le n\\ 1\le j \le m
```
Rendering missing!

```
M_{i,j}
```

\end{equation} |render= $\prod_{\substack{1\le i\le n\\1\le j\le m}} M_{i,j}$ }}

Text in aligned math display

To add small interjections in math environments use **\intertext** command:

```
\begin{minipage}{3in} \begin{align*} \intertext{If}
A &= \sigma_1+\sigma_2\\ B &= \rho_1+\rho_2\\
\intertext{then} C(x) &= e^{Ax^2+\pi}+B \end{align*}
\end{minipage}
```

If

$$A = \sigma_1 + \sigma_2$$
$$B = \rho_1 + \rho_2$$

then

$$C(x) = e^{Ax^2+\pi} + B$$

Note that usage of this command doesn't change alignment, as would stopping and restarting the align environment.

Also in this example, the command **\shortintertext{}** from the mathtools package could have been used instead of intertext to reduce the amount of vertical whitespace added between the lines.

Changing font size

Probably a rare event, but there may be a time when you would prefer to have some control of the size. For example, using text-mode maths, by default a simple fraction will look like this: $\frac{a}{b}$ where as you may prefer to have it displayed larger, like when in display mode, but still keeping it in-line, like this: $\displaystyle\frac{a}{b}$.

A simple approach is to utilize the predefined sizes for maths elements:

Size command	Description
\displaystyle	Size for equations in display mode
\textstyle	Size for equations in text mode
\scriptstyle	Size for first sub/superscripts
\scriptscriptstyle	Size for subsequent sub/superscripts

A classic example to see this in use is typesetting continued fractions (though it's better to use the **\cfrac** command[1] described in the Mathematics chapter over the method provided below). The following code provides an example.

```
\begin{equation} x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 +
\frac{1}{a_3 + a_4
```

Rendering missing!

} \end{equation} |render= $x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + a_4}}}$ }}

As you can see, as the fractions continue, they get smaller (although they will not get any smaller as in this example, they have reached the **\scriptstyle** limit). If you wanted to keep the size consistent, you could declare each fraction to use the display style instead, e.g.:

```
\begin{equation} x = a_0 + \frac{1}{\displaystyle a_1 +
\frac{1}{\displaystyle a_2 + \frac{1}{\displaystyle a_3 + a_4
```

Rendering missing!

} \end{equation} |render= $$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + a_4}}}$$ }}

Another approach is to use the **\DeclareMathSizes** command to select your preferred sizes. You can only define sizes for **\displaystyle** , **\textstyle** , etc. One potential downside is that this command sets the

global maths sizes, as it can only be used in the document preamble.

However, it's fairly easy to use: `\DeclareMathSizes{ds}{ts}{ss}{sss}`, where *ds* is the *display size*, *ts* is the *text size*, etc. The values you input are assumed to be point (pt) size.

NB the changes only take place if the value in the first argument matches the current document text size. It is therefore common to see a set of declarations in the preamble, in the event of the main font being changed. E.g.,

```
\DeclareMathSizes{10}{18}{12}{8} % For size 10 text
\DeclareMathSizes{11}{19}{13}{9} % For size 11 text
\DeclareMathSizes{12}{20}{14}{10} % For size 12 text
```

Forcing \displaystyle for all math in a document

Put

```
\everymath{\displaystyle}
```

before \begin{document} to force all math to \displaystyle.

Notes

[1] Requires the amsmath package
[2] requires the mathtools package

Theorems

With "theorem" we can mean any kind of labelled enunciation that we want to look separated from the rest of the text and with sequential numbers next to it. This approach is commonly used for theorems in mathematics, but can be used for anything. LaTeX provides a command that will let you easily define any theorem-like enunciation.

Basic theorems

First of all, make sure you have the amsthm package enabled:

```
\usepackage{amsthm}
```

The easiest is the following:

```
\newtheorem{name}{Printed output}
```

put it in the preamble. The first argument is the name you will use to reference it, the second argument is the output LaTeX will print whenever you use it. For example:

```
\newtheorem{mydef}{Definition}
```

will define the `mydef` environment; if you use it like this:

```
\begin{mydef}
Here is a new definition
\end{mydef}
```

It will look like this:

Definition 3 *Here is a new definition*

with line breaks separating it from the rest of the text.

Theorem counters

Often the counters are determined by section, for example "Theorem 2.3" refers to the 3rd theorem in the 2nd section of a document. In this case, specify the theorem as follows:

```
\newtheorem{name}{Printed output}[numberby]
```

where *numberby* specifies the section level (section/subsection/etc.) at which the numbering is to take place.

By default, each theorem uses its own counter. However it is common for similar types of theorems (e.g. Theorems, Lemmas and Corollaries) to share a counter. In this case, define subsequent theorems as:

```
\newtheorem{name}[counter]{Printed output}
```

where *counter* is the name of the counter to be used. Usually this will be the name of the master theorem.

You can also create a theorem environment that is not numbered by using the `newtheorem*` command[1] . For instance,

```
\newtheorem*{mydef}{Definition}
```

defines the `mydef` environment, which will generate definitions without numbering. This requires `amsthm` package.

Proofs

The `proof` environment[1] can be used for adding the proof of a theorem. The basic usage is:

```
\begin{proof}
Here is my proof
\end{proof}
```

It just adds *Proof* in italics at the beginning of the text given as argument and a white square (Q.E.D symbol, also known as a tombstone) at the end of it. If you are writing in another language than English, just use babel with the right argument and the word *Proof* printed in the output will be translated accordingly; anyway, in the source the name of the environment remains `proof`.

If you would like to manually name the proof, include the name in square brackets:

```
\begin{proof}[Proof of important theorem]
Here is my important proof
\end{proof}
```

If the last line of the proof is displayed math then the Q.E.D. symbol will appear on a subsequent empty line. To put the Q.E.D. symbol at the end of the last line, use the `\qedhere` command:

```
\begin{proof}
Here is my proof:
\[
a^2 + b^2 = c^2 \qedhere
\]
\end{proof}
```

The method above does not work with the deprecated environment `eqnarray*`. Here is a workaround:

```
\begin{proof}
Here is my proof:
\begin{eqnarray*}
a^2 + b^2 = c^2
\end{eqnarray*}
\vspace{-1.3cm}\[\qedhere\]
\end{proof}
```

To use a custom Q.E.D. symbol, redefine the `\qedsymbol` command. To hide the Q.E.D. symbol altogether, redefine it to be blank:

```
\renewcommand{\qedsymbol}{}
```

Theorem styles

It adds the possibility to change the output of the environments defined by `\newtheorem` using the `\theoremstyle` command[1] command in the header:

```
\theoremstyle{stylename}
```

the argument is the style you want to use. All subsequently defined theorems will use this style. Here is a list of the possible pre-defined styles:

stylename	Description
`plain`	Used for theorems, lemmas, propositions, etc. (default)
`definition`	Used for definitions and examples
`remark`	Used for remarks and notes

Custom styles

To define your own style, use the `\newtheoremstyle` command[1]:

```
\newtheoremstyle{stylename}% name of the style to be used
  {spaceabove}% measure of space to leave above the theorem. E.g.: 3pt
  {spacebelow}% measure of space to leave below the theorem. E.g.: 3pt
  {bodyfont}% name of font to use in the body of the theorem
  {indent}% measure of space to indent
  {headfont}% name of head font
  {headpunctuation}% punctuation between head and body
  {headspace}% space after theorem head; " " = normal interword space
  {headspec}% Manually specify head
```

(Any arguments that are left blank will assume their default value). Here is an example *headspec*:

```
\thmname{#1}\thmnumber{ #2}:\thmnote{ #3}
```

which would look something like:
Definition 2: Topology
for the following:

```
\begin{definition}[Topology]...
```

(The note argument, which in this case is Topology, is always optional, but will not appear by default unless you specify it as above in the head spec).

Conflicts

The theorem environment conflicts with other environments, for example *wrapfigure*. A work around is to redefine theorem, for example the following way:

```
% Fix latex
\def\smallskip{\vskip\smallskipamount}
\def\medskip{\vskip\medskipamount}
\def\bigskip{\vskip\bigskipamount}

% Hand made theorem
\newcounter{thm}[section]
\renewcommand{\thethm}{\thesection.\arabic{thm}}
\def\claim#1{\par\medskip\noindent\refstepcounter{thm}\hbox{\bf
\arabic{chapter}.\arabic{section}.\arabic{thm}. #1.}
\it\ %\ignorespaces
}
\def\endclaim{
\par\medskip}
\newenvironment{thm}{\claim}{\endclaim}
```

In this case theorem looks like:

```
\begin{thm}{Claim}\label{lyt-prob}
Let it be.
Then you know.
\end{thm}
```

Notes

[1] Requires the `amsthm` package

External links

- `amsthm` documentation (ftp://ftp.ams.org/pub/tex/doc/amscls/amsthdoc.pdf)

Linguistics

Recommended packages[1] :

- Glosses: gb4e;
- IPA symbols: tipa;
- OT Tableaux: OTtablx;
- Syntactic trees: qtree + tree-dvips (for drawing arrows);
  - Alternatively, xyling is very powerful but not as user friendly as qtree;
- Attribute-Value Matrices (AVMs): avm

References

[1] (http://jones.ling.indiana.edu/~mdickinson/08/latex/slides.pdf) LaTeX for Linguists presentation

Labels and Cross-referencing

Introduction

Another good point of LaTeX is that you can easily reference almost anything that is numbered (sections, figures, formulas), and LaTeX will take care of numbering, updating it whenever necessary. The commands to be used do not depend on what you are referencing, and they are:

`\label{`*`marker`*`}`

: you give the object you want to reference a *marker*, you can see it like a name.

`\ref{`*`marker`*`}`

: you can reference the object you have *marked* before. This prints the number that was assigned to the object.

`\pageref{`*`marker`*`}`

: It will print the number of the page where the object is.

LaTeX will calculate the right numbering for the objects in the document; the *marker* you have used to label the object will not be shown anywhere in the document. Then LaTeX will replace the string "`\ref{`*`marker`*`}`" with the right number that was assigned to the object. If you reference a *marker* that does not exist, the compilation of the document will be successful but LaTeX will return a warning:

```
LaTeX Warning: There were undefined references.
```

and it will replace "`\ref{`*`unknown-marker`*`}`" with "??" (so it will be easy to find in the document).

As you may have noticed reading how it works, it is a two-step process: first the compiler has to store the labels with the right number to be used for referencing, then it has to replace the `\ref` with the right number. That is why, when you use references, you have to compile your document twice to see the proper output. If you compile it only once, LaTeX will use the older information it collected in previous compilations (that might be outdated), but the compiler will inform you printing on the screen at the end of the compilation:

```
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references
right.
```

Using the command `\pageref{}` you can help the reader to find the referenced object by providing also the page number where it can be found. You could write something like:

```
See figure~\ref{fig:test} on page~\pageref{fig:test}.
```

Since you can use exactly the same commands to reference almost anything, you might get a bit confused after you have introduced a lot of references. It is common practice among LaTeX users to add a few letters to the label to describe *what* you are referencing. Here is an example:

chap:	chapter
sec:	section
fig:	figure
tab:	table
eq:	equation
lst:	code listing

Following this convention, the label of a figure will look like `\label{fig:my_figure}`, etc. You are not obligated to use these prefixes. You can use any string as argument of `\label{...}`, but these prefixes become increasingly useful as your document grows in size.

Another suggestion: try to avoid using numbers within labels. You are better off describing *what* the object is about. This way, if you change the order of the objects, you will not have to rename all your labels and their references.

If you want to be able to see the markers you are using in the output document as well, you can use the `showkeys` package; this can be very useful while developing your document. For more information see the Packages section.

Examples

Here are some practical examples, but you will notice that they are all the same because they all use the same commands.

Sections

```
\section{Greetings}
\label{sec:greetings}

Hello!

\section{Referencing}

I greeted in section~\ref{sec:greetings}.
```

1 Greetings

Hello!

2 Referencing

I greeted in section 1.

You could place the label anywhere in the section; however, in order to avoid confusion, it is better to place it immediately after the beginning of the section. Note how the marker starts with *sec:*, as suggested before. The label is then referenced in a different section. The tilde (~) indicates a non-breaking space.

Pictures

You can reference a picture by inserting it in the `figure` floating environment.

```
\begin{figure}
  \centering
    \includegraphics[width=0.5\textwidth]{gull}
  \caption{Close-up of a gull}
  \label{gull}
\end{figure}
Figure~\ref{gull} shows a photograph of a gull.
```

Figure 1: Close-up of a gull

Figure 1 shows a photograph of a gull.

When a label is declared within a float environment, the `\ref{...}` will return the respective fig/table number, but it must occur **after** the caption. When declared outside, it will give the section number. To be completely safe, the label for any picture or table can go within the `\caption{}` command, as in

```
\caption{Close-up of a gull\label{gull}}
```

See the Floats, Figures and Captions section for more about the `figure` and related environments.

Fixing wrong labels

The command `\label` must appear after (or inside) `\caption`. Otherwise, it will pick up the current section or list number instead of what you intended.

```
\begin{figure}
  \centering
  \includegraphics[width=0.5\textwidth]{gull}
  \caption{Close-up of a gull} \label{fig:gull}
\end{figure}
```

Issues with links to tables and figures handled by hyperref

In case you use the package `hyperref` to create a PDF, the links to tables or figures will point to the caption of the table or figure, which is always below the table or figure itself[1] . Therefore the table or figure will not be visible, if it is above the pointer and one has to scroll up in order to see it. If you want the link point to the top of the image you can use the package `hypcap` [2] with:

```
\usepackage[all]{hypcap}
```

Be sure to call this package *after* the package `hyperref`, which should otherwise be loaded last.

Formulas

Here is an example showing how to reference formulas:

```
\begin{equation} \label{eq:solve}
x^2 - 5 x + 6 = 0
\end{equation}

\begin{equation}
x_1 = \frac{5 + \sqrt{25 - 4 \times 6}}{2} = 3
\end{equation}

\begin{equation}
x_2 = \frac{5 - \sqrt{25 - 4 \times 6}}{2} = 2
\end{equation}

and so we have solved equation \ref{eq:solve}
```

$$x^2 - 5x + 6 = 0 \tag{1}$$

$$x_1 = \frac{5 + \sqrt{25 - 4 \times 6}}{2} = 3 \tag{2}$$

$$x_2 = \frac{5 - \sqrt{25 - 4 \times 6}}{2} = 2 \tag{3}$$

and so we have solved equation 1

As you can see, the label is placed soon after the beginning of the math mode. In order to reference a formula, you have to use an environment that adds numbers. Most of the times you will be using the `equation` environment; that is the best choice for one-line formulas, whether you are using `amsmath` or not. Note also the *eq:* prefix in the label.

eqref

The `amsmath` package adds a new command for referencing formulas; it is `\eqref{}`. It works exactly like `\ref{}`, but it adds brackets so that, instead of printing a plain number as *5*, it will print *(5)*. This can be useful to help the reader distinguish between formulas and other things, without the need to repeat the word "formula" before any reference. Its output can be changed as you wish; for more information see the `amsmath` documentation.

numberwithin

The `amsmath` package adds the `\numberwithin{countera}{counterb}` command which replaces the simple `countera` by a more sophisticated `counterb.countera`. For example `\numberwithin{equation}{section}` in the preamble will prepend the section number to all equation numbers.

The `varioref` package

The `varioref` package introduces a new command called `\vref{}`. This command is used exactly like the basic `\ref`, but it has a different output according to the context. If the object to be referenced is in the same page, it works just like `\ref`; if the object is far away it will print something like "5 on page 25", i.e. it adds the page number automatically. If the object is close, it can use more refined sentences like "on the next page" or "on the facing page" automatically, according to the context and the document class.

This command has to be used very carefully. It outputs more than one word, so it may happen its output falls on two different pages. In this case, the algorithm can get confused and cause a loop. Let's make an example. You label an object on page 23 and the `\vref` output happens to stay between page 23 and 24. If it were on page 23, it would print like the basic `ref`, if it were on page 24, it would print "on the previous page", but it is on both, and this may cause some strange errors at compiling time that are very hard to be fixed. You could think that this happens very rarely; unfortunately, if you write a long document it is not uncommon to have hundreds of references, so situations like these are likely to happen. One way to avoid problems during development is to use the standard `ref` all the time, and convert it to `vref` when the document is close to its final version, and then making adjustments to fix possible problems.

The hyperref package and \autoref{}

The hyperref package introduces another useful command; \autoref{}. This command creates a reference with additional text corresponding to the target's type, all of which will be a hyperlink. For example, the command \autoref{sec:intro} would create a hyperlink to the \label{sec:intro} command, wherever it is. Assuming that this label is pointing to a section, the hyperlink would contain the text "section 3.4", or similar (the full list of default names can be found here [3]). Note that, while there's an **\autoref*** command that produces an unlinked prefix (useful if the label is on the same page as the reference), no alternative **\Autoref** command is defined to produce capitalized versions (useful, for instance, when starting sentences); but since the capitalization or autoref names was chosen by the package author, you can customize the prefixed text by redefining *type*autorefname to the prefix you want, as in:

```
\def\sectionautorefname{Section}
```

This renaming trick can, of course, be used for other purposes as well.

- If you would like a hyperlink reference, but do not want the predefined text that \autoref{} provides, you can do this with a command such as \hyperref[sec:intro]{Appendix~\ref*{sec:intro}}. Note that you can disable the creation of hyperlinks in hyperref, and just use these commands for automatic text.
- Keep in mind that the \label **must** be placed inside an environment with a counter, such as a table or a figure. Otherwise, not only the number will refer to the current section, as mentioned above, but the name will refer to the previous environment with a counter. For example, if you put a label after closing a figure, the label will still say "figure n", on which n is the current section number.

The hyperref package and \nameref{}

The hyperref package also automatically includes the nameref package, and a similarly named command. It is similar to \autoref{}, but inserts text corresponding to the section name, for example.

Input:

```
\section{MyFirstSection} \label{sec:marker}
\section{MySecondSection}
In section~\nameref{sec:marker} we defined...
```

Output:

In section MyFirstSection we defined...

The hyperref package and \phantomsection

When you define a \label outside a figure, a table, or other floating objects, the label points to the current section. In some case, this behavior is not what you'd like and you'd prefer the generated link to point to the line where the \label is defined. This can be achieved with the command \phantomsection as in this example:

```
%The link location will be placed on the line below.
\phantomsection
\label{the_label}
```

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/hyperref/README
[2] http://www.ctan.org/tex-archive/macros/latex/contrib/oberdiek/hypcap.pdf
[3] http://www.tug.org/applications/hyperref/manual.html#TBL-23

Indexing

Especially useful in printed books, an index is an alphabetical list of words and expressions with the pages of the book upon which they are to be found. LaTeX supports the creation of indices with its package makeidx, and its support program `makeindex`, called on some systems `makeidx`.

Using `makeidx`

To enable the indexing feature of LaTeX, the makeidx package must be loaded in the preamble with:

```
\usepackage{makeidx}
```

and the special indexing commands must be enabled by putting the

```
\makeindex
```

command into the input file preamble. This should be done within the preamble, since it tells LaTeX to create the files needed for indexing. To tell LaTeX what to index, use

```
\index{key}
```

where *key* is the index entry and does not appear in the final layout. You enter the index commands at the points in the text that you want to be referenced in the index, likely near the reason for the *key*. For example, the text

```
To solve various problems in physics, it can be advantageous to express any
arbitrary piecewise-smooth function as a Fourier Series composed of multiples
of sine and cosine functions.
```

can be re-written as

```
To solve various problems in physics, it can be advantageous to express any
arbitrary piecewise-smooth function as a Fourier Series \index{Fourier Series}
composed of multiples of sine and cosine functions.
```

to create an entry called 'Fourier Series' with a reference to the target page. Multiple uses of *index* with the same *key* on different pages will add those target pages to the same index entry.

To show the index within the document, merely use the command

```
\printindex
```

It is common to place it at the end of the document. The default index format is two columns.

The showidx package that comes with LaTeX prints out all index entries in the left margin of the text. This is quite useful for proofreading a document and verifying the index.

Compiling Indices

When the input file is processed with LaTeX, each `\index` command writes an appropriate index entry, together with the current page number, to a special file. The file has the same name as the LaTeX input file, but a different extension (`.idx`). This `.idx` file can then be processed with the `makeindex` program. Type in the command line:

```
makeindex filename
```

Note that *filename* is without extension: the program will look for *filename.idx* and use that. You can optionally pass *filename.idx* directly to the program as an argument. The `makeindex` program generates a sorted index with the same base file name, but this time with the extension `.ind`. If now the LaTeX input file is processed again, this sorted index gets included into the document at the point where LaTeX finds `\printindex` .

The index created by latex with the default options may not look as nice or as suitable as you would like it. To improve the looks of the index `makeindex` comes with a set of style files, usually located somewhere in the tex directory structure, usually below the `makeindex` subdirectory. To tell `makeindex` to use a specific style file, run it with the command line option:

```
makeindex -s <style file> filename
```

If you use a GUI for compiling latex and index files, you may have to set this in the options. Here are some configuration tips for typical tools:

MakeIndex Settings in WinEdt

Say you want to add an index style file named `simpleidx.ist`

- Texify/PDFTexify: Options→Execution Modes→Accessories→PDFTeXify, add to the Switches: `--mkidx-option="-s simpleidx.ist"`
- MakeIndex alone: Options->Execution Modes→Accessories→MakeIndex, add to command line: `-s simpleidx.ist`

Sophisticated Indexing

Below are examples of `\index` entries:

Example	Index Entry	Comment
`\index{hello }`	hello, 1	Plain entry
`\index{hello!Peter }`	Peter, 3	Subentry under 'hello'
`\index{Sam@\textsl{Sam }}`	*Sam*, 2	Formatted entry
`\index{Lin@\textbf{Lin }}`	**Lin**, 7	Same as above
`\index{Jenny|textbf }`	Jenny, **3**	Formatted page number
`\index{Joe|textit }`	Joe, *5*	Same as above
`\index{ecole@\'ecole }`	école, 4	Handling of accents
`\index{Peter|see{hello }}`	Peter, *see* hello	Cross-references
`\index{Jen|seealso{Jenny }}`	Jen, *see also* Jenny	Same as above

Subentries

If some entry has subsections, these can be marked off with `!`. For example,

```
\index{encodings!input!cp850}
```

would an index with 'cp850' categorized under 'input' (which itself is categorized into 'encodings'). These are called subsubentries and subentries in makeidx terminology.

Controlling Sorting

In order to determine how an index key is sorted, place a value to sort by before the key with the `@` as a separator. This is useful if there is any formatting or math mode, so one example may be

```
\index{F@$\vec{F}$}
```

so that the entry in the index will show as '$\vec{F}$' but be sorted as 'F'.

Changing Page Number Style

To change the formatting of a page number, append a `|` and the name of some command which does the formatting. This command should only accept one argument.

For example, if on page 3 of a book you introduce bulldogs and include the command

```
\index{bulldog}
```

and on page 10 of the same book you wish to show the main section on bulldogs with a bold page number, use

```
\index{bulldog|textbf}
```

This will appear in the index as bulldog, 3, **10**

If you use `texindy` in place of `makeindex`, the classified entries will be sorted too, such that all the bolded entries will be placed before all others by default.

Multiple Pages

To perform multi-page indexing, add a `|(` and `|)` to the end of the `\index` command, as in

```
\index{Quantum Mechanics!History|(} In 1901, Max Planck released his theory of
radiation dependant on quantized energy. While this explained the ultraviolet
catastrophe in the spectrum of blackbody radiation, this had far larger
consequences as the beginnings of quantum mechanics. ... \index{Quantum
Mechanics!History|)}
```

The entry in the index for the subentry 'History' will be the range of pages between the two `\index` commands.

Using special characters

In order to place values with `!`, `@`, or `|` in the `\index` command, one must quote these characters by using a double quotation mark (") and can only show " by quoting it (i.e., a key for " would be `\index{""}`).

This rule does not hold for \", so to put ä in the index, one may still use `\index{a@\"{a}}`.

Warnings

Note that the `\index` command can affect your layout if not used carefully. Here is an example:

```
My Word \index{Word}. As opposed to Word\index{Word}.
Note the position of the full stop.
```

My Word . As opposed to Word. Note the position of the full stop.

Abbreviation list

You can make a list of abbreviations with the package nomencl [1]. You may also be interested in using the glossaries package described in the Glossary chapter.

To enable the Nomenclature feature of LaTeX, the nomencl package must be loaded in the preamble with:

```
\usepackage[options]{nomencl} \makenomenclature
```

Issue the `\nomenclature[prefix]{symbol}{description}` command for each symbol you want to have included in the nomenclature list. The best place for this command is immediately after you introduce the symbol for the first time. Put `\printnomenclature` at the place you want to have your nomenclature list.

Run LaTeX 2 times then

```
makeindex filename.nlo -s nomencl.ist -o filename.nls
```

followed by running LaTeX once again.

To add the abbreviation list to the table of content, intoc option can be used when declare the nomencl package, i.e.

```
\usepackage[intoc]{nomencl}
```

instead of using the code in Adding Index to Table Of Contents section.

The title of the list can be changed using the following command:

```
\renewcommand{\nomname}{List of Abbreviations}
```

Multiple indices

If you need multiple indices you can use the package multind [2].

This package provides the same commands as makeidx, but now you also have to pass a name as the first argument to every command.

```
\usepackage{multind} \makeindex{books} \makeindex{authors} ... \index{books}{A
book to index} \index{authors}{Put this author in the index} ...
\printindex{books}{The Books index} \printindex{authors}{The Authors index}
```

Adding Index to Table Of Contents

By default, Index won't show in Table Of Contents, you have to add it manually.

To add index as a chapter, use this commands:

```
\clearpage \addcontentsline{toc}{chapter}{Index} \printindex
```

If you use book class, you may want to start it on odd page, for this, use `\cleardoublepage` .

International indices

If you want to sort entries that have international characters (such as ő, ą, ó, ç, etc.) you may find that the sorting "is not quite right". In most cases the characters are treated as special characters and end up in the same group as @, ¶ or µ. In most languages that use Latin alphabet it's not correct.

Generating index

Unfortunately, current version of `xindy` and hyperref are incompatible. When you use textbf or textit modifiers, `texindy` will print error message:`unknown cross-reference-class `hyperindexformat'! (ignored)` and won't add those pages to index. Work-around for this bug is described on the talk page.

To generate international index file you have to use `texindy` instead of `makeindex`.

xindy [3] is a much more extensible and robust indexing system than the `makeindex` system.

For example, one does not need to write:

```
\index{Lin@\textbf{Lin
```

}} to get the `Lin` entry after `LAN` and before `LZA`, instead, it's enough to write

```
\index{\textbf{Lin
```

}}

But what is much more important, it can properly sort index files in many languages, not only English.

Unfortunately, generating indices ready to use by `LaTeX` using `xindy` is a bit more complicated than with `makeindex`.

First, we need to know in what encoding the `.tex` project file is saved. In most cases it will be UTF-8 or ISO-8859-1, though if you live, for example in Poland it may be ISO-8859-2 or CP-1250. Check the parameter to the `inputenc` package.

Second, we need to know which language is prominently used in our document. `xindy` can natively sort indices in Albanian, Belarusian, Bulgarian, Croatian, Czech, Danish, Dutch, English, Esperanto, Estonian, Finnish, French, Georgian, German, Greek, Gypsy, Hausa, Hebrew, Hungarian, Icelandic, Italian, Klingon, Kurdish, Latin, Latvian, Lithuanian, Macedonian, Mongolian, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian Slovak, Slovenian, Sorbian, Spanish, Swedish, Turkish, Ukrainian and Vietnamese,

I don't know if other languages have similar problems, but with Polish, if your `.tex` is saved using UTF-8, the `.ind` produced by texindy will be encoded in ISO-8859-2 if you use only `-L polish`. While it's not a problem for entries containing polish letters, as LaTeX internally encodes all letters to plain ASCII, it is for accented letters at beginning of words, they create new index entry groups, if you have, for example an "średnia" entry, you'll get a "Ś" encoded in ISO-8859-2 `.ind` file. LaTeX doesn't like if part of the file is in UTF-8 and part is in IS-8859-2. The obvious solution (adding `-C utf8`) doesn't work, `texindy` stops with

```
ERROR: Could not find file "tex/inputenc/utf8.xdy"
```

error. The fix this, you have to load the definiton style for the headings using `-M switch`:

```
-M lang/polish/utf8
```

In the end we have to run such command:

```
texindy -L polish -M lang/polish/utf8 filename.idx
```

Additional way to fix this problem is use "iconv" to create utf8.xdy from latin2.xdy

```
 iconv -f latin2 -t utf8 latin2.xdy >utf8.xdy
```

in folder

```
  /usr/share/xindy/tex/inputenc
```

(You must have root privileges)

xindy in kile

To use `texindy` instead of `makeindex` in kile, you have to either redefine the MakeIndex tool in Settings → Configure Kile... → Tools → Build, or define new tool and redefine other tools to use it.

The `xindy` definition should look similar to this:

```
General:
 Command: texindy
 Options: -L polish -M lang/polish/utf8 -I latex '%S.idx'
Advanced:
 Type: Run Outside of Kile
 Class: Compile
 Source extension: idx
 Target extension: ind
 Target file: <empty>
 Relative dir: <empty>
 State: Editor
Menu:
 Add tool to Build menu: Compile
 Icon: the one you like
```

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/nomencl/
[2] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=multind
[3] http://xindy.sourceforge.net/

Algorithms and Pseudocode

LaTeX has a variety of packages that can help to format algorithms, code, and "pseudocode". These packages provide stylistic enhancements over a uniform style (i.e., typewriter fonts) so that constructs such as loops or conditionals are visually separated from other text.

Typesetting Algorithms

Typesetting using the `algorithmic` package

The `algorithmic` environment provides a number of popular constructs for algorithm designs. Put `\usepackage{algorithmic}` in the preamble to use the algorithmic environment to write algorithm pseudocode (`\begin{algorithmic}...\end{algorithmic}`). You might want to use the algorithm environment (`\usepackage{algorithm}`) to wrap your algorithmic code in an algorithm environment (`\begin{algorithm}...\end{algorithm}`) to produce a floating environment with numbered algorithms.

The command `\begin{algorithmic}` can be given the optional argument of a positive integer, which if given will cause line numbering to occur at multiples of that integer. E.g. `\begin{algorithmic}[5]` will enter the algorithmic environment and number every fifth line.

Below is an example of typesetting a basic algorithm using the `algorithms` package (remember to add the `\usepackage{algorithmic}` statement to your document preamble):

```
\begin{algorithmic}
\IF {$i\geq maxval$}
        \STATE $i\gets 0$
\ELSE
        \IF {$i+k\leq maxval$}
                \STATE $i\gets i+k$
        \ENDIF
\ENDIF
\end{algorithmic}
```

The LaTeX source can be written to a format familiar to programmers so that it is easy to read. This will not, however, affect the final layout in the document.

if $i \geq maxval$ **then**
  $i \leftarrow 0$
else
  if $i + k \leq maxval$ **then**
    $i \leftarrow i + k$
  end if
end if

There are several constructs provided by `algorithmic` detailed below

Single line statements

```
\STATE <text>
```

A simple statement, e.g. for setting a variable. For example,

```
\begin{algorithmic}
\STATE i=0
\end{algorithmic}
```

would produce

```
i = 0
```

If-statements

There are three forms of this construct

```
\IF{<condition>} <text> \ENDIF
```

```
\IF{<condition>} <text> \ELSE <text> \ENDIF
```

```
\IF{<condition>} <text> \ELSIF{<condition>} <text> \ELSE <text> \ENDIF
```

The third form accepts as many \ELSIF{ } clauses as required.

For-loops

There are two forms

```
\FOR{<condition>} <text> \ENDFOR
```

```
\FORALL{<condition>} <text> \ENDFOR
```

A traditional "for" loop. The method of iteration is usually described in the first argument,

e.g.

```
\FOR{$i = 1 \to 10$}
\STATE $i \gets i + 1$
\ENDFOR
```

While-loops

```
\WHILE{<condition>} <text> \ENDWHILE
```

```
\WHILE{$i \leq 10$}
    \STATE i=i+1;
\ENDWHILE
```

Repeat until condition

```
\REPEAT <text> \UNTIL{<condition>}
```

Infinite loops

```
\LOOP <text> \ENDLOOP
```

Precondition

```
\REQUIRE <text>
```

Postcondition

```
\ENSURE <text>
```

Returning variables

```
\RETURN <text>
```

Printing variables

```
\PRINT <text>
```

This is included because it is used so frequently it is considered an operation in its own right.

Comments

```
\COMMENT{<text>}
```

Note that you cannot use \COMMENT as the first statement of any closed structure, such as \IF..\ENDIF, \FOR..\ENDFOR, \FORALL..\ENDFORALL, \WHILE..\ENDWHILE, and \begin{algorithmic}..\end{algorithmic}. An error "LaTeX Error: Something's wrong--perhaps a missing \item" will be reported (It does not make much sense). There are two workarounds:

1. Use \STATE \COMMENT{<text>}.
2. Use the optional arguments in those closed structures. For example, \WHILE[<comment-text>]{<condition>}. To use math in comment text, replace $..$ by \ensuremath{..}

Compatibility with hyperref

Due to a bug, the algorithmic package is not compatible with hyperref. A workaround is the algorithmic-fix [1] package. Copy the code found on the linked page to a file called algorithmic-fix.sty and include it with \usepackage{algorithmic,algorithmic-fix}. However, if this trick fails, try using \usepackage{hyperref} before using \usepackage{algorithmic}. In this case, you might not even need the algorithmic-fix.sty.

Renaming things: algorithm to procedure, require/ensure to input/output

```
\floatname{algorithm}{Procedure}
\renewcommand{\algorithmicrequire}{\textbf{Input:}}
\renewcommand{\algorithmicensure}{\textbf{Output:}}
```

The `algorithm` environment

It is often useful for the algorithm produced by `algorithmic` to be "floated" to the optimal point in the document to avoid it being split across pages. The `algorithm` environment provides this and a few other useful features. Include it by adding the
`\usepackage{algorithm}` to your document's preamble. It is entered into by

```
\begin{algorithm}
\caption{<your caption for this algorithm>}
\label{<your label for references later in your document>}
\begin{algorithmic}
<algorithmic environment>
\end{algorithmic}
\end{algorithm}
```

Algorithm numbering

The default numbering system for the `algorithm` package is to number algorithms sequentially. This is often not desirable, particularly in large documents where numbering according to chapter is more appropriate. The numbering of algorithms can be influenced by providing the name of the document component within which numbering should be recommenced. The legal values for this option are: part, chapter, section, subsection, subsubsection or nothing (default). For example:

```
\usepackage[chapter]{algorithm}
```

List of algorithms

When you use figures or tables, you can add a list of them close to the table of contents; the `algorithm` package provides a similar command. Just put

```
\listofalgorithms
```

anywhere in the document, and LaTeX will print a list of the "algorithm" environments in the document with the corresponding page and the caption.

An example from the manual

This is an example taken from the manual (official manual, p.7)

```
\begin{algorithm}                       % enter the algorithm
environment
\caption{Calculate $y = x^n$}           % give the algorithm a caption
\label{alg1}                            % and a label for \ref{}
commands later in the document
\begin{algorithmic}                     % enter the algorithmic
environment
\REQUIRE $n \geq 0 \vee x \neq 0$
\ENSURE $y = x^n$
\STATE $y \Leftarrow 1$
\IF{$n < 0$}
\STATE $X \Leftarrow 1 / x$
\STATE $N \Leftarrow -n$
\ELSE
```

```
\STATE $X \Leftarrow x$
\STATE $N \Leftarrow n$
\ENDIF
\WHILE{$N \neq 0$}
\IF{$N$ is even}
\STATE $X \Leftarrow X \times X$
\STATE $N \Leftarrow N / 2$
\ELSE[$N$ is odd]
\STATE $y \Leftarrow y \times X$
\STATE $N \Leftarrow N - 1$
\ENDIF
\ENDWHILE
\end{algorithmic}
\end{algorithm}
```

More information about all possible commands available at the project page

http://developer.berlios.de/docman/?group_id=3442

The official manual is located at

http://developer.berlios.de/docman/display_doc.php?docid=800&group_id=3442

Typesetting using the `program` package

The `program` package provides macros for typesetting algorithms. Each line is set in math mode, so all the indentation and spacing is done automatically. The notation `|variable_name|` can be used within normal text, maths expressions or programs to indicate a variable name. Use `\origbar` to get a normal `|` symbol in a program. The commands `\A`, `\B`, `\P`, `\Q`, `\R`, `\S`, `\T` and `\Z` typeset the corresponding bold letter with the next object as a subscript (eg `\S1` typesets `{\bf S$_1$}` etc). Primes work normally, eg `\S`'.

Below is an example of typesetting a basic algorithm using the `program` package (remember to add the `\usepackage{program}` statement to your document preamble):

```
\begin{program}
\mbox{A fast exponentiation procedure:}
\BEGIN %
  \FOR i:=1 \TO 10 \STEP 1 \DO
     |expt|(2,i); \\ |newline|() \OD %
\rcomment{This text will be set flush to the right margin}
\WHERE
\PROC |expt|(x,n) \BODY
          z:=1;
          \DO \IF n=0 \THEN \EXIT \FI;
             \DO \IF |odd|(n) \THEN \EXIT \FI;
\COMMENT{This is a comment statement};
                n:=n/2; x:=x*x \OD;
             \{ n>0 \};
             n:=n-1; z:=z*x \OD;
          |print|(z) \ENDPROC
\END
\end{program}
```

A fast exponentiation procedure:
begin
  for $i := 1$ **to** 10 **step** 1 **do**
    expt$(2, i)$;
    newline() **od** This text will be set flush to the right margin
where
proc expt(x, n) $\equiv$
  $z := 1$;
  do if $n = 0$ **then exit fi**;
    do if odd(n) **then exit fi**;
      comment: This is a comment statement;
      $n := n/2$; $x := x * x$ **od**;
    $\{n > 0\}$;
    $n := n - 1$; $z := z * x$ **od**;
  print(z).
end

The commands `\(` and `\)` are redefined to typeset an algorithm in a minipage, so an algorithm can appear as a single box in a formula. For example, to state that a particular action system is equivalent to a WHILE loop you can write:

```
\[
\( \ACTIONS A:
        A \EQ \IF \B{} \THEN \S{}; \CALL A
                       \ELSE \CALL Z \FI \QE
   \ENDACTIONS \)
\EQT
\( \WHILE \B{} \DO \S{} \OD \)
\]
```

Dijkstra conditionals and loops:

```
\begin{program}
\IF x = 1 \AR y:=y+1
\BAR x = 2 \AR y:=y^2
\utdots
\BAR x = n \AR y:=\displaystyle\sum_{i=1}^n y_i \FI

\DO 2 \origbar x \AND x>0 \AR x:= x/2
\BAR \NOT 2 \origbar x    \AR x:= \modbar{x+3} \OD
\end{program}
```

Loops with multiple exits:

```
\begin{program}
\DO \DO \IF \B1 \THEN \EXIT \FI;
        \S1;
        \IF \B2 \THEN \EXIT(2) \FI \OD;
    \IF \B1 \THEN \EXIT \FI \OD
\end{program}
```

A Reverse Engineering Example.

Here's the original program:

```
\begin{program}
 \VAR \seq{m := 0, p := 0, |last| := `` ''};
 \ACTIONS |prog|:
|prog| \ACTIONEQ %
    \seq{|line| := `` '', m := 0, i := 1};
    \CALL |inhere| \ENDACTION
l \ACTIONEQ %
    i := i+1;
    \IF (i=(n+1)) \THEN \CALL |alldone| \FI ;
    m := 1;
    \IF |item|[i] \neq |last|
        \THEN |write|(|line|); |line| := `` ''; m := 0;
              \CALL |inhere| \FI ;
    \CALL |more| \ENDACTION
|inhere| \ACTIONEQ %
    p := |number|[i]; |line| := |item|[i];
    |line| := |line| \concat `` '' \concat p;
    \CALL |more| \ENDACTION
|more| \ACTIONEQ %
    \IF (m=1) \THEN p := |number|[i];
    |line| := |line| \concat ``, '' \concat p \FI ;
    |last| := |item|[i];
    \CALL l  \ENDACTION
|alldone| \ACTIONEQ |write|(|line|); \CALL Z \ENDACTION \ENDACTIONS
\END
\end{program}
```

And here's the transformed and corrected version:

```
\begin{program}
\seq{|line| := `` '', i := 1};
\WHILE i \neq n+1 \DO
  |line| := |item|[i] \concat `` '' \concat |number|[i];
  i := i+1;
  \WHILE i \neq n+1 \AND |item|[i] = |item|[i-1] \DO
    |line| := |line| \concat ``, '' \concat |number|[i]);
    i := i+1 \OD ;
  |write|(|line|) \OD
\end{program}
```

The package also provides a macro for typesetting a set like this: `\set{x \in N | x > 0}`.

Lines can be numbered by setting `\NumberProgramstrue` and numbering turned off with `\NumberProgramsfalse`

Package page [2]

Package documentation [3]

Source Code Formatting using the `Listings` package

(See the Listings package reference page for more information.)

A complete reference manual can be found at http://tug.ctan.org/tex-archive/macros/latex/contrib/listings/listings.pdf

This is a basic example for some Pascal code:

```
\documentclass{article}
\usepackage{listings}              % Include the listings-package
\begin{document}
\lstset{language=Pascal}           % Set your language (you can change
the language for each code-block optionally)

\begin{lstlisting}[frame=single]                % Start your code-block
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive ');
Write('Pascal keywords.');
\end{lstlisting}

\end{document}
```

```
begin
{ do nothing }
end;
Write(Case insensitive );
Write(Pascal keywords.);
```

References

- The official manual for the algorithms package, Rogério Brito (2009), http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf

References

[1] http://mrunix.de/forums/showpost.php?s=f85d42fd6f15e8f112bbc31d94d21424&p=258363&postcount=6

[2] http://www.ctan.org/pkg/program

[3] http://mirror.ctan.org/macros/latex/contrib/program/program-doc.pdf

Pictures and Graphics

Importing Graphics

Strictly speaking, LaTeX cannot manage pictures directly: in order to introduce graphics within documents, LaTeX just creates a box with the same size as the image you want to include and embeds the picture, without any other processing. This means you will have to take care that the images you want to include are in the right format to be included. This is not such a hard task because LaTeX supports the most common picture formats around.

The `graphicx` package

As stated before, LaTeX can't manage pictures directly, so we will need some extra help: we have to load the `graphicx` package in the preamble of our document:

```
\usepackage{graphicx}
```

This package accepts as an argument the external driver to be used to manage pictures; however, the latest version of this package takes care of everything by itself, changing the driver according to the compiler you are using, so you don't have to worry about this. Still, just in case you want to understand better how it works, here are the possible options you can pass to the package:

- `dvips` (default if compiling with *latex*), if you are compiling with *latex* to get a DVI and you want to see your document with a DVI or PS viewer.
- `dvipdfm`, if you are compiling with *latex* to get a DVI that you want to convert to PDF using *dvipdfm*, to see your document with any PDF viewer.
- `pdftex` (default if compiling with *pdflatex*), if you are compiling with *pdftex* to get a PDF that you will see with any PDF viewer.

but, again, you don't need to pass any option to the package because the default settings are fine in most of the cases.

In many respects, importing your images into your document using LaTeX is fairly simple... *once* you have your images in the right format that is! Therefore, I fear for many people the biggest effort will be the process of converting their graphics files. Now we will see which formats we can include and then we will see how to do it.

Document Options

The graphics and graphicx packages recognize the "draft" and "final" options given in the `\documentclass[...]{...}` command at the start of the file. Using "draft" as the option will suppress the inclusion of the image in the output file and will replace the contents with the name of the image file that would have been seen. Using "final" will result in the image being placed in the output file. The default is "draft".

Supported image formats

As explained before, the image formats you can use depend on the driver that `graphicx` is using but, since the driver is automatically chosen according to the compiler, then the allowed image formats will depend on the compiler you are using.

Compiling with *latex*

The only format you can include while compiling with *latex* is Encapsulated PostScript (**EPS**).

The EPS format was defined by Adobe Systems for making it easy for applications to import postscript-based graphics into documents. Because an EPS file declares the size of the image, it makes it easy for systems like LaTeX to arrange the text and the graphics in the best way. EPS is a vector format—this means that it can have very high quality if it is created properly, with programs that are able to manage vector graphics. It is also possible to store bit-map pictures within EPS, but they will need *a lot* of disk space.

Many graphics software packages have the ability to save images in the EPS format (extension is normally `.eps`). Here are some examples of software that can output EPS formats:

- Printing in an EPS file:
  - Under Windows, PDFCreator [1] is an open source software that can create PDF as well as EPS files. It installs a virtual printer that can be accessed from other software having a "print..." entry in their menu (virtually any program).
- Creating and converting vector graphics:
  - Commercial vector graphics software, such as Adobe Illustrator, CorelDRAW, and FreeHand are commonly used and can *read* and *write* EPS figures. However, these products are limited to Windows and Mac OS platforms.
  - Inkscape [2] can save in vector EPS format, and it can run on multiple platforms. Inkscape cannot open EPS figures directly; however, with the epstopdf utility [3] one can convert EPS into PDF and Inkscape can import PDF. From version 0.48, Inkscape has a special PDF+LaTeX output option (and for EPS/PS too). See Inkscape website [4].
  - Dia [5] is a cross platform diagramming utility which can export eps.
- Creating and converting raster-only graphics to EPS:
  - GIMP [6], has a graphical user interface, and it is multi-platform.
  - For command-line:
    - Sam2p [7] (`convert`) or
    - ImageMagick [8] (`convert`) or
    - GraphicsMagick [9] (`gm convert`).
  - These three programs operate much the same way, and can convert between most graphics formats. Sam2p however is the most recent of the three and seems to offer both the best quality and to result in the smallest files.
    - imgtops [10]. A lightweight graphics utility.
- Creating publication-quality vector-based plots and charts:
  - Gnuplot [11], producing scientific graphics since 1986.
  - R [12], statistical and scientific figures.
  - Generic Mapping Tools (GMT) [13], maps and a wide range of highly customisable plots.
  - Gnumeric [14], spreadsheets has SVG, EPS, PDF export
  - matplotlib [15], plotting library written in python, with PDF and EPS export.

There are some tricks to be able to import formats other than EPS into your DVI document, but they're very complicated. On the other hand, converting any image to EPS is very simple, so it's not worth considering them.

Compiling with `pdflatex`

If you are compiling with `pdflatex` to produce a PDF, you have a wider choice. You can insert:

- **JPG**, widely used on Internet, digital cameras, etc. They are the best choice if you want to insert photos
- **PNG**, a very common format (even if not as much as JPG); it's a lossless format and it's the best choice for diagrams (if you were not able to generate a vector version) and screenshots
- **PDF**, it is widely used for documents but can be used to store images as well. It supports both vector and bit-map images, but it's not recommended for the latter, as JPG or PNG will provide the same result using less disk space.
- **Vector formats** can be used with the help of Inkscape. There are instructions [16] on how to save your vector images in a PDF format understood by LaTeX and have LaTeX manage the text styles and sizes in the image automatically.
- **EPS** can be used with the help of the epstopdf package. Please see these instructions [17].

JPG and PNG are supported by any image processing program, so you just have to use the one you prefer. If you want to create high quality vector PDF to embed within your PDF document, you can use Inkscape [2]: it supports many vector formats and so you can use it to convert from one to an other. You could also create your graphics directly with Inkscape. If you want to make mathematical plots, then Gnuplot [11] can save in any format.

Note, that EPS files cannot be used with pdflatex, however they can be converted to PDF using the epstopdf utility [3], included in most LaTeX distributions. This can be called automatically by LaTeX using the epstopdf package [18]. In Windows, multiple files can be converted by placing the following line in a batch file (a text file with a .BAT extension) in the same directory as the images:

```
for %%f in (*.eps) do epstopdf %%f
```

which can then be run from the command line. If epstopdf [3] produces whole page with your small graphics somewhere on it, use

```
$ epstopdf --gsopt=-dEPSCrop foo.eps
```

or try using ps2pdf [19] utility

```
$ ps2pdf -dEPSCrop foo.eps
```

to crop final PDF.

Images can be saved in multiple formats for different purposes. For example, a directory can have "`diagram.pdf`" for high-resolution printing, while "`diagram.png`" can be used for previewing on the monitor. You can specify which image file is to be used by `pdflatex` through the preamble command:

```
\DeclareGraphicsExtensions{.pdf,.png,.jpg}
```

which specifies the files to include in the document, if files with the same basename exist, but with different extensions.

Acrobat Reader sometimes has problems with displaying colors correctly if you include graphics in PNG format with alpha channel. You can solve this problem by dropping the alpha channel. On Linux it can be achieved with `convert` from the ImageMagick program:

```
convert -alpha off input.png output.png
```

Including graphics

Now that we have seen which formats we can include and how we could manage those formats, it's time to learn how to include them in our document. After you have loaded the `graphicx` package in your preamble, you can include images with `\includegraphics`, whose syntax is the following:

```
\includegraphics[attr1=val1, attr2=val2, ..., attrn=valn]{imagename}
```

As you should hopefully be aware by now, arguments in square brackets are optional, whereas arguments in curly braces are compulsory. The argument in the curly braces is the name of the image. Write it *without* the extension. This way the LaTeX compiler will look for any supported image format in that directory and will take the best one (EPS if the output is DVI; JPEG, PNG or PDF if the output is PDF). The variety of possible attributes that can be set is fairly large, so only the most common are covered below:

`width=xx`	Specify the preferred width of the imported image to *xx*.	*NB. Only specifying either width or height will scale the image whilst maintaining the aspect ratio.*
`height=xx`	Specify the preferred height of the imported image to *xx*.	
`keepaspectratio`	This can be set to either *true* or *false*. When true, it will scale the image according to both height and width, but will not distort the image, so that neither width nor height are exceeded.	
`scale=xx`	Scales the image by the desired scale factor. e.g, 0.5 to reduce by half, or 2 to double.	
`angle=xx`	This option can rotate the image by *xx* degrees (counter-clockwise)	
`trim=l b r t`	This option will crop the imported image by *l* from the left, *b* from the bottom, *r* from the right, and *t* from the top. Where l, b, r and t are lengths.	
`clip`	For the `trim` option to work, you must set `clip=true`.	
`page=x`	If the image file is a pdf file with multiple pages, this parameter allows you to use a different page than the first.	

In order to use more than one option at a time, simply separate each with a comma. The order you give the options matters. E.g you should first rotate your graphic (with angle) and then specify its width.

Included graphics will be inserted just *there*, where you placed the code, and the compiler will handle them as "big boxes". As we will see in the next section, this can disrupt the layout; you'll probably want to place graphics inside floating objects.

Also note that the trim option does not work with XeLaTex.

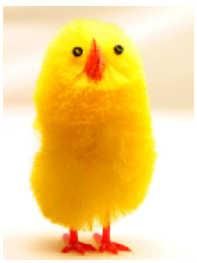
Examples

OK, it's time to see graphicx in action. Here are some examples:

```
\includegraphics{chick}
```

This simply imports the image, without any other processing. However, it is very large (so I won't display it here!). So, let's scale it down:

```
\includegraphics[scale=0.5]{chick}
```



This has now reduced it by half. If you wish to be more specific and give actual lengths of the image dimensions, this is how to go about it:

```
\includegraphics[width=2.5cm]{chick}
```

One can also specify the scale with respect to the width of a line in the local environment (`\linewidth`), the width of the text on a page (`\textwidth`) or the height of the text on a page (`\textheight`) (pictures not shown):

```
\includegraphics[width=\linewidth]{chick}

\includegraphics[width=\textwidth]{chick}

\includegraphics[height=\textheight]{chick}
```

To rotate (I also scaled the image down):

```
\includegraphics[scale=0.5, angle=180]{chick}
```

And finally, an example of how to crop an image should you wish to focus on one particular area of interest:

```
%trim option's parameter order: left bottom right top
\includegraphics[trim = 10mm 80mm 20mm 5mm, clip, width=3cm]{chick}
```

Note the presence of `clip`, as the trim operation will not work without it.

As you may have noticed, the file name of the picture is always without the extension: LaTeX will take care of getting the right version for us. Consider the following situation: you have added some pictures to your document in JPG and you have successfully compiled it in PDF. Now you want to compile it in DVI, you run *latex* and you get a lot of errors... because you forgot to provide the EPS versions of the pictures you want to insert. At the beginning of this book, we had stated that the same LaTeX source can be compiled in both DVI and PDF without any change. This is true, as long as you don't use particular packages, and `graphicx` is one of those. In any case, you can still use both compilers with documents with pictures as well, as long as you always remember to provide the pictures in two formats (EPS and one of JPG, PNG or PDF).

Borders

It is possible to have LaTeX create a border around your image by using `fbox`:

```
\setlength\fboxsep{0pt}
\setlength\fboxrule{0.5pt}
\fbox{\includegraphics{chick}}
```

You can control the border padding with the `\setlength\fboxsep{0pt}` command, in this case I set it to 0pt to avoid any padding, so the border will be placed tightly around the image. You can control the thickness of the border by adjusting the `\setlength\fboxrule{0.5pt}` command.

Graphics storage

There is a way to tell LaTeX where to look for images: for example, it can be useful if you store images centrally for use in many different documents. The answer is in the command `\graphicspath` which you supply with an argument giving the name of an additional directory path you want searched when a file uses the `\includegraphics` command, here are some examples (trailing / is required):

```
\graphicspath{{c:\mypict~1\camera}}
\graphicspath{{c:/mypict~1/camera/}} *
\graphicspath{{/var/lib/images/}}
\graphicspath{{images_folder/}{other_folder/}{third_folder/}}
\graphicspath{{./images/}}
```

* works well in Win XP

Please see http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.pdf. In the last example shown you would have a directory named "images" in the same directory as your man tex file, i.e. this is RELATIVE addressing.

As you may have noticed, in the first example I've used the "safe" (MS-DOS) form of the Windows *MyPictures* folder because it's a bad idea to use directory names containing spaces. Using absolute paths, `\graphicspath` does make your file less portable, while using relative paths (like the last example), you shouldn't have any problem with portability, but remember not to use spaces in file-names. Alternatively, if you are using PDFLaTeX, you can use the package `grffile` which will then allow you to use spaces in file names.

Note that you cannot make the graphicx package search directories recursively. Under Linux/Unix, you can achieve a recursive search using the environment variable `TEXINPUTS`, e.g., by setting it to

```
export TEXINPUTS=./images//:./Snapshots//
```

before running latex/pdflatex or your TeX-IDE. (But this, of course, is not a portable method.)

Images as Figures

There are many scenarios where you might want to accompany an image with a caption and possibly a cross-reference. This is done using the figure environment. The following code sample shows the bare minimum required to use an image as a figure.

```
\begin{figure}[htb]
\includegraphics{image.png}
\end{figure}
```

The above code extract is relatively trivial, and doesn't offer much functionality. The following code sample shows an extended use of the figure environment which is almost universally useful, offering a caption and label, centering the image and scaling it to 80% of the width of the text.

```
\begin{figure}[htb]
\centering
\includegraphics[width=0.8\textwidth]{image.png}
\caption{Awesome Image}
\label{fig:awesome_image}
\end{figure}
```

The figure environment is not exclusively used for images. More information on the figure environment and how to use it can be found in Floats, Figures and Captions.

Text Wrapping around Images

Text can also be wrapped around images. (This is especially useful if you include tall pictures.)

```
%import section
\usepackage{wrapfig}

% content section
\begin{wrapfigure}{r}{8cm} % "l" or "r" for the side on the page. And
the width parameter for the width of the image space.
\centering
\includegraphics[height=80mm]{Abb/bluesniper.jpg}
\caption{Selbstgebaute „Bluesniper“ um Bluetooth-Geräte aus über 1 km
Entfernung anzugreifen. (Stand: 2004)}
\label{bluesniper}
\end{wrapfigure}
```

Including full PDF pages

There is a great package for including full pages of PDF files: pdfpages [20]. It is capable of inserting full pages as is and more pages per one page in any layout (e.g. 2x3). See more information in its documentation [21].

Creating Vector Graphics

TikZ/PGF

More thorough introduction to TikZ is available at the Creating Graphics chapter

You can draw graphics directly with TeX commands using the tikz package: http://ftp.dante.de/tex-archive/help/Catalogue/entries/pgf.html It comes with very good documentation with many examples.

```
% This needs \usepackage{tikz} in the preamble
\begin{figure}
  \centering
  \begin{tikzpicture}
    \draw[thick,rounded corners=8pt]
(0,0) -- (0,2) -- (1,3.25) -- (2,2) -- (2,0) -- (0,2) -- (2,2) -- (0,0)
 -- (2,0);
  \end{tikzpicture}
  \caption{This is the caption of my figure}
  \label{fig:test}
\end{figure}
```

An extensive collection of examples can be found here: http://www.texample.net/tikz/

Further packages which are based on TikZ (e.g. for drawing electrical circuits), can be found here: http://ftp.dante.de/tex-archive/help/Catalogue/bytopic.html#pgftikzsection

Xfig

Vector graphics can be created using the vector painting program Xfig (see Installation), and exported for LaTeX. In Xfig, once your graphic is saved as a file `test.fig`, you need to export it using the **File > Export** drop down menu from the main Xfig window and then select the "Combined PS/Latex (both parts)" in the language drop down list. If you don't change any other settings, two files will be created in the same directory as the `test.fig` file, such as: `test.pstex_t` and `test.pstex`. The figure can then be placed in a LaTeX document:

```
\begin{figure}
  \centering
  \input{./xfig/test.pstex_t}
  \caption{This is the caption of my figure}
  \label{fig:test}
\end{figure}
```

ipe

The Ipe extensible drawing editor is a free vector graphics editor for creating figures in PDF or EPS format. Unlike Xfig, ipe represents LaTeX fonts in their correct size on the screen which makes it easier to place text labels at the right spot. ipe also has various snapping modes (for example, snapping to points, lines, or intersections) that can be used for geometric constructions.

Inkscape

Another program for creating vector graphics is Inkscape [22]. It works with Scalable Vector Graphics (SVG) [23] files, although it can export to many formats that can be included in LaTeX files, such as EPS and PDF. From version 0.48, there is a combined PDF/EPS/PS+LaTeX output option, like XFig has.

Editing EPS graphics

As described above, graphics content can be imported into LaTeX from outside programs as EPS files. But sometimes you want to edit or retouch these graphics files. An EPS file can be edited with any text editor since it is formatted as ASCII. In a text editor, you can achieve simple operations like replacing strings or moving items slightly, but anything further becomes cumbersome.

To properly edit an EPS file, you can convert it to an *editable* format using pstoedit [24]. For instance, to get an Xfig-editable file, do:

```
$ pstoedit -f fig input.eps output.fig
```

And to get an SVG file for Inkscape you can do:

```
$ pstoedit -f plot-svg input.eps output.svg
```

Sometimes pstoedit fails to create the target format (for example when the EPS file contains clipping information). A more robust way to edit EPS files is achieved by converting it first to PDF and then importing the resulting PDF in Inkscape. Inkscape uses the Cairo library that achieves a high-quality transformation of the original EPS figure:

```
$ epstopdf input.eps
$ inkscape input.pdf
```

When all of the above fails, one can simplify the EPS file before attempting other conversions, by using the eps2eps [25] tool (also see next section):

```
$ eps2eps input.eps input-e2.eps
```

This will convert all the fonts to pre-drawn images, which is sometimes desirable when submitting manuscripts for publication. However, on the downside, the fonts are NOT converted to lines, but instead to bitmaps, which reduces the quality of the fonts.

Converting a color EPS to grayscale

Sometimes color EPS figures need to be converted to black-and-white or grayscale to meet publication requirements. This can be achieved with the eps2eps [25] of the Ghostscript [26] package and [27] programs:

```
$ eps2eps input.eps input-e2.eps
$ pscol -0gray input-e2.eps input-gray.eps
```

References

[1] http://sourceforge.net/projects/pdfcreator/
[2] http://www.inkscape.org
[3] http://www.ctan.org/tex-archive/support/epstopdf/
[4] http://wiki.inkscape.org/wiki/index.php/LaTeX
[5] http://live.gnome.org/Dia
[6] http://www.gimp.org
[7] http://pts.szit.bme.hu/sam2p/
[8] http://www.imagemagick.org/
[9] http://www.graphicsmagick.org/
[10] http://imgtops.sourceforge.net/
[11] http://www.gnuplot.info
[12] http://www.r-project.org/
[13] http://gmt.soest.hawaii.edu/
[14] http://projects.gnome.org/gnumeric/
[15] http://matplotlib.sourceforge.net/
[16] http://mirrors.ctan.org/info/svg-inkscape/InkscapePDFLaTeX.pdf
[17] http://dirkraffel.com/2007/11/19/include-eps-files-in-latex
[18] http://www.ctan.org/tex-archive/help/Catalogue/entries/epstopdf-pkg.html
[19] http://svn.ghostscript.com/ghostscript/trunk/gs/doc/Ps2pdf.htm
[20] http://www.ctan.org/tex-archive/macros/latex/contrib/pdfpages
[21] http://www.ctan.org/tex-archive/macros/latex/contrib/pdfpages/pdfpages.pdf
[22] http://www.inkscape.org/
[23] http://www.w3.org/Graphics/SVG/
[24] http://www.pstoedit.net/
[25] http://linuxcommand.org/man_pages/eps2eps1.html
[26] http://ghostscript.com/
[27] http://www.pa.op.dlr.de/~PatrickJoeckel/pscol/index.html

Creating Graphics

In the previous chapter, you learned that you can import or link graphics into LaTeX, such as graphics that you have created in another program or obtained elsewhere. In this chapter, you will learn how to create or embed graphics directly in a LaTeX document. The graphics is marked up using commands similar to those for typesetting bold text or creating mathematical formulas, as the following example of embedded graphics shows:

```
\begin{displaymath}
    \xymatrix{ \bullet \ar[r] \ar@{.>}[r] & \bullet }
\end{displaymath}
```

There are several packages supporting the creation of graphics directly in LaTeX, including `picture`, `xy-Pic`, and `PGF/TikZ`, described in the following sections.

Overview

The `picture` environment allows programming pictures directly in LaTeX. On the one hand, there are rather severe constraints, as the slopes of line segments as well as the radii of circles are restricted to a narrow choice of values. On the other hand, the picture environment of LaTeX2e brings with it the `\qbezier` command, "q" meaning *quadratic*. Many frequently-used curves such as circles, ellipses, and catenaries can be satisfactorily approximated by quadratic Bézier curves, although this may require some mathematical toil. If a programming language like Java is used to generate `\qbezier` blocks of LaTeX input files, the picture environment becomes quite powerful.

Although programming pictures directly in LaTeX is severely restricted, and often rather tiresome, there are still reasons for doing so. The documents thus produced are "small" with respect to bytes, and there are no additional graphics files to be dragged along.

Packages like `epic`, `eepic` or `pstricks` enhance the original picture environment, and greatly strengthen the graphical power of LaTeX.

While the former two packages just enhance the picture environment, the `pstricks` package has its own drawing environment, `pspicture`. The power of `pstricks` stems from the fact that this package makes extensive use of PostScript possibilities. Unfortunately it has one big shortcoming: it doesn't work together with pdfLaTeX, as such to generate a PDF document form TeX source you have to go TeX→DVI→PDF; losing hyperlinks, metadata and microtypographic features of pdflatex. In addition, numerous packages have been written for specific purposes. One of them is *XY-pic,* described at the end of this chapter. A wide variety of these packages is described in detail in *The LaTeX Graphics Companion* (not to be confused with *The LaTeX Companion*).

Perhaps the most powerful graphical tool related with LaTeX is MetaPost, the twin of Donald E. Knuth's METAFONT. MetaPost has the very powerful and mathematically sophisticated programming language of METAFONT. Contrary to METAFONT, which generates bitmaps, MetaPost generates encapsulated PostScript files, which can be imported in LaTeX. For an introduction, see *A User's Manual for MetaPost.* A very thorough discussion of LaTeX and TEX strategies for graphics (and fonts) can be found in *TEX Unbound.*

The last but certainly not least is the PGF/TikZ system. While the previous systems (`picture`, `epic`, `pstricks` or `metapost`) focus on the *how* to draw, TikZ focuses more on the *what* to draw. One could say that TikZ is to drawing in LaTeX as LaTeX is to digital typesetting. It's recommended to use it if your LaTeX distribution includes it.

The picture Environment

Basic Commands

A `picture` environment is available in any LaTeX distribution, without the need of loading any external package. This environment is created with one of the two commands

```
\begin{picture}(x, y) ... \end{picture}
```

or

```
\begin{picture}(x, y)(x0, y0) ... \end{picture}
```

The numbers *x, y, x0, y0* are numbers (lengths) in the units of `\unitlength`, which can be reset any time (but not within a picture environment) with a command such as

```
\setlength{\unitlength}{1.2cm}
```

The default value of `\unitlength` is `1pt`. The first pair, (x, y), effects the reservation, within the document, of rectangular space for the picture. The optional second pair, (x_0, y_0), assigns arbitrary coordinates to the bottom left corner of the reserved rectangle.
Most drawing commands have one of the two forms

```
\put(x, y){object}
```

or

```
\multiput(x, y)(dx, dy){n}{object}
```

Bézier curves are an exception. They are drawn with the command

```
\qbezier(x1, y1)(x2, y2)(x3, y3)
```

Line Segments

Line segments are drawn with the command:

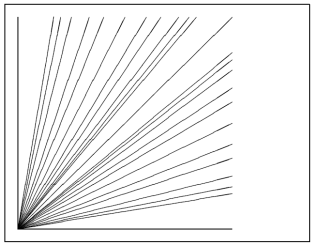
```
\put(x, y){\line(x1, y1){length}}
```

The `\line` command has two arguments:

1. a direction vector,
2. a "length" (sort of: this argument is the length of the vertical coordinate in the case of a vertical line segment and of the horizontal coordinate in all other cases, rather than the length of the segment itself).

The components of the direction vector are restricted to the integers (*–6, –5, ... , 5, 6*) and they have to be coprime (no common divisor except 1). The figure below illustrates all 25 possible slope values in the first quadrant. The length is relative to `\unitlength`.

```
\setlength{\unitlength}{5cm}
\begin{picture}(1,1)
\put(0,0){\line(0,1){1}}
\put(0,0){\line(1,0){1}}
\put(0,0){\line(1,1){1}}
\put(0,0){\line(1,2){.5}}
\put(0,0){\line(1,3){.3333}}
\put(0,0){\line(1,4){.25}}
\put(0,0){\line(1,5){.2}}
\put(0,0){\line(1,6){.1667}}
\put(0,0){\line(2,1){1}}
\put(0,0){\line(2,3){.6667}}
\put(0,0){\line(2,5){.4}}
\put(0,0){\line(3,1){1}}
\put(0,0){\line(3,2){1}}
\put(0,0){\line(3,4){.75}}
\put(0,0){\line(3,5){.6}}
\put(0,0){\line(4,1){1}}
\put(0,0){\line(4,3){1}}
\put(0,0){\line(4,5){.8}}
\put(0,0){\line(5,1){1}}
\put(0,0){\line(5,2){1}}
\put(0,0){\line(5,3){1}}
\put(0,0){\line(5,4){1}}
\put(0,0){\line(5,6){.8333}}
\put(0,0){\line(6,1){1}}
\put(0,0){\line(6,5){1}}
\end{picture}
```



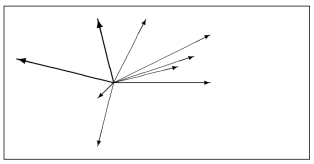
Arrows

Arrows are drawn with the command

```
\put(x, y){\vector(x1, y1){length}}
```

For arrows, the components of the direction vector are even more narrowly restricted than for line segments, namely to the integers (*–4, –3, ... , 3, 4*). Components also have to be coprime (no common divisor except 1). Notice the effect of the `\thicklines` command on the two arrows pointing to the upper left.

```
\setlength{\unitlength}{0.75mm}
\begin{picture}(60,40)
\put(30,20){\vector(1,0){30}}
\put(30,20){\vector(4,1){20}}
\put(30,20){\vector(3,1){25}}
\put(30,20){\vector(2,1){30}}
\put(30,20){\vector(1,2){10}}
\thicklines
\put(30,20){\vector(-4,1){30}}
\put(30,20){\vector(-1,4){5}}
\thinlines
\put(30,20){\vector(-1,-1){5}}
\put(30,20){\vector(-1,-4){5}}
\end{picture}
```



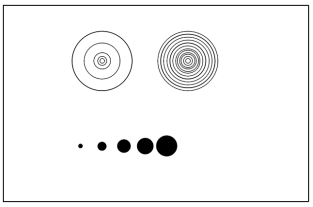
Circles

The command

```
\put(x, y){\circle{diameter}}
```

draws a circle with center (x, y) and diameter (not radius) specified by *diameter*. The picture environment only admits diameters up to approximately 14mm, and even below this limit, not all diameters are possible. The `\circle*` command produces disks (filled circles). As in the case of line segments, one may have to resort to additional packages, such as `eepic` or `pstricks`.

```
\setlength{\unitlength}{1mm}
\begin{picture}(60, 40)
\put(20,30){\circle{1}}
\put(20,30){\circle{2}}
\put(20,30){\circle{4}}
\put(20,30){\circle{8}}
\put(20,30){\circle{16}}
\put(20,30){\circle{32}}
\put(40,30){\circle{1}}
\put(40,30){\circle{2}}
\put(40,30){\circle{3}}
\put(40,30){\circle{4}}
\put(40,30){\circle{5}}
\put(40,30){\circle{6}}
\put(40,30){\circle{7}}
\put(40,30){\circle{8}}
\put(40,30){\circle{9}}
\put(40,30){\circle{10}}
\put(40,30){\circle{11}}
\put(40,30){\circle{12}}
\put(40,30){\circle{13}}
\put(40,30){\circle{14}}
\put(15,10){\circle*{1}}
\put(20,10){\circle*{2}}
\put(25,10){\circle*{3}}
\put(30,10){\circle*{4}}
\put(35,10){\circle*{5}}
\end{picture}
```

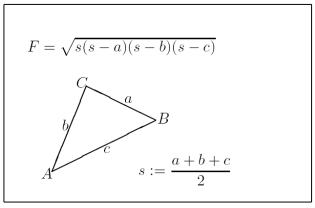


There is another possibility within the picture environment. If one is not afraid of doing the necessary calculations (or leaving them to a program), arbitrary circles and ellipses can be patched together from quadratic Bézier curves. See *Graphics in LaTeX2e* for examples and Java source files.

Text and formulas

As this example shows, text and formulas can be written in the environment with the `\put` command in the usual way:

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,5)
\thicklines
\put(1,0.5){\line(2,1){3}}
\put(4,2){\line(-2,1){2}}
\put(2,3){\line(-2,-5){1}}
\put(0.7,0.3){$A$}
\put(4.05,1.9){$B$}
\put(1.7,2.95){$C$}
\put(3.1,2.5){$a$}
\put(1.3,1.7){$b$}
\put(2.5,1.05){$c$}
\put(0.3,4){$F=
\sqrt{s(s-a)(s-b)(s-c)}$}
\put(3.5,0.4){$\displaystyle
s:=\frac{a+b+c}{2}$}
\end{picture}
```



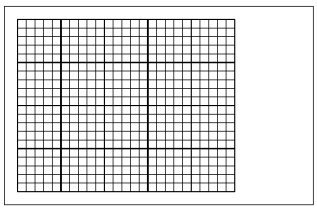
`\multiput` and `\linethickness`

The command

```
\multiput(x, y)(dx, dy ){n}{object}
```

has 4 arguments: the starting point, the translation vector from one object to the next, the number of objects, and the object to be drawn. The `\linethickness` command applies to horizontal and vertical line segments, but neither to oblique line segments, nor to circles. It does, however, apply to quadratic Bézier curves!

```
\setlength{\unitlength}{2mm}
\begin{picture}(30,20)
\linethickness{0.075mm}
\multiput(0,0)(1,0){26}%
{\line(0,1){20}}
\multiput(0,0)(0,1){21}%
{\line(1,0){25}}
\linethickness{0.15mm}
\multiput(0,0)(5,0){6}%
{\line(0,1){20}}
\multiput(0,0)(0,5){5}%
{\line(1,0){25}}
\linethickness{0.3mm}
\multiput(5,0)(10,0){2}%
{\line(0,1){20}}
\multiput(0,5)(0,10){2}%
{\line(1,0){25}}
\end{picture}
```



Ovals

The command

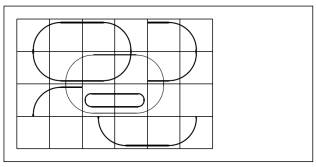
```
\put(x, y){\oval(w, h)}
```

or

```
\put(x, y){\oval(w, h)[position]}
```

produces an oval centered at *(x, y)* and having width *w* and height *h*. The optional position arguments *b, t, l, r* refer to "top", "bottom", "left", "right", and can be combined, as the example illustrates. Line thickness can be controlled by two kinds of commands: `\linethickness{length}` on the one hand, `\thinlines` and `\thicklines` on the other. While `\linethickness{length}` applies only to horizontal and vertical lines (and quadratic Bézier curves), `\thinlines` and `\thicklines` apply to oblique line segments as well as to circles and ovals.

```
\setlength{\unitlength}{0.75cm}
\begin{picture}(6,4)
\linethickness{0.075mm}
\multiput(0,0)(1,0){7}%
{\line(0,1){4}}
\multiput(0,0)(0,1){5}%
{\line(1,0){6}}
\thicklines
\put(2,3){\oval(3,1.8)}
\thinlines
\put(3,2){\oval(3,1.8)}
\thicklines
\put(2,1){\oval(3,1.8)[tl]}
\put(4,1){\oval(3,1.8)[b]}
\put(4,3){\oval(3,1.8)[r]}
\put(3,1.5){\oval(1.8,0.4)}
\end{picture}
```



Multiple Use of Predefined Picture Boxes

A picture box can be *declared* by the command

```
\newsavebox{name}
```

then *defined* by

```
\savebox{name}(width,height)[position]{content}
```

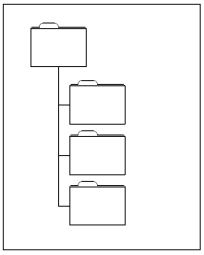
and finally arbitrarily often be *drawn* by

```
\put(x, y){\usebox{name}}
```

The optional position parameter has the effect of defining the "anchor point" of the savebox. In the example it is set to "bl" which puts the anchor point into the bottom left corner of the savebox. The other position specifiers are top and right.

The *name* argument refers to a LaTeX storage bin and therefore is of a command nature (which accounts for the backslashes in the current example). Boxed pictures can be nested: In this example, `\foldera` is used within the definition of `\folderb`. The `\oval` command had to be used as the `\line` command does not work if the segment length is less than about 3 mm.

```
\setlength{\unitlength}{0.5mm}
\begin{picture}(120,168)
\newsavebox{\foldera}
\savebox{\foldera}
  (40,32)[bl]{% definition
  \multiput(0,0)(0,28){2}
    {\line(1,0){40}}
  \multiput(0,0)(40,0){2}
    {\line(0,1){28}}
  \put(1,28){\oval(2,2)[tl]}
  \put(1,29){\line(1,0){5}}
  \put(9,29){\oval(6,6)[tl]}
  \put(9,32){\line(1,0){8}}
  \put(17,29){\oval(6,6)[tr]}
  \put(20,29){\line(1,0){19}}
  \put(39,28){\oval(2,2)[tr]}
}

\newsavebox{\folderb}
\savebox{\folderb}
  (40,32)[l]{% definition
  \put(0,14){\line(1,0){8}}
  \put(8,0){\usebox{\foldera}}
}

\put(34,26){\line(0,1){102}}
\put(14,128){\usebox{\foldera}}
\multiput(34,86)(0,-37){3}
{\usebox{\folderb}}
\end{picture}
```



Quadratic Bézier Curves

The command

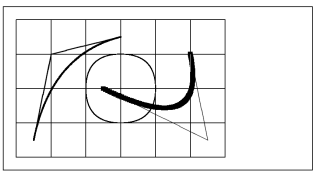
```
\qbezier(x1, y1)(x, y)(x2, y2)
```

draws a quadratic bezier curve where $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$denote the end points, and $S = (x, y)$ denotes the intermediate control point. The respective tangent slopes, $m_1$and m_2, can be obtained from the equations

$$\begin{cases} x = \frac{m_2 x_2 - m_1 x_1 - (y_2 - y_1)}{m_2 - m_1} \\ y = y_i + m_i(x - x_i); \quad (i = 1, 2 \text{ gives same solution}) \end{cases}$$

See *Graphics in LaTeX2e* for a Java program which generates the necessary `\qbezier` command line.

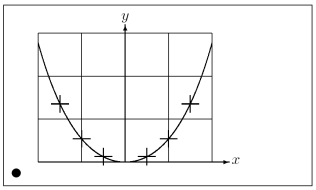
```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)
\linethickness{0.075mm}
\multiput(0,0)(1,0){7}
{\line(0,1){4}}
\multiput(0,0)(0,1){5}
{\line(1,0){6}}
\thicklines
\put(0.5,0.5){\line(1,5){0.5}}
\put(1,3){\line(4,1){2}}
\qbezier(0.5,0.5)(1,3)(3,3.5)
\thinlines
\put(2.5,2){\line(2,-1){3}}
\put(5.5,0.5){\line(-1,5){0.5}}
\linethickness{1mm}
\qbezier(2.5,2)(5.5,0.5)(5,3)
\thinlines
\qbezier(4,2)(4,3)(3,3)
\qbezier(3,3)(2,3)(2,2)
\qbezier(2,2)(2,1)(3,1)
\qbezier(3,1)(4,1)(4,2)
\end{picture}
```



As this example illustrates, splitting up a circle into 4 quadratic Bézier curves is not satisfactory. At least 8 are needed. The figure again shows the effect of the `\linethickness` command on horizontal or vertical lines, and of the `\thinlines` and the `\thicklines` commands on oblique line segments. It also shows that both kinds of commands affect quadratic Bézier curves, each command overriding all previous ones.

Catenary

```
\setlength{\unitlength}{1cm}
\begin{picture}(4.3,3.6)(-2.5,-0.25)
\put(-2,0){\vector(1,0){4.4}}
\put(2.45,-.05){$x$}
\put(0,0){\vector(0,1){3.2}}
\put(0,3.35){\makebox(0,0){$y$}}
\qbezier(0.0,0.0)(1.2384,0.0)
(2.0,2.7622)
\qbezier(0.0,0.0)(-1.2384,0.0)
(-2.0,2.7622)
\linethickness{.075mm}
\multiput(-2,0)(1,0){5}
{\line(0,1){3}}
\multiput(-2,0)(0,1){4}
{\line(1,0){4}}
\linethickness{.2mm}
\put( .3,.12763){\line(1,0){.4}}
\put(.5,-.07237){\line(0,1){.4}}
\put(-.7,.12763){\line(1,0){.4}}
\put(-.5,-.07237){\line(0,1){.4}}
\put(.8,.54308){\line(1,0){.4}}
\put(1,.34308){\line(0,1){.4}}
\put(-1.2,.54308){\line(1,0){.4}}
\put(-1,.34308){\line(0,1){.4}}
\put(1.3,1.35241){\line(1,0){.4}}
\put(1.5,1.15241){\line(0,1){.4}}
\put(-1.7,1.35241){\line(1,0){.4}}
\put(-1.5,1.15241){\line(0,1){.4}}
\put(-2.5,-0.25){\circle*{0.2}}
\end{picture}
```



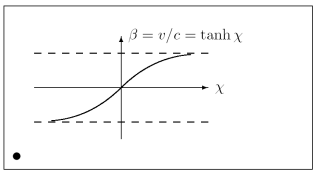
In this figure, each symmetric half of the catenary $y = \cosh x - 1$ is approximated by a quadratic Bézier curve. The right half of the curve ends in the point (2, 2.7622), the slope there having the value m = 3.6269. Using again equation (*), we can calculate the intermediate control points. They turn out to be (1.2384, 0) and (−1.2384, 0). The crosses indicate points of the real catenary. The error is barely noticeable, being less than one percent. This example points out the use of the optional argument of the `\begin{picture}` command. The picture is defined in convenient "mathematical" coordinates, whereas by the command

```
\begin{picture}(4.3,3.6)(-2.5,-0.25)
```

its lower left corner (marked by the black disk) is assigned the coordinates (–2.5,–0.25).

Plotting graphs

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)(-3,-2)
\put(-2.5,0){\vector(1,0){5}}
\put(2.7,-0.1){$\chi$}
\put(0,-1.5){\vector(0,1){3}}
\multiput(-2.5,1)(0.4,0){13}
{\line(1,0){0.2}}
\multiput(-2.5,-1)(0.4,0){13}
{\line(1,0){0.2}}
\put(0.2,1.4)
{$\beta=v/c=\tanh\chi$}
\qbezier(0,0)(0.8853,0.8853)
(2,0.9640)
\qbezier(0,0)(-0.8853,-0.8853)
(-2,-0.9640)
\put(-3,-2){\circle*{0.2}}
\end{picture}
```



The control points of the two Bézier curves were calculated with formulas (*). The positive branch is determined by $P_1 = (0,0)$, $m_1 = 1$and $P_2 = (2, \tanh 2)$, $m_2 = 1/\cosh^2 2$. Again, the picture is defined in mathematically convenient coordinates, and the lower left corner is assigned the mathematical coordinates (–3,–2) (black disk).

The `picture` environment and gnuplot

The powerful scientific plotting package gnuplot has the capability to output directly to a LaTeX `picture` environment. It is often far more convenient to plot directly to LaTeX, since this saves having to deal with potentially troublesome postscript files. Plotting scientific data (or, indeed, mathematical figures) this way gives much greater control, and of course typesetting ability, than is available from other means (such as postscript). Such pictures can then be added to a document by an `\include{}` command.

N.B. gnuplot is a powerful piece of software with a vast array of commands. A full discussion of gnuplot lies beyond the scope of this note.

Xy-pic

`xy` is a special package for drawing diagrams. To use it, simply add the following line to the preamble of your document:

```
\usepackage[all]{xy}
```

where "all" means you want to load a large standard set of functions from *Xy-pic*, suitable for developing the kind of diagrams discussed here.

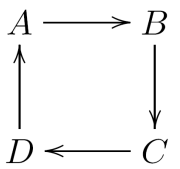
The primary way to draw *Xy-pic* diagrams is over a matrix-oriented canvas, where each diagram element is placed in a matrix slot:

```
\begin{displaymath}
    \xymatrix{A & B \\
              C & D }
\end{displaymath}
```

$$\begin{matrix} A & B \\ C & D \end{matrix}$$

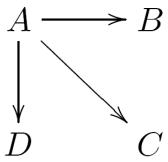
The `\xymatrix` command must be used in math mode. Here, we specified two lines and two columns. To make this matrix a diagram we just add directed arrows using the `\ar` command.

```
\begin{displaymath}
    \xymatrix{ A \ar[r] & B \ar[d] \\
               D \ar[u] & C \ar[l] }
\end{displaymath}
```



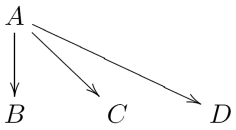
The arrow command is placed on the origin cell for the arrow. The arguments are the direction the arrow should point to (up, down, right and left).

```
\begin{displaymath}
    \xymatrix{
        A \ar[d] \ar[dr] \ar[r] & B \\
        D                       & C }
\end{displaymath}
```



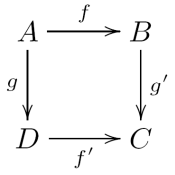
To make diagonals, just use more than one direction. In fact, you can repeat directions to make bigger arrows.

```
\begin{displaymath}
    \xymatrix{
        A \ar[d] \ar[dr] \ar[drr] &   &   \\
        B                         & C & D }
\end{displaymath}
```



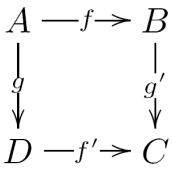
We can draw even more interesting diagrams by adding labels to the arrows. To do this, we use the common superscript and subscript operators.

```
\begin{displaymath}
    \xymatrix{
        A \ar[r]^f \ar[d]_g & B \ar[d]^{g'} \\
        D \ar[r]_{f'}       & C }
\end{displaymath}
```



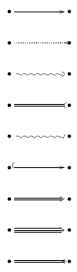
As shown, you use these operators as in math mode. The only difference is that that superscript means "on top of the arrow", and subscript means "under the arrow". There is a third operator, the vertical bar: | It causes text to be placed in the arrow.

```
\begin{displaymath}
    \xymatrix{
        A \ar[r]|f \ar[d]|g & B \ar[d]|{g'} \\
        D \ar[r]|{f'}       & C }
\end{displaymath}
```



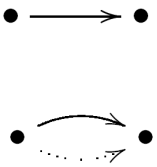
To draw an arrow with a hole in it, use `\ar[...]|\hole`. In some situations, it is important to distinguish between different types of arrows. This can be done by putting labels on them, or changing their appearance

```
\shorthandoff{"}
\begin{displaymath}
    \xymatrix{
        \bullet\ar@{->}[rr]      && \bullet\\
        \bullet\ar@{.<}[rr]      && \bullet\\
        \bullet\ar@{~)}[rr]      && \bullet\\
        \bullet\ar@{=(}[rr]      && \bullet\\
        \bullet\ar@{~/}[rr]      && \bullet\\
        \bullet\ar@{^{(}->}[rr] && \bullet\\
        \bullet\ar@2{->}[rr]     && \bullet\\
        \bullet\ar@3{->}[rr]     && \bullet\\
        \bullet\ar@{=+}[rr]      && \bullet }
\end{displaymath}
\shorthandon{"}
```



Notice the difference between the following two diagrams:

```
\begin{displaymath}
    \xymatrix{ \bullet \ar[r] \ar@{.>}[r] & \bullet }
\end{displaymath}

\begin{displaymath}
    \xymatrix{
        \bullet \ar@/^/[r]
        \ar@/_/@{.>}[r] &
        \bullet }
\end{displaymath}
```



The modifiers between the slashes define how the curves are drawn. *Xy-pic* offers many ways to influence the drawing of curves; for more information, check the *Xy-pic* documentation.

If you are interested in a more thorough introduction then consult the Xy-pic Home Page [1], which contains links to several other tutorials as well as the reference documentation.

PGF/TikZ

One possible solution how to draw graphics directly with TeX commands is PGF/TikZ [2]. Ti*k*Z can produce portable graphics in both PDF and PostScript formats using either plain (pdf)TEX, (pdf)Latex or ConTEXt. It comes with very good documentation and an extensive collection of examples: http://www.texample.net/tikz/

PGF ("portable graphics format") is the basic layer, providing a set of basic commands for producing graphics, and Ti*k*Z ("Ti*k*Z ist kein Zeichenprogramm") is the frontend layer with a special syntax, making the use of PGF easier. Ti*k*Z commands are prevalently similar to Metafont, the option mechanism is similar to PsTricks syntax.

Other packages building on top of Ti*k*Z (e.g., for drawing electrical circuits) can be found here: http://ftp.dante.de/tex-archive/help/Catalogue/bytopic.html#pgftikzsection

In the following some basics of Ti*k*Z are presented.

Loading Package, Libraries - tikzpicture environment

Using Ti*k*Z in a LaTeX document requires to include

```
\usepackage{tikz}
```

somewhere in the preamble. This will automatically load the pgf package. To load further libraries use

```
\usetikzlibrary{list of libraries separated by commas}
```

Examples for libraries are "arrows", "automata", "backgrounds", "calendar", "chains", "matrix", "mindmap", "patterns", "petri", "shadows", "shapes.geometric", "shapes.misc", "spy", "trees".

Drawing commands have to be enclosed in an tikzpicture environment

```
\begin{tikzpicture}[options ]
  tikz commands
\end{tikzpicture}
```

or alternatively

```
\tikz[options]{tikz commands}
```

One possible option useful for inlined graphics is

```
baseline=dimension
```

Without that option the lower end of the picture is put on the baseline of the surrounding text. Using this option, you can specify that the picture should be raised or lowered such that the height dimension is on the baseline.

Another option to scale the entire picture is

```
scale=factor
```

Specifying Coordinates

Coordinates are specified in round brackets in an arbitrary TEX dimension either using cartesian coordinates (comma separated), e.g. 1cm in x and 2pt in y direction

```
(1cm,2pt)
```

or using polar coordinates (colon separated), e.g. 1cm in 30 degree direction

```
(30:1cm)
```

Without specifying a unit `(1,2)`, the standard one is cm `(1cm,2cm)`.

Relative coordinates to the previous given point are given by adding one or two plus signs in front of the coordinate. With "++" the last point of the path becomes the current position, with "+" the previous point stays the current path position. Example: 2 standard units to the right of the last point used:

```
++(2,0)
```

Syntax for Paths

A path is a series of straight and curved line segments. It has to end with a semicolon.

```
\path[<options>] specification;
```

Options for path actions are e.g: "draw", "fill", "pattern", "shade" (filling, in which its color changes smoothly from one to another), "clip" (all subsequent drawings up to the end of the current scope are clipped against the current path and the size of subsequent paths will not be important for the picture size), "use as bounding box".

The "\path" command with these options can be combined to: "\draw", "\fill", "\filldraw", "\pattern", "\shade", "\shadedraw", "\clip", "\useasboundingbox" .

Geometric path options: "rotation=<angle in degree>", "xshift=<length>", "yshift=<length>", "scaling=<factor>", "xscale=<factor>", "yscale=<factor>".

Color options for drawing paths: "color=<color name>", "draw=<line color>", "opacity=<factor>".

Line width options: "line width=<dimension>", and abbreviations "ultra thin" for 0.1pt, "very thin" for 0.2pt, "thin" for 0.4pt (the default width), "semithick" for 0.6pt, "thick" for 0.8pt, "very thick" for 1.2pt, "ultra thick" for 1.6pt.

Line end, line join options: "line cap=<type: round, rect, or butt>", "arrows=<start arrow kind>-<end arrow kind>", "rounded corners", "rounded corners=<size>", "line join=<type: round, bevel, or miter>".

Line pattern options: "dash pattern=<dash pattern>" (e.g. "dash pattern=on 2pt off 3pt on 4pt off 4pt"), "dash phase=dash phase", "solid", "dashed", "dotted", "dashdotted", "densely dotted", "loosely dotted", "double".

Options for filling paths are e.g. "fill=<fill color>", "pattern=<name>", "pattern color=<color>"

Straight lines are given by coordinates separated by a double minus,

```
\draw (1,0) -- (0,0) -- (0,1);
```

The first coordinate represents a move-to operation. This is followed by a series of “path extension operations”, like "-- (coordinates)".

A further move-to operation in an existing path starts a new part of the path, which is not connected to the previous part of the path. Here: Move to (0,0) straight line to (2,0), move to (0,1) straight line to (2,1):

```
\draw (0,0) -- (2,0) (0,1) -- (2,1);
```

Connecting two points via straight lines that are only horizontal and vertical, use for first horizontal then vertial

```
\draw (0,0) -| (1,1);
```

or for first vertical then horizontal

```
\draw (0,0) |- (1,1);
```

Curved paths using a Bezier curve can be created using the "..controls() ..()" command, with one or two control points.

```
\draw (0,0) .. controls (1,1) .. (4,0)
      (5,0) .. controls (6,0) and (6,1) .. (5,2);
```

A connected path can be closed using the "--cycle" operation:

```
\draw (1,0) -- (0,0) -- (0,1) -- cycle;
```

User-defined paths can be created using the "to" operation. Without an option it corresponds to a straight line, exactly like the double minus command. Using the "out" and "in" option a curved path can created. E.g. "[out=135,in=45]" causes the path to leave at an angle of 135 degree at the first coordinate and arrive at an angle of 45 degree at the second coordinate.

```
\draw (0,0) to (3,2);
\draw (0,0) to[out=90,in=180] (3,2);
\draw (0,0) to[bend right] (3,2);
```

For rectangles a special syntax exist. Use a move-to operation to one corner and after "rectangle" the coordinates of the diagonal corner. The last one becomes the new current point.

```
\draw (0,0) rectangle (1,1);
\shade[top color=yellow, bottom color=black] (0,0) rectangle (2,-1);
```

Circles and ellipses paths are defined beginning with their center then using the "circle command" either with one length as radius of a circle or with two lengths as semi-axes of an ellipse.

```
\draw (0,0) circle [radius=1.5];
\draw (0,0) circle (2cm); % old syntax
\draw (0,0) circle [x radius=1.5cm, y radius=10mm];
\draw (0,0) circle (1.2cm and 8mm); % old syntax
\draw (0,0) circle [x radius=1cm, y radius=5mm, rotate=30];
\draw[rotate=30] (0,0) ellipse (20pt and 10pt);  % old syntax
```

There are many more predefined commands for special paths, like "arc" for a part of an ellipse, "grid", "parabola", "sin", "cos" (sine or cosine curve in the interval [0,π/2]).

```
\draw (0,0) arc (0:270:8mm);
\draw (0,0) arc (0:315:1.75cm and 1cm);
\filldraw[fill=green!20!white, draw=green!40!black] (0,0) -- (12mm,0mm)
 arc (0:30:12mm) -- (0,0);
```

The fill color "green!20!white" means 20% green and 80% white mixed together.

```
\draw[step=0.5, gray, very thin] (-1.4,-1.4) grid (1.4,1.4);
\draw (0,0) parabola (1,1.5) parabola[bend at end] (2,0);
\draw (0,0) sin (1,1) cos (2,0) sin (3,-1) cos (4,0) sin (5,1);
```

To add arrow tips there are simple options for the drawing command:

```
\draw [->] (0,0) -- (30:20pt);
\draw [<->] (1,0) arc (180:30:10pt);
\draw [<<->] (2,0) -- ++(0.5,10pt) -- ++(0.5,-10pt) -- ++(0.5,10pt);
```

A loop can be realized by "\foreach ⟨variable⟩ in {⟨list of values⟩} ⟨commands⟩".

```
\foreach \x in {0,...,9}
  \draw (\x,0) circle (0.4);
```

Nodes

A node is typically a rectangle or circle or another simple shape with some text on it. In the simplest case, a node is just some text that is placed at some coordinate. Nodes are not part of the path itself, they are added to the picture after the path has been drawn.

Inside a path operation use the following syntax after a given coordinate:

```
node[<options>](<name>){<text>}
```

The "`(<name>)`" is a name for later reference and it is optional. If you only want to name a certain position without writing text there are two possibilities:

```
node[<options>](<name>){}
coordinate[<options>](<name>)
```

Writing text along a given path using the note command is shown as simple example:

```
\draw[dotted]
    (0,0) node {1st node}
 -- (1,1) node {2nd node}
 -- (0,2) node {3rd node}
 -- cycle;
```

Possible options for the node command are e.g. "`inner sep=<dimension>`", "`outer sep=<dimension>`", "`minimum size=<dimension>`", "`shape aspect=<aspect ratio>`", "`text=<color>`", "`font=`", "`align=<left_right_center>`".

A node is centered at the current coordinate by default. Often it would be better to have the node to the besides the actual coordinate: Right ("`right`" or "`anchor=west`"), left ("`left`" or "`anchor=east`"), above ("`above`" or "`anchor=south`"), below ("`below`" or "`anchor=north`"). Combinations are also possible, like "`anchor=north east`" or "`below left`".

```
\fill[fill=yellow]
    (0,0) node {1st node}
 -- (1,1) node[circle,inner sep=0pt,draw] {2nd node}
 -- (0,2) node[fill=red!20,draw,double,rounded corners] {3rd node};
```

To place nodes on a line or a curve use the "`pos=<fraction>`" option, where fraction is a floating point number between 0 representing the previous coordinate and 1 representing the current coordinate.

```
\draw (0,0) -- (3,1)
  node[pos=0]{0} node[pos=0.5]{1/2} node[pos=0.9]{9/10};
```

There exist some abbreviations: "`at start`" for "`pos=0`", "`very near start`" for "`pos=0.125`", "`near start`" for "`pos=0.25`", "`midway`" for "`pos=0.5`", "`near end`" for "`pos=0.75`", "`very near end`" for "`pos=0.875`", "`at end`" for "`pos=1`".

The "`sloped`" option causes the node to be rotated to become a tangent to the curve.

Since nodes are often the only path operation on paths, there are special commands for creating paths containing only a node, the first with text ouput, the secound without:

```
\node[<options>](<name>) at (<coordinate>){<text>};
\coordinate[<options>](<name>) at (<coordinate>);
```

One can connect nodes using the nodes' labels as coordinates. Having "`\path(0,0) node(x) {} (3,1) node(y) {};`" defined, the node at (0,0) got the name "`(x)`" and the one at (3,1) got a label "`(y)`".

```
\path (0,0) node(x) {}
      (3,1) node(y) {};
\draw (x) -- (y);
```

Equivalent to

```
\coordinate (x) at (0,0);
\coordinate (y) at (3,1);
\draw (x) -- (y);
```

Path construction operations try to be clever, such that the path starts at the border of the node's shape and not from the node's center.

```
\path (0,0) node(x) {Hello World!}
(3,1) node[circle,draw](y) {$\int_1^2 x \mathrm d x$};
\draw[->,blue] (x) -- (y);
\draw[->,red] (x) -| node[near start,below] {label} (y);
\draw[->,orange] (x) .. controls +(up:1cm) and +(left:1cm) ..
node[above,sloped] {label} (y);
```

Once the node x has been defined, you can use anchors as defined above relative to (x) as "`(x.<anchor>)`", like "`(x.north)`".

Examples

Example 1

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
  \begin{tikzpicture}
  \draw[thick,rounded corners=8pt] (0,0) -- (0,2) -- (1,3.25)
   -- (2,2) -- (2,0) -- (0,2) -- (2,2) -- (0,0) -- (2,0);
  \end{tikzpicture}
\end{document}
```

Example 2

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
 \begin{tikzpicture}[scale=3]
 \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid (1.2,1.2);
 \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm,0mm) arc
(0:30:3mm) -- cycle;
 \draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
 \draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
 \draw (0,0) circle (1cm);
 \draw[very thick,red] (30:1cm) -- node[left,fill=white] {$\sin
\alpha$} (30:1cm |- x axis);
```

```
 \draw[very thick,blue] (30:1cm |- x axis) --
node[below=2pt,fill=white] {$\cos \alpha$} (0,0);
 \draw (0,0) -- (30:1cm);
 \foreach \x/\xtext in {-1, -0.5/-\frac{1}{2}, 1}
   \draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north,fill=white]
{$\xtext$};
 \foreach \y/\ytext in {-1, -0.5/-\frac{1}{2}, 0.5/\frac{1}{2}, 1}
   \draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=east,fill=white]
{$\ytext$};
 \end{tikzpicture}
\end{document}
```

Chemical Graphics

unknown operator: u'strong' unknown operator: u'strong'

chemfig [3] is a package used to draw 2D chemical structures. It is an alternative to ochem [4]. Whereas ochem requires Perl to draw chemical structures, chemfig uses the tikz [5] package to produce its graphics. chemfig is used by adding the following to the preamble:

```
\usepackage{chemfig}
```

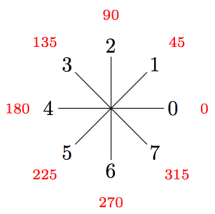
Basic Usage

The primary command used in this package is `\chemfig{}`:

```
\chemfig{<atom1><bond type>[<angle>,<coeff>,<tikz code>]<atom2>}
```

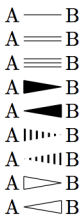
<angle> is the bond angle between two atoms (or nodes). There are three types of angles: absolute, relative, and predefined. Exact angles give a precise angle (generally, 0 to 360, though they can also be negative), and are represented with the syntax `[:<absolute angle>]`. Relative angles require the syntax `[::<relative angle>]` and produce an angle relative to the angle of the preceding bond. Finally, predefined angles are whole numbers from 0 to 7 indicating intervals of 45 degrees. These are produced with the syntax `[< predefined angle>]`. The predefined angles and their corresponding absolute angles are represented in the diagram below.

```
\chemfig{(-[:0,1.5,,,draw=none]\scriptstyle\color{red}0)
(-[1]1)(-[:45,1.5,,,draw=none]\scriptstyle\color{red}45)
(-[2]2)(-[:90,1.5,,,draw=none]\scriptstyle\color{red}90)
(-[3]3)(-[:135,1.5,,,draw=none]\scriptstyle\color{red}135)
(-[4]4)(-[:180,1.5,,,draw=none]\scriptstyle\color{red}180)
(-[5]5)(-[:225,1.5,,,draw=none]\scriptstyle\color{red}225)
(-[6]6)(-[:270,1.5,,,draw=none]\scriptstyle\color{red}270)
(-[7]7)(-[:315,1.5,,,draw=none]\scriptstyle\color{red}315)
-0}
```



<bond type> describes the bond attaching <atom1> and <atom2>. There are 9 different bond types:

```
\chemfig{A-B}\\
\chemfig{A=B}\\
\chemfig{A~B}\\
\chemfig{A>B}\\
\chemfig{A<B}\\
\chemfig{A>:B}\\
\chemfig{A<:B}\\
\chemfig{A>|B}\\
\chemfig{A<|B}\\
```

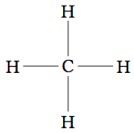


<coeff> represents the factor by which the bond's length will be multiplied.

<tikz code> includes additional options regarding the color or style of the bond.

A methane molecule, for instance, can be produced with the following code:

```
\chemfig{C(-[:0]H)(-[:90]H)(-[:180]H)(-[:270]H)}
```



Linear molecules (such as methane) are a weak example of this, but molecules are formed in chemfig by nesting.

Skeletal Diagrams

Skeleton diagrams can be produced as follows:

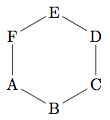
```
\chemfig{-[:30]-[:-30]-[:30]}
```

```
\chemfig{-[:30]=[:-30]-[:30]}
```

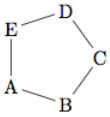
Rings

Rings follow the syntax `<atom>*<n>(code)`, where "n" indicates the number of sides in the ring and "code" represents the specific content of each ring (bonds and atoms).

```
\chemfig{A*6(-B-C-D-E-F-)}
```




```
\chemfig{A*5(-B-C-D-E-)}
```




```
\chemfig{*6(=-=-=-)}
```

```
\chemfig{**5(------)}
```

Lewis Structures

Lewis structures can be created by using the command

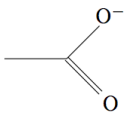
```
\lewis{<electron angle><electron>,<atom>}
```

within `\chemfig{}`.

Ions

For example, consider an acetate ion:

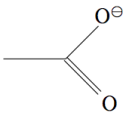
```
\chemfig{-(-[1]O^{-})=[7]O}
```



Because the chemfig commands enters the math mode, ion charges can be added as superscripts (one caveat: a negative ion requires that the minus sign be enclosed in brackets, as in the example).

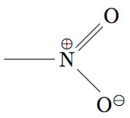
The charge of an ion can be circled by using `\oplus` and `\ominus`:

```
\chemfig{-(-[1]O^{\ominus})=[7]O}
```



Alternatively, charges can be placed above ions using \chemabove{}{}:

```
\chemfig{-\chemabove{N}{\scriptstyle\oplus}(=[1]O)-[7]O^{\ominus}}
```



Resonance Structures and Formal Charges

Resonance structures require a few math commands:

```
% see "Advanced Mathematics" for use of \left and \right
% add to preamble:
%      \usepackage{mathtools}      % \Longleftrightarrow
$\left\{\chemfig{O-N(=[:60]O)-[:300]O}\right\}
\Longleftrightarrow
\left\{\chemfig{O=N(-[:60]O)-[:300]O}\right\}
\Longleftrightarrow
\left\{\chemfig{O-N(-[:60]O)=[:300]O}\right\}$
```

Chemical Reactions

Chemical reactions can be created with the following commands:

```
\chemrel[<arg1>][<arg2>]{<arrow code>}  \chemsign+      % produces a +
```

In \chemrel{}, <arg1> and <arg2> represent text placed above and below the arrow, respectively.

There are four types of arrows that can be produced with tt>\chemrel{}:

```
A\chemrel{->}B\par
A\chemrel{<-}B\par
A\chemrel{<->}B\par
A\chemrel{<>}B
```

Naming Chemical Graphics

Molecules can be named with the command

```
\chemname[<dim>]{\chemfig{<code of the molecule>}}{<name>}
```

<dim> is inserted between the bottom of the molecule and the top of the name defined by <name>. It is 1.5ex by default.

<name> will be centered relative to the molecule it describes.

```
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
\chemsign{+}
\chemname{\chemfig{R'OH}}{Alcohol}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_2O}}{Water}
```

In the reaction above, \chemname{} inserts 1.5ex plus the depth of the carboxylic acid molecule in between each molecule and their respective names. This is because the graphic for the first molecule in the reaction (carboxylic acid) extends deeper than the rest of the molecules. A different result is produced by putting the alcohol first:

```
\chemname{\chemfig{R'OH}}{Alcohol}
\chemsign{+}
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_2O}}{Water}
```

This is fixed by adding \chemnameinit{<deepest molecule>} before the first instance of \chemname{} in a reaction and by adding \chemnameinit{} after the reaction:

```
\chemnameinit{\chemfig{R-C(-[:-30]OH)=[:30]O}}
\chemname{\chemfig{R'OH}}{Alcohol}
\chemsign{+}
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
```

```
\chemsign{+}
\chemname{\chemfig{H_2O}}{Water}
\chemnameinit{}
```

Lastly, adding \\ in <name> will produce a line-break, allowing the name to span multiple lines.

Advanced Graphics

For advanced commands and examples, refer to the chemfig manual [6], where a more thorough and complete introduction to the package can be found.

Alternatives

In many cases, especially for more advanced diagrams, it may be easier to draw the graphics using external vector graphics software, and then import the file into the document (see ../Importing Graphics). However most software does not support LaTeX fonts or mathematical notation, which can result in ugly and inconsistent graphics. There are several solutions to this problem.

The easiest solution is to use the picture environment and then simply use the "put" command to put a graphics file inside the picture, along with any other desired LaTeX element. For example:

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,5)
\put(3.5,0.4){$\displaystyle
s:=\frac{a+b+c}{2}$}
\put(1,1){\includegraphics[width=2cm,height=2cm]{picture.eps}}
\end{picture}
```

Another solution is to use textext [7], a plug-in for Inkscape [22] which allows one to insert small LaTeX objects into .SVG images. These images can then be saved as .EPS (or .PDF) files which may then be imported into the LaTeX document proper.

Yet another solution is provided by lpic [8], which allows TeX annotations to imported graphics.

Using xfig to create pictures

An option that allows significantly more flexibility while creating graphics that are consistent with LaTeX is to use xfig [9]. xfig is a drawing program, which allows exports into various formats from which it's possible to import into LaTeX. While the software is designed for Linux, it can be run using Macports [10] for Macs and under any X-Window System for Windows.

There are many ways to use xfig to create graphics for LaTeX documents. One method is to export the drawing as a LaTeX document. This method, however, suffers from various drawbacks: lines can be drawn only at angles that are multiples of 30 and 45 degrees, lines with arrows can only be drawn at angles that are multiples of 45 degrees, several curves are not supported, etc.

Exporting a file as PDF/LaTeX or PS/LaTeX, on the other hand, offers a good deal more flexibility in drawing. Here's how it's done:

1. Create the drawing in xfig. Wherever you need LaTeX text, such as a mathematical formula, enter a LaTeX string in a textbox.
2. Use the Edit tool to open the properties of each of those textboxes, and change the option on the "Special Flag" field to Special. This tells LaTeX to interpret these textboxes when it opens the figure.
3. Go to File -> Export and export the file as PDF/LaTeX (both parts) or PS/LaTeX (both parts), depending on whether you are using pdflatex or pslatex to compile your file.

4. In your LaTeX document, where the picture should be, use the following, where "test" is replaced by the name of the image:

```
\begin{figure}[htbp]
 \centering
 \input{test.pdf_t}
 \caption{Your figure}
 \label{figure:example}
\end{figure}
```

   Observe that this is just like including a picture, except that rather than using `\includegraphics`, we use `\input`. If the export was into PS/LaTeX, the file extension to include would be .pstex_t instead of .pdf_t.
5. Make sure to include packages `graphicx` and `color` in the file, with the `usepackage` command right below the `documentclass` command, like this:

```
\usepackage{graphicx}
\usepackage{color}
```

And you're done!

For more details on using xfig with LaTeX, this chapter [11] of the xfig User Manual [12] may prove helpful.

References

[1] http://xy-pic.sourceforge.net
[2] http://en.wikipedia.org/wiki/PGF/TikZ
[3] http://www.ctan.org/tex-archive/macros/latex/contrib/chemfig/
[4] http://www.2k-software.de/ingo/ochem.html
[5] http://az.ctan.org/pkg/pgf
[6] http://mirror.ctan.org/macros/latex/contrib/chemfig/chemfig_doc_en.pdf
[7] http://pav.iki.fi/software/textext/
[8] http://www.math.uni-leipzig.de/~matveyev/lpic/
[9] http://www.xfig.org
[10] http://www.macports.org
[11] http://www-epb.lbl.gov/xfig/latex_and_xfig.html
[12] http://www-epb.lbl.gov/xfig/contents.html

Floats, Figures and Captions

The previous chapter introduced importing graphics. However, just having a picture stuck in between paragraphs does not look professional. For starters, we want a way of adding captions, and to be able to cross-reference. What we need is a way of defining *figures*. It would also be good if LaTeX could apply principles similar to when it arranges text to look its best to arranging pictures as well. This is where *floats* come into play.

Floats

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes "table" and "figure" floats, but you can define new ones of your own (see Custom Floats below). Floats are there to deal with the problem of the object that won't fit on the present page, and to help when you really don't want the object here just now.

Floats are not part of the normal stream of text, but separate entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the designer specifies). They always have a caption describing them and they are always numbered so they can be referred to from elsewhere in the text. LaTeX automatically floats Tables and Figures, depending on how much space is left on the page at the point that they are processed. If there is not enough room on the current page, the float is moved to the top of the next page. This can be changed by moving the Table or Figure definition to an earlier or later point in the text, or by adjusting some of the parameters which control automatic floating.

Authors sometimes have many floats occurring in rapid succession, which raises the problem of how they are supposed to fit on the page and still leave room for text. In this case, LaTeX stacks them all up and prints them together if possible, or leaves them to the end of the chapter in protest. The skill is to space them out within your text so that they intrude neither on the thread of your argument or discussion, nor on the visual balance of the typeset pages.

Figures

To create a figure that floats, use the `figure` environment.

```
\begin{figure}[placement specifier]
... figure contents ...
\end{figure}
```

The previous section mentioned how floats are used to allow Latex to handle figures, while maintaining the best possible presentation. However, there may be times when you disagree, and a typical example is with its positioning of figures. The *placement specifier* parameter exists as a compromise, and its purpose is to give the author a greater degree of control over where certain floats are placed.

Specifier	Permission
`h`	Place the float *here*, i.e., *approximately* at the same point it occurs in the source text (however, not *exactly* at the spot)
`t`	Position at the *top* of the page.
`b`	Position at the *bottom* of the page.
`p`	Put on a special *page* for floats only.
`!`	Override internal parameters Latex uses for determining "good" float positions.
`H`	Places the float at precisely the location in the LaTeX code. Requires the `float` package,[1] e.g., `\usepackage{float}`. This is somewhat equivalent to `h!`.

What you do with these *placement permissions* is to list which of the options you wish to make available to LaTeX. These are simply possibilities, and Latex will decide when typesetting your document which of your supplied specifiers it thinks is best.

Use `\listoffigures` to add a list of the figures in the beginning of the document. To change the name used in the caption from **Figure** to **Example**, use `\renewcommand{\figurename}{Example}` in the figure contents.

Figures with borders

It's possible to get a thin border around all figures. You have to write the following once at the beginning of the document:

```
\usepackage{float}
\floatstyle{boxed}
\restylefloat{figure}
```

The border will not include the caption.

Tables

Although tables have already been covered, it was only the internal syntax that was discussed. The `tabular` environment that was used to construct the tables is not a float by default. Therefore, for tables you wish to float, wrap the `tabular` environment within a `table` environment, like this:

```
\begin{table}
  \begin{tabular}{...}
  ... table data ...
  \end{tabular}
\end{table}
```

You may feel that it is a bit long winded, but such distinctions are necessary, because you may not want all tables to be treated as a float.

Use `\listoftables` to add a list of the tables in the beginning of the document.

Captions

It is always good practice to add a caption to any figure or table. Fortunately, this is very simple in LaTeX. All you need to do is use the `\caption{text}` command within the float environment. Because of how LaTeX deals sensibly with logical structure, it will automatically keep track of the numbering of figures, so you do not need to include this within the caption text.

The location of the caption is traditionally underneath the float. However, it is up to you to therefore insert the caption command after the actual contents of the float (but still within the environment). If you place it before, then the caption will appear above the float. Try out the following example to demonstrate this effect:

```
\documentclass[a4paper,12pt]{article}

\usepackage[english]{babel}
\usepackage{graphicx}

\begin{document}

\begin{figure}[h!]
  \caption{A picture of a gull.}
  \centering
    \includegraphics[width=0.5\textwidth]{gull}
\end{figure}

\begin{figure}[h!]
  \centering
    \reflectbox{%
      \includegraphics[width=0.5\textwidth]{gull}}
  \caption{A picture of the same gull
           looking the other way!}
\end{figure}

\begin{table}[h!]
  \begin{center}
    \begin{tabular}{| l c r |}
    \hline
    1 & 2 & 3 \\
    4 & 5 & 6 \\
    7 & 8 & 9 \\
    \hline
    \end{tabular}
  \end{center}
  \caption{A simple table}
\end{table}

Notice how the tables and figures
have independent counters.

\end{document}
```

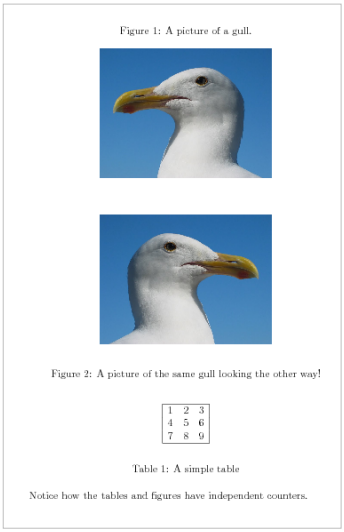


Figure 1: A picture of a gull.

Figure 2: A picture of the same gull looking the other way!

1	2	3
4	5	6
7	8	9

Table 1: A simple table

Notice how the tables and figures have independent counters.

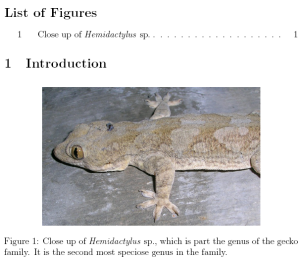
note that the command `\reflectbox{...}` flips its content horizontally.

Lists of figures and tables

Captions can be listed at the beginning of a paper or report in a "List of Tables" or a "List of Figures" section by using the `\listoftables` or `\listoffigures` commands, respectively. The caption used for each figure will appear in these lists, along with the figure numbers, and page numbers that they appear on.

The `\caption` command also has an optional parameter, `\caption[short]{long}` which is used for the *List of Tables* or *List of Figures*. Typically the `short` description is for the caption listing, and the `long` description will be placed beside the figure or table. This is particularly useful if the caption is long, and only a "one-liner" is desired in the figure/table listing. Here is an example of this usage:

```
\documentclass[12pt]{article}
\usepackage{graphics}

\begin{document}

\listoffigures

\section{Introduction}

\begin{figure}[hb]
  \centering
  \includegraphics[width=4in]{gecko}
  \caption[Close up of \textit{Hemidactylus} sp.]%
  {Close up of \textit{Hemidactylus} sp., which is
   part the genus of the gecko family. It is the
   second most speciose genus in the family.}
\end{figure}

\end{document}
```


List of Figures

1 Close up of *Hemidactylus* sp. 1

1 Introduction

Figure 1: Close up of *Hemidactylus* sp., which is part the genus of the gecko family. It is the second most speciose genus in the family.

Side captions

It is sometimes desirable to have a caption appear on the side of a float, rather than above or below. The `sidecap` package can be used to place a caption beside a figure or table. The following example demonstrates this for a figure by using a `SCfigure` environment in place of the `figure` environment.

```
\documentclass{article}

\usepackage[pdftex]{graphicx}
\usepackage{sidecap}

\begin{document}

\begin{SCfigure}
  \centering
  \includegraphics[width=5\textwidth]%
    {Giraff_picture}% picture filename
  \caption{ ... caption text ... }
\end{SCfigure}

\end{document}
```


Figure 1: The giraffe (*Giraffa camelopardalis*) is an African even-toed ungulate mammal, the tallest of all land-living animal species. Males can be 4.8 to 5.5 metres tall and weigh up to 1,360 kilograms. The record-sized bull was 5.87 m tall and weighed approximately 2,000 kg. Females are generally slightly shorter and weigh less than the males do.

Labels and Cross-referencing

Labels and cross-references work fairly similarly to the general case - see the Labels and Cross-referencing section for more information.

> *Warning:* If you want to label a figure so that you can reference it later, you have to add the label **after the caption** (inside seems to work in LaTeX 2e) but **inside the floating environment**. If it is declared outside, it will give the section number. If the label picks up the section or list number instead of the figure number, put the label inside the caption to ensure correct numbering.

Wrapping text around figures

Although not normally the case in academic writing, an author may prefer that some floats do not break the flow of text, but instead allow text to wrap around it. (Obviously, this effect only looks decent when the figure in question is significantly narrower than the text width.)

A word of warning: Wrapping figures in LaTex will require a lot of manual adjustment of your document. There are several packages available for the task, but none of them work perfectly. Before you make the choice of including figures with text wrapping in your document, make sure you have considered all the options. For example, you could use a layout with two columns for your documents and have no text-wrapping at all.

Anyway, we will look at the package `wrapfig`. (Note: wrapfig may not come with the default installation of LaTeX; you might need to install additional packages manually.)

To use `wrapfig`, you must first add this to the preamble:

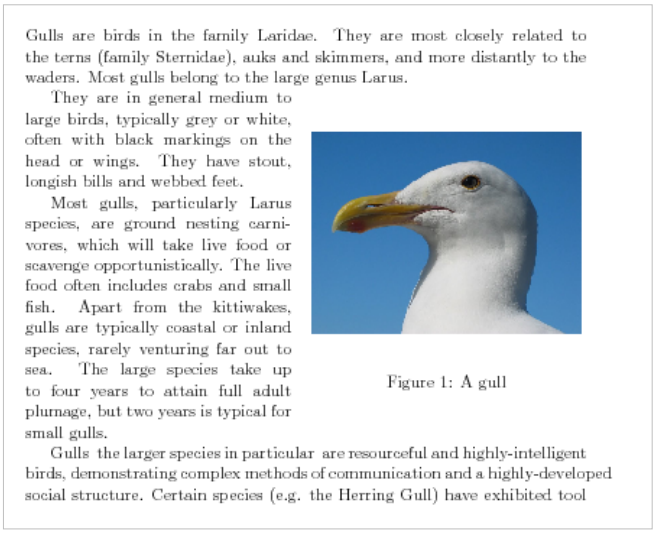
```
\usepackage{wrapfig}
```

This then gives you access to:

```
\begin{wrapfigure}[lineheight]{alignment}{width}
```

Alignment can normally be either *l* for left, or *r* for right. Lowercase *l* or *r* forces the figure to start precisely where specified (and may cause it to run over page breaks), while capital *L* or *R* allows the figure to float. If you defined your document as twosided, the alignment can also be *i* for inside or *o* for outside, as well as *I* or *O*. The *width* is, of course, the width of the figure. An example:

```
\begin{wrapfigure}{r}{0.5\textwidth}
  \begin{center}
    \includegraphics[width=0.48\textwidth]{gull}
  \end{center}
  \caption{A gull}
\end{wrapfigure}
```


Gulls are birds in the family Laridae. They are most closely related to the terns (family Sternidae), auks and skimmers, and more distantly to the waders. Most gulls belong to the large genus Larus.

They are in general medium to large birds, typically grey or white, often with black markings on the head or wings. They have stout, longish bills and webbed feet.

Most gulls, particularly Larus species, are ground nesting carnivores, which will take live food or scavenge opportunistically. The live food often includes crabs and small fish. Apart from the kittiwakes, gulls are typically coastal or inland species, rarely venturing far out to sea. The large species take up to four years to attain full adult plumage, but two years is typical for small gulls.

Figure 1: A gull

Gulls the larger species in particular are resourceful and highly-intelligent birds, demonstrating complex methods of communication and a highly-developed social structure. Certain species (e.g. the Herring Gull) have exhibited tool

Note that we have specified a size for both the `wrapfigure` environment and the image we have included. We did it in terms of the text width: it is always better to use relative sizes in LaTeX, let LaTeX do the work for you! The

"wrap" is slightly bigger than the picture, so the compiler will not return any strange warning and you will have a small white frame between the image and the surrounding text. You can change it to get a better result, but if you don't keep the image smaller than the "wrap", you will see the image *over* the text.

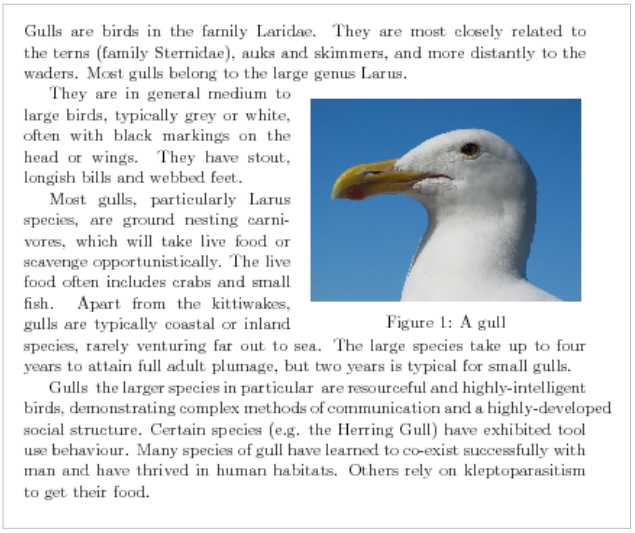
The wrapfig package can also be used with user-defined floats with float package. See below in the section on custom floats.

Tip for figures with too much white space

It happens that you'll generate figures with too much (or too little) white space on the top or bottom. In such a case, you can simply make use of the optional argument `[lineheight]`. It specifies the height of the figure in number of lines of text.

Another possibility is adding space within the float using the `\vspace{...}` command. The argument is the size of the space you want to add, you can use any unit you want, including pt, mm, in, etc. If you provide a negative argument, it will add a *negative* space, thus removing some white space. Using `\vspace` tends to move the caption relative to the float while the `[lineheight]` argument does not. Here is an example using the `\vspace` command, the code is exactly the one of the previous case, we just added some negative vertical spaces to shrink everything up:

```
\begin{wrapfigure}{r}{0.5\textwidth}
  \vspace{-20pt}
  \begin{center}
    \includegraphics[width=0.48\textwidth]{gull}
  \end{center}
  \vspace{-20pt}
  \caption{A gull}
  \vspace{-10pt}
\end{wrapfigure}
```


Gulls are birds in the family Laridae. They are most closely related to the terns (family Sternidae), auks and skimmers, and more distantly to the waders. Most gulls belong to the large genus Larus.

They are in general medium to large birds, typically grey or white, often with black markings on the head or wings. They have stout, longish bills and webbed feet.

Most gulls, particularly Larus species, are ground nesting carnivores, which will take live food or scavenge opportunistically. The live food often includes crabs and small fish. Apart from the kittiwakes, gulls are typically coastal or inland species, rarely venturing far out to sea. The large species take up to four years to attain full adult plumage, but two years is typical for small gulls.

Figure 1: A gull

Gulls the larger species in particular are resourceful and highly-intelligent birds, demonstrating complex methods of communication and a highly-developed social structure. Certain species (e.g. the Herring Gull) have exhibited tool use behaviour. Many species of gull have learned to co-exist successfully with man and have thrived in human habitats. Others rely on kleptoparasitism to get their food.

In this case it may look too shrunk, but you can manage spaces the way you like. In general, it is best not to add any space at all: let LaTeX do the formatting work!

(In this case, the problem is the use of `\begin{center}` to center the image. The `center` environment adds extra space that can be avoided if `\centering` is used instead.)

Alternatively you might use the `picins` package instead of the wrapfigure package which produces a correct version without the excess white space out of the box without any hand tuning.

There is also an alternative to `wrapfig`: the package `floatflt` [2] - for documentation see [3].

Subfloats

A useful extension is the `subfig` package [4] (not to be confused with the deprecated `subfigure` package), which uses subfloats within a single float. This gives the author the ability to have subfigures within figures, or subtables within table floats. Subfloats have their own caption, and an optional global caption. An example will best illustrate the usage of this package:

```
\usepackage{subfig}

\begin{figure}
  \centering
  \subfloat[A
gull]{\label{fig:gull}\includegraphics[width=0.3\textwidth]{gull}}

  \subfloat[A
tiger]{\label{fig:tiger}\includegraphics[width=0.3\textwidth]{tiger}}
  \subfloat[A
mouse]{\label{fig:mouse}\includegraphics[width=0.3\textwidth]{mouse}}
  \caption{Pictures of animals}
  \label{fig:animals}
\end{figure}
```

(a) A gull (b) A tiger (c) A mouse

Figure 1: Pictures of animals

You will notice that the figure environment is set up as usual. You may also use a table environment for subtables. For each subfloat, you need to use:

```
\subfloat[sub caption]{ ... figure or table ... }
```

If you intend to cross-reference any of the subfloats, see where the label is inserted; `\caption` will provide the global caption.

`subfig` will arrange the figures or tables side-by-side providing they can fit, otherwise, it will automatically shift subfloats below. This effect can be added manually, by putting the newline command (`\\`) before the figure you wish to move to a newline.

Horizontal spaces between figures is controlled by one of several commands, which are placed in between each `\subfloat{}` command:

- Any whitespace (such as spaces, returns, and tabs) will result in one regular space
- Math spaces: `\qquad`, `\quad`, `\;`, and `\,`
- Generic space: `\hspace{`*`length`*`}`

- Automatically expanding/contracting space: `\hfill`

Wide figures in two column documents

If you are writing a document using two columns (i.e. you started your document with something like `\documentclass[twocolumn]{article}`), you might have noticed that you can't use floating elements that are wider than the width of a column (using a LaTeX notation, wider than `0.5\textwidth`), otherwise you will see the image overlapping with text. If you really have to use such wide elements, the only solution is to use the "starred" variants of the floating environments, that are `{figure*}` and `{table*}`. Those "starred" versions work exactly like the standard ones, but they will be as wide as the page, so you will get no overlapping.

A bad point of those environments is that they can be placed only at the top of the page or on their own page. If you try to specify their position using modifiers like *b* or *h* they will be ignored. Add `\usepackage{stfloats}` to the preamble in order to alleviate this problem with regard to placing these floats at the bottom of a page, using the optional specifier `[b]`. Default is `[tbp]`. However, *h* still does not work.

To prevent the figures from being placed out-of-order with respect to their "non-starred" counterparts, the package `fixltx2e` [5] should be used (e.g. `\usepackage{fixltx2e}`).

Custom Floats

If tables and figures are not adequate for your needs, then you always have the option to create your own! Examples of such instances could be source code examples, or maps. For a program float example, one might therefore wish to create a float named `program`. The package `float` is your friend for this task. *All commands to set up the new float must be placed in the preamble, and not within the document.*

1. Add `\usepackage{float}` to the preamble of your document
2. Declare your new float using: `\newfloat{type}{placement}{ext}[outer counter]`, where:
   - *type* - the new name you wish to call your float, in this instance, 'program'.
   - *placement* - t, b, p, or h (as previously described in Placement), where letters enumerate permitted placements.
   - *ext* - the file name extension of an auxiliary file for the list of figures (or whatever). Latex writes the captions to this file.
   - *outer counter* - the presence of this parameter indicates that the counter associated with this new float should depend on outer counter, for example 'chapter'.
3. The default name that appears at the start of the caption is the type. If you wish to alter this, use `\floatname{type}{floatname}`
4. Changing float style can be issued with `\floatstyle{style}` (Works on all subsequent `\newfloat` commands, therefore, must be inserted before `\newfloat` to be effective).
   - `plain` - the normal style for Latex floats, i.e., nothing!
   - `boxed` - a box is drawn that surrounds the float, and the caption is printed below.
   - `ruled` - the caption appears above the float, with rules immediately above and below. Then the float contents, followed by a final horizontal rule.

Float styles can also be customized as the second example below illustrates.

An example document using a new `program` float type:

```
\documentclass{article}

\usepackage{float}

\floatstyle{ruled}
\newfloat{program}{thp}{lop}
```

```
\floatname{program}{Program}

\begin{document}

\begin{program}
  \begin{verbatim}

class HelloWorldApp {
  public static void main(String[] args) {
    //Display the string
    System.out.println("Hello World!");
  }
}
\end{verbatim}
  \caption{The Hello World! program in Java.}
\end{program}

\end{document}
```

The `verbatim` environment is an environment that is already part of Latex. Although not introduced so far, its name is fairly intuitive! Latex will reproduce everything you give it, including new lines, spaces, etc. It is good for source code, but if you want to introduce a lot of code you might consider using the `listings` package, that was made just for it.

While this is useful, one should be careful when embedding the float within another float. In particular, the error `not in outer par mode` may occur. One solution might be to use the [H] option (not any other) on the inner float, as this option "pins" the inner float to the outer one.

Newly created floats with *newfloat* can also be used in combination with the *wrapfig* package from above. E.g. the following code creates a floating text box, which floats in the text on the right side of the page and is complete with caption, numbering, an index file with the extension .lob and a customization of the float's visual layout:

```
\documentclass{article}

% have hyperref package before float in order to get strange errors
with .\theHfloatbox
\usepackage[pdftex]{hyperref}

\usepackage{float}

%allows use of "@" before \begin{document}
\makeatletter

% this creates a custom and simpler ruled box style
\newcommand\floatc@simplerule[2]{{\@fs@cfont #1 #2}\par}
\newcommand\fs@simplerule{\def\@fs@cfont{\bfseries}\let\@fs@capt\floatc@simplerule
  \def\@fs@pre{\hrule height.8pt depth0pt \kern4pt}%
  \def\@fs@post{\kern4pt\hrule height.8pt depth0pt \kern4pt \relax}%
  \def\@fs@mid{\kern8pt}%
  \let\@fs@iftopcapt\iftrue}
```

```
% this code block defines the new and custom floatbox float environment
\floatstyle{simplerule}
\newfloat{floatbox}{thp}{lob}[section]
\floatname{floatbox}{Text Box}

\begin{document}

\begin{floatbox}{r}{}
  \textit{Bootstrapping} is a resampling technique used
  for robustly estimating statistical quantities, such as
  the model fit $R^2$. It offers some protection against
  the sampling bias.
  \caption{Bootstrapping}
\end{floatbox}

\end{document}
```

Caption Styles

To change the appearance of captions, use the `caption` [6] package. For example, to make all caption labels small and bold:

```
\usepackage[small,bf]{caption}
```

The KOMA script packages [7] have their own caption customizing features with e.g. `\captionabove`, `\captionformat` and `\setcapwidth`. However these definitions have limited effect on newly created float environments with the `wrapfig` package.

Labels in the figures

There is a LaTeX package lpic [8] to put LaTeX on top of included graphics, thus allowing to add TeX annotations to imported graphics. It defines a convenient interface to put TeX over included graphics, and allows for drawing a white background under the typeset material to overshadow the graphics. It is a better alternative for labels inside of graphics; you do not have to change text size when rescaling pictures, and all LaTeX power is available for labels.

A very similar package, with somewhat different syntax, is pinlabel [8]. The link given also points to the packages psfrag and overpic.

Summary

That concludes all the fundamentals of floats. You will hopefully see how much easier it is to let Latex do all the hard work and tweak the page layouts in order to get your figures in the best place. As always, the fact that LaTeX takes care of all caption and reference numbering is a great time saver.

This page uses material from Andy Roberts' Getting to grips with Latex [3] *with permission from the author.*

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/float/
[2] http://tug.ctan.org/tex-archive/macros/latex/contrib/floatflt/
[3] http://www.ctan.org/tex-archive/macros/latex/contrib/floatflt/floatflt.pdf
[4] http://www.ctan.org/tex-archive/macros/latex/contrib/subfig/subfig.pdf
[5] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=2colfltorder
[6] http://mirror.ctan.org/macros/latex/contrib/caption/caption-eng.pdf
[7] http://www.komascript.de/
[8] http://www.ctan.org/tex-archive/help/Catalogue/entries/pinlabel.html

Colors

Adding colors to your text is supported by the `color` package. Using this package, you can set the color of the font of the text, and set the background color of the page. You can use one of the predefined colors such as *white*, *red*, or *yellow*, or you can define your own named colors. It's also possible to color formulas in math-environments.

Adding the color package

To make use of these color features the color package must be inserted into the preamble.

```
\usepackage{color}
```

Entering colored text

The simplest way to type colored text is by:

```
\textcolor{declared-color}{text}
```

where *declared-color* is a color that was defined before by **`\definecolor`** .

Another possible way is by

```
{\color{declared-color} text}
```

that will switch the standard text color to the color you want. It will work until the end of the current TeX group. For example:

```
\emph{some black text, \color{red} followed by a red fragment}, going
black again.
```

The difference between **`\textcolor`** and **`\color`** is the same as that between **`\texttt`** and **`\ttfamily`** , you can use the one you prefer.

You can change the background color of the whole page by:

```
\pagecolor{declared-color}
```

Entering colored background for the text

```
\colorbox{declared-color}{text}
```

If the background color and the text color is changed, then:

```
\colorbox{declared-color1}{\color{declared-color2} text}
```

There is also \fcolorbox to make framed background color in yet another color:

```
\fcolorbox{declared-color1}{declared-color2}{text}
```

Predefined colors

The predefined color names are `white`, `black`, `red`, `green`, `blue`, `cyan`, `magenta`, `yellow`. There may be other pre-defined colors on your system, but these should be available on all systems.

If you would like a color not pre-defined, you can use one of the 68 dvips colors, or define your own. These options are discussed in the following sections

The 68 standard colors known to dvips

Invoke the package with the usenames and dvipsnames option. If you are using TikZ package you must declare the color package before that, otherwise it will not work.

```
\usepackage[usenames,dvipsnames]{color}
```

Apricot	Aquamarine	Bittersweet	Black
Blue	BlueGreen	BlueViolet	BrickRed
Brown	BurntOrange	CadetBlue	CarnationPink
Cerulean	CornflowerBlue	Cyan	Dandelion
DarkOrchid	Emerald	ForestGreen	Fuchsia
Goldenrod	Gray	Green	GreenYellow
JungleGreen	Lavender	LimeGreen	Magenta
Mahogany	Maroon	Melon	MidnightBlue
Mulberry	NavyBlue	OliveGreen	Orange
OrangeRed	Orchid	Peach	Periwinkle
PineGreen	Plum	ProcessBlue	Purple
RawSienna	Red	RedOrange	RedViolet
Rhodamine	RoyalBlue	RoyalPurple	RubineRed
Salmon	SeaGreen	Sepia	SkyBlue
SpringGreen	Tan	TealBlue	Thistle
Turquoise	Violet	VioletRed	White
WildStrawberry	Yellow	YellowGreen	YellowOrange

Defining new colors

If the predefined colors are not adequate, you may wish to define your own.

Place

Define the colors in the *preamble* of your document. (Reason: Do so in the preamble, so that you can already refer to them in the preamble, which is useful, for instance, in an argument of another package that supports colors as arguments, such as listings package.)

Method

To define a new color, follow the following example, which defines orange for you, by setting the red to the maximum, the green to one half (0.5), and the blue to the minimum:

```
\definecolor{orange}{rgb}{1,0.5,0}
```

You can use small letters **rgb** and choose a value between 0 and 1 or use capital letters **RGB** and choose a value between 0 and 255. The following code should give a similar results to the last code chunk.

```
\definecolor{orange}{RGB}{255,127,0}
```

Trick : When surfing on the web, you can get hexadecimal code for each color on a web page using the *colorzilla* extension to Firefox.

In the abstract, the colors are defined following this scheme:

```
\definecolor{''name''}{''model''}{''color-spec''}
```

where:

- *name* is the name of the color; you can call it as you like
- *model* is the way you *describe* the color, and is one of *gray*, *rgb* and *cmyk*.
- *color-spec* is the description of the color

Color Models

Among the models you can use to describe the color are the following (several more are described in the xcolor manual [1]):

Color Models

Model	Description	Color Specification	Example
`gray`	Shades of gray.	just one number between 0 (black) and 1 (white), so 0.95 will be very light gray, 0.30 will be dark gray	`\definecolor{light-gray}{gray}{0.95}`
`rgb`	Red, Green, Blue	three numbers given in the form *red,green,blue*; the quantity of each color is represented with a number between 0 and 1	`\definecolor{orange}{rgb}{1,0.5,0}`
`RGB`	Red, Green, Blue	three numbers given in the form *red,green,blue*; the quantity of each color is represented with a number between 0 and 255	`\definecolor{orange}{RGB}{255,127,0}`
`HTML`	Red, Green, Blue	six hexadecimal numbers given in the form *RRGGBB*; similar to what is used in HTML	`\definecolor{orange}{HTML}{FF7F00}`
`cmyk`	Cyan, Magenta, Yellow, Black	four numbers given in the form *cyan,magenta,yellow,black*	`\definecolor{orange}{cmyk}{0,0.5,1,0}`

Advanced color settings

The xcolor package provides extended versions of the described color related commands. Tints can be defined as follow:

```
\color{blue!20} \color{blue!20!black} \color{blue!20!black!30!green}
```

The first specifies 20 percent blue, the second is a mixture of 20 percent blue and 80 percent black and the last one is a mixture of 20 percent blue, 30 percent black and 50 percent green.

Other features include: support of hsb color model, html style notation, special row coloring support in tables and more.

Sources

- The xcolor manual [1]

References

[1] http://mirror.ctan.org/macros/latex/contrib/xcolor/xcolor.pdf

Glossary

Many technical documents use terms or acronyms unknown to the general population. It's common practice to add glossaries to make those works more understandable.

The glossaries package was created to assist users in creating glossaries. It supports multiple glossaries, acronyms and symbols.

It replaces the glossary package and can be used instead of the nomencl package.

Using glossaries

To enable the use of glossaries package, you have to load the package:

```
\usepackage{glossaries}
```

if you will be using `xindy` (highly recommended) rather than `makeindex` you need to specify xindy option:

```
\usepackage[xindy]{glossaries}
```

for the glossary to show up in Table of Contents you need to additionally add toc option:

```
\usepackage[toc]{glossaries}
```

the glossary index won't be generated until you place the following command in document preamble:

```
\makeglossaries
```

Note that the links in generated glossary won't be “clickable” unless you load this package *after* the hyperref package.

Windows users will need to install Perl for makeglossaries to work.

Defining glossary entries

To use an entry from glossary you first need to define it. There are few ways to define an entry depending on what you define and how it is going to be used.

Note that a defined entry *won't* be included in the printed glossary *unless* it is used in the document. This enables you to create a glossary of general terms and just `\include` it in all your documents.

Defining terms

To define a term in glossary you use `\newglossaryentry` macro:

```
\newglossaryentry{<label>}{<settings>}
```

<label> is a unique label used to identify an entry in glossary, <settings> are comma separated `key=value` pairs used to define an entry.

For example, to define a computer entry:

```
\newglossaryentry{computer} { description={is a programmable machine that
receives input, stores and manipulates data, and provides output in a useful
format} }
```

The above example defines an entry that has the same label and entry name. This is not always the case as the next entry will show:

```
\newglossaryentry{naiive} { name=na\"{\i}ve, description={is a French loanword
(adjective, form of naïf) indicating having or showing a lack of experience,
understanding or sophistication} }
```

When you define terms, you need to remember that they will be sorted by `makeindex` or `xindy`. While `xindy` is a bit more LaTeX aware, it does it by ommiting latex macros (`\"{\i` }) thus incorrectly sorting the above example as `nave`. `makeindex` won't fare much better, because it doesn't understand TeX macros, it will interpret the word exactly as it was defined, putting it inside symbol class, before words beginning with `naa`. Therefore it's needed to extend our example and specify how to sort the word:

```
\newglossaryentry{naiive} { name=na\"{\i}ve, description={is a French loanword
(adjective, form of naïf) indicating having or showing a lack of experience,
understanding or sophistication}, sort=naive }
```

You can also specify plural forms, if they are not formed by adding “s” (we will learn how to use them in next section):

```
\newglossaryentry{Linux} { description={is a generic term referring to the
family of Unix-like computer operating systems that use the Linux kernel},
plural=Linuces }
```

Defining symbols

Defined entries can also be symbols:

```
\newglossaryentry{pi} { name={\ensuremath{\pi
```

,

```
 description={ratio of circumference of circle to its
              diameter},
 sort=pi
```

} }}

You can also define both a name and a symbol:

```
\newglossaryentry{real number} { name={real number}, description={include both
rational numbers, such as $42$ and $\frac{-23}{129}$, and irrational numbers,
such as $\pi$ and the square root of two; or, a real number can be given by an
infinite decimal representation, such as $2.4871773339\ldots$ where the digits
continue in some way; or, the real numbers may be thought of as points on an
infinitely long number line}, symbol={\ensuremath{\mathbb{R
```

} } }} Note that not all glossary styles show defined symbols.

Defining acronyms

Defined acronyms can be put in separate list if you use `acronym` package option:

```
\usepackage[acronym]{glossaries}
```

To define a new acronym you use the **\newacronym** macro:

```
\newacronym{<label>}{<abbrv>}{<full>}
```

where <label> is the unique label identifying the acronym, <abbrv> is the abbreviated form of the acronym and <full> is the expanded text. For example:

```
\newacronym{lvm}{LVM}{Logical Volume Manager}
```

Using defined terms

When you have defined a term, you can use it in a document. There are many different commands used to refer to glossary terms.

General references

A general reference is used with **\gls** command. If, for example, you have glossary entries defined as those above, you might use it in this way:

```
\Gls{naiive} people don't know about alternative
\gls{computer} operating systems: \glspl{Linux}, BSDs and
GNU/Hurd.
```

Naïve people don't know about alternative computer operating systems: Linuces, BSDs and GNU/Hurd. </pre>

Description of commands used in above example:

```
\gls{<label>}
```

This command prints the term associated with <label> passed as its argument. If the hyperref package was loaded before glossaries it will also be hyperlinked to the entry in glossary.

```
\glspl{<label>}
```

This command prints the plural of the defined therm, other than that it behaves in the same way as `gls` .

```
\Gls{<label>}
```

This command prints the singular form of the term with the first character converted to upper case.

```
\Glspl{<label>}
```

This command prints the plural form with first letter of the term converted to upper case.

Referring acronyms

Acronyms behave a bit differently than normal glossary terms. On first use the **`\gls`** command will display "<full> (<abbrv>)". On subsequent uses only the abbreviation will be displayed.

To reset the first use of an acronym use:

```
\glsreset{<label>}
```

or, if you want to reset the use status of all acronyms:

```
\glsresetall
```

Displaying the Glossary

To display the sorted list of terms you need to put:

```
\printglossaries
```

at the place you want the glossary and the list of acronyms to be printed.

Then you have to do three steps:

1. Build LaTeX document, this will generate the files used by `makeglossaries`
2. Run the indexing/sorting, the recommended way is to use `makeglossaries` (a script that runs `xindy` or `makeindex` depending on options in the document with correct encoding and language settings):

```
makeglossaries <myDocument>
```

3. Build LaTeX document again to get document with glossary entries

If your entries are interlinked (entries themselves link to other entries with **`\gls`**) you will need to run points 1 and 2 twice for a total of five program invocations: 1, 2, 1, 2, 3.

References

- The glossaries documentation, http://tug.ctan.org/tex-archive/macros/latex/contrib/glossaries/
- *Using LaTeX to Write a PhD Thesis*, Nicola L.C. Talbot, http://theoval.cmp.uea.ac.uk/~nlct/latex/thesis/node25.html

Letters

Sometimes the mundane things are the most painful. However, it doesn't have to be that way because of evolved, user-friendly templates. Thankfully, LaTeX allows for very quick letter writing, with little hassle.

The letter class

To write letters use the standard document class *letter*.

You can write multiple letters in one LaTeX file - start each one with \begin{letter}{*recipient*} and end with \end{letter}. You can leave *recipient* blank. Each letter consists of four parts:

1. opening (like \opening{Dear Sir or Madam,} or \opening{Dear Kate,}
2. main body - written as usual in LaTeX
3. closing (like \closing{Yours sincerely,}

   LaTeX will leave some space after closing for your hand-written signature; then it will put your name and surname, if you have declared them.
4. additional elements: post scripta, carbon copy and list of enclosures

If you want your name, address and telephone number to appear in the letter, you have to declare them first signature, address and telephone.

The output letter will look like this:

21 Bridge Street
Smallville
Dunwich DU3 4WE

April 24, 2007

Director
Doe & Co
35 Anthony Road
Newport
Ipswich IP3 5RT

Dear Sir or Madam:

I am writing to you on behalf of the Wikipedia project (http://www.wikipedia.org/), an endeavour to build a fully-fledged multilingual encyclopaedia in an entirely open manner, to ask for permission to use your copyrighted material.

...

That said, allow me to reiterate that your material will be used to the noble end of providing a free collection of knowledge for everyone; naturally enough, only if you agree. If that is the case, could you kindly fill in the attached form and post it back to me? We shall greatly appreciate it.

Thank you for your time and consideration.

I look forward to your reply.

Yours Faithfully,

Joe Bloggs

P.S. You can find the full text of GFDL license at http://www.gnu.org/copyleft/fdl.html.

encl: Copyright permission form

A sample letter.

Here is the example's code:

```
\documentclass{letter}
\usepackage{hyperref}
\signature{Joe Bloggs}
\address{21 Bridge Street \\ Smallville \\ Dunwich DU3 4WE}
\begin{document}
```

```
\begin{letter}{Director \\ Doe \& Co \\ 35 Anthony Road
\\ Newport \\ Ipswich IP3 5RT}
\opening{Dear Sir or Madam:}

I am writing to you on behalf of the Wikipedia project
(http://www.wikipedia.org/),
an endeavour to build a fully-fledged multilingual encyclopaedia in an
entirely
open manner, to ask for permission to use your copyrighted material.

% The \ldots command produces dots in a way that will not upset
% the typesetting of the document.
\ldots

That said, allow me to reiterate that your material will be used to the
 noble end of
providing a free collection of knowledge for everyone; naturally
enough, only if you
agree. If that is the case, could you kindly fill in the attached form
and post it
back to me? We shall greatly appreciate it.

Thank you for your time and consideration.

I look forward to your reply.

\closing{Yours Faithfully,}

\ps{P.S. You can find the full text of GFDL license at
\url{http://www.gnu.org/copyleft/fdl.html}.}
\encl{Copyright permission form}

\end{letter}

\end{document}
```

To move the closing and signature parts to the left, insert the following before \begin{document}:

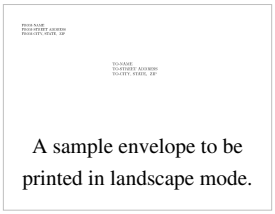
```
\longindentation=0pt
```

The amount of space to the left can be adjusted by increasing the 0pt.

Envelopes

Here is a relatively simple envelope which uses the `geometry` package which is used because it vastly simplifies the task of rearranging things on the page (and the page itself).

```
% envelope.tex
\documentclass{letter}
\usepackage[left=1in,top=0.15in,papersize={4.125in,9.5in},landscape,twoside=false]{geometry}
\setlength\parskip{0pt}
\pagestyle{empty}

\begin{document}

FROM-NAME

FROM-STREET ADDRESS

FROM-CITY, STATE, \ ZIP

\vspace{1.0in}\large
\setlength\parindent{3.6in}

TO-NAME

TO-STREET ADDRESS

TO-CITY, STATE, \ ZIP

\end{document}
```



A sample envelope to be printed in landscape mode.

This will certainly take care of the spacing but the actual printing is between you and your printer. After all, different printers have different feeding mechanisms for envelopes. You may find the following commands useful for printing the envelope.

```
$ pdflatex envelope.tex
$ pdf2ps envelope.pdf
$ lpr -o landscape envelope.ps
```

Alternatively, you can use the latex dvi output driver.

In the first line, dvips command converts the .dvi file produced by latex into a .ps (PostScript) file. In the second line, the PostScript file is sent to the printer.

```
$ latex envelope.tex && dvips -t unknown -T 9.5in,4.125in envelope.dvi
$ lpr -o landscape envelope.ps
```

I have found that `pdflatex` creates the right page size but not `dvips` despite what it says in the `geometry` manual. It will never work though unless your printer settings are adjusted to the correct page style. These settings depend on the printer filter you are using and in CUPS might be available on the `lpr` command line if you are masochistic.

Windowed envelopes

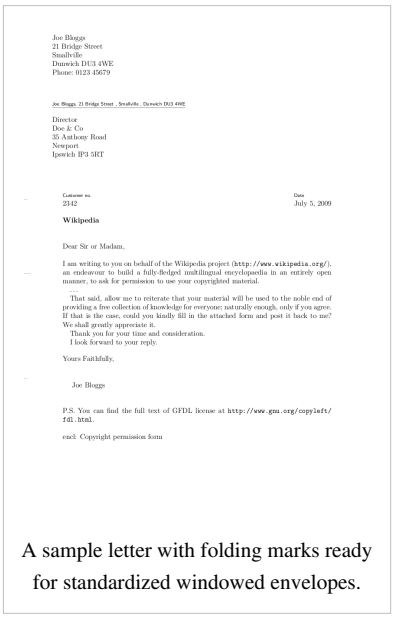
An alternative to separately printing addresses on envelopes is to use the letter class from the KOMA package. It supports additional features like folding marks and the correct address placement for windowed envelopes. Using the scrlttr2 document class from the KOMA package the example letter code is:

```
% koma_env.tex
\documentclass[a4paper]{scrlttr2}
\usepackage{lmodern}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[english]{babel}
\usepackage{url}



\setkomavar{fromname}{Joe Bloggs}
\setkomavar{fromaddress}{21 Bridge Street \\ Smallville \\ Dunwich DU3
4WE}
\setkomavar{fromphone}{0123 45679}

\begin{document}

\begin{letter}{Director \\ Doe \& Co \\ 35 Anthony Road
\\ Newport \\ Ipswich IP3 5RT}

\KOMAoptions{fromphone=true,fromfax=false}
\setkomavar{subject}{Wikipedia}
\setkomavar{customer}{2342}
\opening{Dear Sir or Madam,}

I am writing to you on behalf of the Wikipedia project
(\url{http://www.wikipedia.org/}), an endeavour to build a
fully-fledged multilingual encyclopaedia in an entirely open
manner, to ask for permission to use your copyrighted material.

\ldots

That said, allow me to reiterate that your material will be used
to the noble end of providing a free collection of knowledge for
everyone; naturally enough, only if you agree. If that is the
case, could you kindly fill in the attached form and post it back
to me? We shall greatly appreciate it.
```

```
Thank you for your time and consideration.

I look forward to your reply.

\closing{Yours Faithfully,}
\ps{P.S. You can find the full text of GFDL license at
\url{http://www.gnu.org/copyleft/fdl.html}.}
\encl{Copyright permission form}

\end{letter}

\end{document}
```

The output is generated via

```
$ pdflatex koma_env
```


Joe Bloggs
21 Bridge Street
Smallville
Dunwich DU3 4WE
Phone: 0123 45679

Director
Doe & Co
35 Anthony Road
Newport
Ipswich IP3 5RT

Customer no. 2142
Date July 5, 2009

Wikipedia

Dear Sir or Madam,

I am writing to you on behalf of the Wikipedia project (http://www.wikipedia.org/), an endeavour to build a fully-fledged multilingual encyclopaedia in an entirely open manner, to ask for permission to use your copyrighted material.

That said, allow me to reiterate that your material will be used to the noble end of providing a free collection of knowledge for everyone; naturally enough, only if you agree. If that is the case, could you kindly fill in the attached form and post it back to me? We shall greatly appreciate it.
Thank you for your time and consideration.
I look forward to your reply.

Yours Faithfully,

Joe Bloggs

P.S. You can find the full text of GFDL license at http://www.gnu.org/copyleft/fdl.html.

encl: Copyright permission form

A sample letter with folding marks ready for standardized windowed envelopes.

Folding the print of the resulting file koma_env.pdf according the folding marks it can be placed into standardized windowed envelopes DIN C6/5, DL, C4, C5 or C6.

In addition to the default, the KOMA-package includes predefined format definitions for different standardized Swiss and Japanese letter formats.

Reference: letter.cls commands

command	description
\name{}	
\signature{}	
\address{}	
\location{}	
\telephone{}	
\makelabels	
\stopbreaks	
\startbreaks	
\opening{}	
\closing{}	
\cc{}	Start a parbox introduced with \ccname:
\encl{}	Start a parbox introduced with \enclname:
\ps	Begins a new paragraph, normally at the close of the letter
\stopletter	(empty)
\returnaddress	(empty)
\startlabels	
\mlabel{}{}	
\descriptionlabel{}	
\ccname	"cc"
\enclname	"encl"
\pagename	"Page"
\headtoname	"To"
\today	Long form date

environment	Description
letter{}	See main article
description	
verse	
quotation	
quote	

Sources

- KOMA-Script - The Guide [1]

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/koma-script/scrguien.pdf

Teacher's Corner

Intro

LaTeX has specific features for teachers. We present the **exam** class[1] which is useful for designing exams and exercises with solutions. Interested people could also have a look at the **probsoln** package[2] or the **mathexm** document class[3] .

The exam class

We present the **exam** class. The exam class is well suited to design exams with solutions. You just have to specify in the preamble if you want the solutions to be printed or not. You can also count the number of points.

Preamble

In the preamble you can specify the following lines :

```
\documentclass[a4paper,11pt]{exam} \printanswers % If you want to print answers
% \noprintanswers % If you don't want to print answers \addpoints % if you want
to count the points % \noaddpoints % if you don't want to count the points %
Specifies the way question are displayed:
\qformat{\textbf{Question\thequestion}\quad(\thepoints)\hfill}
\usepackage{color} % defines a new color
\definecolor{SolutionColor}{rgb}{0.8,0.9,1} % light blue \shadedsolutions %
defines the style of the solution environment % \framedsolutions % defines the
style of the solution environment % Defines the title of the solution
environment:
\renewcommand{\solutiontitle}{\noindent\textbf{Solution:}\par\noindent}
```

You can replace the 3 first lines with the following :

```
\documentclass[a4paper,11pt,answers,addpoints]{exam}
```

Document

- The exam is included in the **questions** environment.
- The command **\question** introduces a new question.
- The number of points is specified in squared brackets.
- The solution is given in the **solution** environment. It appears only if **\printanswers** or **answers** as an option of the **\documentclass** are specified in the preamble.

Here is an example :

```
\begin{questions} % Begins the questions environment
\question[2] What is the solution? % Introduces a new question which is
 worth 2 points
\begin{solution}
Here is the solution
\end{solution}
\question[5] What is your opinion?
\begin{solution}
This is my opinion
\end{solution}
```

```
\end{questions}
```

It is also possible to add stuff only if answers are printed using the **\ifprintanswers** command.

```
\ifprintanswers
Only if answers are printed
\else
Only if answers are not printed
\fi
```

Introduction

The macro **\numquestions** gives the total number of questions. The macro **\numpoints** gives the total number of points.

```
\begin{minipage}{.8\textwidth}
This exam includes \numquestions\ questions. The total number of points
 is \numpoints.
\end{minipage}
```

The backslash after **\numquestion** prevents the macro from gobbling the following whitespace as it normally would.

References

[1] examdoc (http://www-math.mit.edu/~psh/exam/examdoc.pdf) Using the exam document class

[2] Probsoln (http://www.tex.ac.uk/tex-archive/macros/latex/contrib/probsoln/probsoln.pdf) creating problem sheets optionally with solutions

[3] http://mat140.bham.ac.uk/~richard/programming/tex/exams/msexdoc.pdf

Presentations

LaTeX can be used for creating presentations. There are several packages for the task, including the Beamer package.

The Beamer package

The beamer package is provided with most LaTeX distributions, but is also available from CTAN [1]. If you use MikTeX, all you have to do is to include the beamer package and let LaTeX download all wanted packages automatically. The documentation [2] explains the features in great detail. You can also have a look at the PracTex article **Beamer by example**[3]

Introductory example

The beamer package is loaded by calling the `beamer` class:

```
\documentclass{beamer}
```

The usual header information may then be specified. Note that if you are compiling with XeTeX then you should use

```
\documentclass[xetex,mathserif,serif]{beamer}
```

Inside the usual `document` environment, multiple `frame` environments specify the content to be put on each slide. The `frametitle` command specifies the title for each slide (See image):

```
\begin{document}
  \begin{frame}
    \frametitle{This is the first slide}
    %Content goes here
  \end{frame}
  \begin{frame}
    \frametitle{This is the second slide}
    \framesubtitle{A bit more information about this}
    %More content goes here
  \end{frame}
% etc
\end{document}
```

Usual environments (`itemize`,`enumerate`, `equation`, etc.) may be used as usual.

Inside of frames, you can use environments like `block`, `theorem`, `proof`, ... Also, `\maketitle` is possible to create the frontpage, if `title` and `author` is set.

Trick : Instead of using *\begin{frame}…\end{frame}*, you can also use *\frame{…}*.

For the actual talk, if you can compile it with pdfLaTeX then you could use Adobe Reader with its fullscreen mode. If you want to navigate in your presentation, you can use the almost invisible links in the bottom right corner without leaving the fullscreen mode.

Document Structure

Title page and information

You give information about authors, titles and dates in the preamble

```
\title[Crisis]%(optional, only for long titles)
{The Economics of Financial Crisis}
\subtitle{Evidence from India}
\author[Author, Anders] % (optional, for multiple authors)
{F.~Author\inst{1} \and S.~Anders\inst{2}}
\institute[Universitäten Hier und Dort] % (optional)
{
  \inst{1}%
  Institut für Informatik\\
  Universität Hier
  \and
  \inst{2}%
  Institut für theoretische Philosophie\\
```

```
  Universität Dort
}
\date[KPT 2004] % (optional)
{Konferenz über Präsentationstechniken, 2004}
\subject{Informatik}
```

In the document, you add the title page :

```
\frame{\titlepage}
```

Table of Contents

You can print the table of contents and highlight the current section/subsection by typing :

```
\begin{frame}
\frametitle{Table of Contents}
\tableofcontents[currentsection]
\end{frame}
```

You can automatically print the table of contents at the beginning of each section by adding in the preamble the following line.

```
\AtBeginSection[]
{
  \begin{frame}
    \frametitle{Table of Contents}
    \tableofcontents[currentsection]
  \end{frame}
}
```

You can do the same for subsections :

```
\AtBeginSubsection[]
{
  \begin{frame}
    \frametitle{Table of Contents}
    \tableofcontents[currentsection,currentsubsection]
  \end{frame}
}
```

References (Beamer)

```
\begin{frame}[allowframebreaks]
  \frametitle<presentation>{Weiterf¸hrende Literatur}
  \begin{thebibliography}{10}
  \beamertemplatebookbibitems
  \bibitem{Autor1990}
    A.~Autor.
    \newblock {\em Einf¸hrung in das Pr‰sentationswesen}.
    \newblock Klein-Verlag, 1990.
  \beamertemplatearticlebibitems
  \bibitem{Jemand2000}
    S.~Jemand.
```

```
    \newblock On this and that.
    \newblock {\em Journal of This and That}, 2(1):50--100, 2000.
  \end{thebibliography}
\end{frame}
```

Style

Themes

The first solution is to use a built-in theme such as Warsaw, Berlin, etc. The second solution is to specify colors, inner themes and outer themes.

The Built-in solution

To the preamble you can add the following line:

```
\usetheme{Warsaw}
```

to use the "Warsaw" theme. `Beamer` has several themes, many of which are named after cities (e.g. Barcelona, Madrid, Berlin, etc.). Color themes, typically with animal names, can be specified with

```
\usecolortheme{beaver}
```

This Theme Matrix [4] contains the various theme and color combinations included with `beamer`. For more customizing options, have a look to the official documentation included in your distribution of `beamer`, particularly the part *Change the way it looks*.

The list of all themes : Antibes, Bergen, Berkeley, Berlin, Boadilla, Copenhagen, Darmstadt, Dresden, Frankfurt, Goettingen, Hannover, Ilmenau, JuanLesPins, Luebeck, Madrid, Malmoe, Marburg, Montpellier, PaloAlto, Pittsburgh, Rochester, Singapore, Szeged, Warsaw, boxes, default

The *do it yourself* solution

First you can specify the *outertheme*. The outertheme defines the head and the footline of each slide.

```
\useoutertheme{infolines}
```

Here is a list of all available outer themes

- infolines
- miniframes
- shadow
- sidebar
- smoothbars
- smoothtree
- split
- tree

Then you can add the innertheme :

```
\useinnertheme{rectangles}
```

Here is a list of all available inner themes :

- rectangles
- circles
- inmargin
- rounded

You can define the color of every element :

```
\setbeamercolor{alerted text}{fg=orange}
\setbeamercolor{background canvas}{bg=white}
\setbeamercolor{block body alerted}{bg=normal text.bg!90!black}
\setbeamercolor{block body}{bg=normal text.bg!90!black}
\setbeamercolor{block body example}{bg=normal text.bg!90!black}
\setbeamercolor{block title alerted}{use={normal text,alerted
text},fg=alerted text.fg!75!normal text.fg,bg=normal text.bg!75!black}
\setbeamercolor{block title}{bg=blue}
\setbeamercolor{block title example}{use={normal text,example
text},fg=example text.fg!75!normal text.fg,bg=normal text.bg!75!black}
\setbeamercolor{fine separation line}{}
\setbeamercolor{frametitle}{fg=brown}
\setbeamercolor{item projected}{fg=black}
\setbeamercolor{normal text}{bg=black,fg=yellow}
\setbeamercolor{palette sidebar primary}{use=normal text,fg=normal
text.fg}
\setbeamercolor{palette sidebar
quaternary}{use=structure,fg=structure.fg}
\setbeamercolor{palette sidebar
secondary}{use=structure,fg=structure.fg}
\setbeamercolor{palette sidebar tertiary}{use=normal text,fg=normal
text.fg}
\setbeamercolor{section in sidebar}{fg=brown}
\setbeamercolor{section in sidebar shaded}{fg= grey}
\setbeamercolor{separation line}{}
\setbeamercolor{sidebar}{bg=red}
\setbeamercolor{sidebar}{parent=palette primary}
\setbeamercolor{structure}{bg=black, fg=green}
\setbeamercolor{subsection in sidebar}{fg=brown}
\setbeamercolor{subsection in sidebar shaded}{fg= grey}
\setbeamercolor{title}{fg=brown}
\setbeamercolor{titlelike}{fg=brown}
```

Remember that you can define your own colors :

```
\definecolor{chocolate}{RGB}{33,33,33}
```

You can also define the style of blocks :

```
\setbeamertemplate{blocks}[shadow=false]
```

```
\setbeamertemplate{background canvas}[vertical
shading][bottom=white,top=structure.fg!25]
```

You can also suppress the navigation bar:

```
\beamertemplatenavigationsymbolsempty
```

Fonts

You may also change the fonts for particular elements. If you wanted the title of the presentation as rendered by `\frame{\titlepage}` to occur in a serif font instead of the default sanserif, you would use:

```
\setbeamerfont{title}{family=\rm}
```

You could take this a step further if you are using OpenType fonts with Xe(La)TeX and specify a serif font with increased size and oldstyle proportional alternate number glyphs:

```
\setbeamerfont{title}{family=\rm\addfontfeatures{Scale=1.18,
Numbers={Lining, Proportional}}}
```

Math Fonts

The default settings for `beamer` use a different set of math fonts than one would expect from creating a simple math article. One quick fix for this is to use at the beginning of the file the option `mathserif`

```
\documentclass[mathserif]{beamer}
```

Others have proposed to use the command

```
\usefonttheme[onlymath]{serif}
```

but it is not clear if this works for absolutely every math character.

Frames Options

The *plain* option. Sometimes you need to include a large figure or a large table and you don't want to have the bottom and the top of the slides. In that case, use the plain option :

```
\frame[plain]{
…
}
```

If you want to include lots of text on a slide, use the *shrink* option.

```
\frame[shrink]{
…
}
```

Text animations

You can simply use the *pause* statement :

```
\begin{frame}
\frametitle{Some background}
We start our discussion with some concepts.
\pause
The first concept we introduce originates with Erd\H os.
\end{frame}
```

For text animations, for example in the itemize environment, you can write:

```
\begin{itemize}
  \item This one is always shown
  \item<1-> The first time
  \item<2-> The second time
```

```
  \item<1-> Also the first time
  \only<1-> This one is shown at the first time, but it will hide soon.
\end{itemize}
```

```
\begin{frame}
      \frametitle{`Hidden higher-order concepts?'}
      \begin{itemize}[<+->]
      \item The truths of arithmetic which are independent of PA in
some
      sense themselves `{contain} essentially {\color{blue}{hidden
higher-order}},
       or infinitary, concepts'???
      \item `Truths in the language of arithmetic which \ldots
      \item      That suggests stronger version of Isaacson's thesis.
      \end{itemize}
\end{frame}
```

Handout mode

In beamer class, the default mode is *presentation* which makes the slides. However, you can work in a different mode that is called *handout* by setting this option when calling the class:

```
\documentclass[12pt,handout]{beamer}
```

This mode is useful to see each slide only one time with all its stuff on it, making the itemize<+-> to be there all at once (for instance, printable version). Nevertheless, this makes an issue when working with the only command, because its purpose is to have *only* some text or figures at a time and not all of them together.

If you want to solve this, you can add a statement to precise the behavior it must have when dealing with only commands in handout mode. Suppose you have a code like this

```
\only<1>{\includegraphics{pic1.eps}}
\only<2>{\includegraphics{pic2.eps}}
```

These pictures being completely different, you want them both in the handout, but they cannot be both on the same slide since they are large. The solution is to add the handout statement to have the following:

```
\only<1| handout:1>{\includegraphics{pic1.eps}}
\only<2| handout:2>{\includegraphics{pic2.eps}}
```

This will ensure the handout will make a slide for each picture.

Now imagine you still have your two pictures with the only statements, but the second one show the first one plus some other graphs and you don't need the first one to appear in the handout. You can thus precise the handout mode not to include some only commands by:

```
\only<1| handout:0>{\includegraphics{pic1.eps}}
\only<2>{\includegraphics{pic2.eps}}
```

The command can also be used to hide frames, e.g.

```
\begin{frame}<handout:0>
```

or even, if you have written a frame that you don't want anymore but maybe you will need it later, you can write

```
\begin{frame}<0| handout:0>
```

and this will hide your slide in both modes. (The order matters. Don't put handout:0|beamer:0 or it won't work.)

A last word about the handout mode is about the notes. Actually, the full syntax for a frame is

```
\begin{frame}
...
\end{frame}
\note{...}
\note{...}
...
```

and you can write your notes about a frame in the field *note* (many of them if needed). Using this, you can add an option to the class calling, either

```
\documentclass[12pt,handout,notes=only]{beamer}
```

or

```
\documentclass[12pt,handout,notes=show]{beamer}
```

The first one is useful when you make a presentation to have only the notes you need, while the second one could be given to those who have followed your presentation or those who missed it, for them to have both the slides with what you said.

Note that the 'handout' option in the \documentclass line suppress all the animations.

Important: the *notes=only* mode is **literally** doing only the notes. This means there will be no output file but the DVI. Thus it requires you to have run the compilation in another mode before. If you use separate files for a better distinction between the modes, you may need to copy the .aux file from the handout compilation with the slides (w/o the notes).

Columns and Blocks

There are two handy environments for structuring a slide: "blocks", which divide the slide (horizontally) into headed sections, and "columns" which divides a slide (vertically) into columns.

Columns

Example

```
\begin{frame}
    \begin{columns}[c] % the "c" option specifies center vertical
alignment
    \column{.5\textwidth} % column designated by a command
     Contents of the first column
    \column{.5\textwidth}
     Contents split \\ into two lines
    \end{columns}
\end{frame}

\begin{frame}
     \begin{columns}[t] % contents are top vertically aligned
     \begin{column}[T]{5cm} % each column can also be its own
environment
     Contents of first column \\ split into two lines
```

```
    \end{column}
    \begin{column}[T]{5cm} % alternative top-align that's better for
graphics
        \includegraphics[height=3cm]{graphic.png}
    \end{column}
    \end{columns}
\end{frame}
```

Blocks

Enclosing text in the *block* environment creates a distinct, headed block of text. This allows to visually distinguish parts of a slide easily. There are three basic types of block. Their formating depends on the theme being used.

Simple

```
\begin{frame}

   \begin{block}{This is a Block}
      This is important information
   \end{block}

   \begin{alertblock}{This is an Alert block}
   This is an important alert
   \end{alertblock}

   \begin{exampleblock}{This is an Example block}
   This is an example
   \end{exampleblock}

\end{frame}
```

PDF options

You can specify the default options of your PDF.

```
\hypersetup{pdfstartview={FitH}} % By default the pdf fit the width of
the screen.
```

The powerdot package

The powerdot package is available from CTAN [5]. The documentation [6] explains the features in great detail.

The powerdot package is loaded by calling the `powerdot` class:

```
\documentclass{powerdot}
```

The usual header information may then be specified.

Inside the usual `document` environment, multiple `slide` environments specify the content to be put on each slide.

```
\begin{document}
  \begin{slide}{This is the first slide}
    %Content goes here
  \end{slide}
```

```
  \begin{slide}{This is the second slide}
    %More content goes here
  \end{slide}
% etc
\end{document}
```

References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/
[2] http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf
[3] Andrew Mertz and William Slough *Beamer by Example*
[4] http://www.hartwork.org/beamer-theme-matrix/
[5] http://www.ctan.org/tex-archive/macros/latex/contrib/powerdot/
[6] http://mirrors.ctan.org/macros/latex/contrib/powerdot/powerdot.pdf

Links

- Wikipedia:Beamer (LaTeX)
- beamer user guide (http://www.ctan.org/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf) (pdf) from CTAN
- A tutorial for creating a presentation using beamer package (http://www.math-linux.com/spip.php?article77)
- The powerdot class (http://www.ctan.org/get/macros/latex/contrib/powerdot/doc/powerdot.pdf) (pdf) from CTAN
- Making LaTeX Beamer Presentations (http://happymutant.com/latex/misce/beamer.php)

Hyperlinks

LaTeX enables typesetting of hyperlinks, useful when the resulting format is PDF, and the hyperlinks can be followed. It does so using the package *hyperref*.

Hyperref

The package `hyperref` provides LaTeX the ability to create hyperlinks within the document. It works with *pdflatex* and also with standard "latex" used with dvips and ghostscript or dvipdfm to build a PDF file. If you load it, you will have the possibility to include interactive external links and all your internal references will be turned to hyperlinks. The compiler *pdflatex* makes it possible to create PDF files directly from the LaTeX source, and PDF supports more features than DVI. In particular PDF supports hyperlinks, and the only way to introduce them in LaTeX is using `hyperref`. Moreover, PDF can contain other information about a document such as the title, the author, etc., and you can edit those using this same package.

Usage

The basic usage with the standard settings is straightforward. Just load the package in the preamble, at the end of **all** the other packages but prior to other settings:

```
\usepackage{hyperref}
```

This will automatically turn all your internal references into hyperlinks. It won't affect the way to write your documents: just keep on using the standard `\label` -`\ref` system; with `hyperref` those "connections" will become links and you will be able to click on them to be redirected to the right page. Moreover the table of contents, list of figures/tables and index will be made of hyperlinks, too.

Commands

The package provides three useful commands for inserting links pointing outside the document.

\hyperref

Usage: `\hyperref[label_name]{''link text''}`

This will have the same effect as `\ref{label_name}` but will make the text *link text* a full link, instead. The two can be combined, for example in

```
we use \hyperref[mainlemma]{lemma \ref*{mainlemma}}
```

Note the `*` after `\ref` for avoiding nested hyperlinks.

If the lemma labelled as "mainlemma" was number 4.1.1, then the outputted text would be "we use lemma 4.1.1" with the hyperlink as expected.

\url

Usage: `\url{''my_url''}`

It will show the URL using a mono-spaced font and, if you click on it, your browser will be opened pointing at it.

\href

Usage: `\href{''my_url''}{''description''}`

It will show the string "description" using standard document font but, if you click on it, your browser will be opened pointing at "my_url". Here is an example:

```
\url{http://www.wikibooks.org}
\href{http://www.wikibooks.org}{wikibooks home}
```

Both point at the same page, but in the first case the URL will be shown, while in the second case the URL will be hidden. Note that, if you print your document, the link stored using `\href` will not be shown anywhere in the document.

What we can do with them

Website

Already discussed...

Mail address

A possible way to insert emails is by

```
\href{mailto:my_address@wikibooks.org}{my_address@wikibooks.org}
```

it just shows your email address (so people can know it even if the document is printed on paper) but, if the reader clicks on it, (s)he can easily send you an email. Or, to incorporate the `url` package's formatting and line breaking abilities into the displayed text, use[1]

```
\href{mailto:my_address@wikibooks.org}{\nolinkurl{my_address@wikibooks.org}}
```

When using this form, note that the `\nolinkurl` command is fragile and if the hyperlink is inside of a moving argument, it must be preceeded by a `\protect` command.

Local file

Files can also be linked using the url or the href commands. You simply have to add the string `run:` at the beginning of the link string:

```
\url{run:/path/to/my/file.ext}
\href{run:/path/to/my/file.ext}{text displayed}
```

You can use relative paths to link documents near the location of your current document; in order to do so, use the standard Unix-like notation (`./` is the current directory, `../` is the previous directory, etc.)

Hyperlink and Hypertarget

It is also possible to create an anchor anywhere in the document (with or without caption) and to link to it, with:

```
\hyperlink{label}{anchor caption}
```

and

```
\hypertarget{label}{link caption}
```

Customization

The standard settings should be fine for most users, but if you want to change something, you can easily do it. There are several variables you can change and there are two methods to pass those to the package. You can pass the options as an argument of the package when you load it (that's the standard way packages work), or you can use the `\hypersetup` command:

```
\hypersetup{''option1'' [, ...]}
```

you can pass as many options as you want; separate them with a comma. Options have to be in the form:

```
variable_name=new_value
```

exactly the same format has to be used if you pass those options to the package while loading it, like this:

```
\usepackage[''option1, option2'']{hyperref}
```

Here is a list of the possible variables you can change (for the complete list, see the official documentation). The default values are written in an upright font:

variable	values	comment
bookmarks	=true,*false*	show or hide the bookmarks bar when displaying the document
unicode	=false,*true*	allows to use characters of non-Latin based languages in Acrobat's bookmarks
pdfborder	={*RadiusH RadiusV Width* [*Dash-Pattern*]}	set the style of the border around a link. The first two parameters (RadiusH, RadiusV) have no effect in most pdf viewers. *Width* defines the thickness of the border. *Dash-Pattern* is a series of numbers separated by space and enclosed by box-brackets. It is an optional parameter to specify the length of each line & gap in the dash pattern. For example, {0 0 0.5 [3 3]} is supposed to draw a square box (no rounded corners) of width 0.5 and a dash pattern with a dash of length 3 followed by a gap of length 3. There is no uniformity in whether/how different pdf viewers render the dash pattern.
pdftoolbar	=true,*false*	show or hide Acrobat's toolbar
pdfmenubar	=true,*false*	show or hide Acrobat's menu
pdffitwindow	=true,*false*	resize document window to fit document size
pdfstartview	={FitH},*{FitV}*,etc[2] .	fit the width of the page to the window
pdftitle	={text}	define the title that gets displayed in the "Document Info" window of Acrobat
pdfauthor	={text}	the name of the PDF's author, it works like the one above
pdfsubject	={text}	subject of the document, it works like the one above
pdfcreator	={text}	creator of the document, it works like the one above
pdfproducer	={text}	producer of the document, it works like the one above
pdfkeywords	={text}	list of keywords, separated by brackets, example below
pdfnewwindow	(=true,*false)*	define if a new window should get opened when a link leads out of the current document
pagebackref	(=false,*true)*	activate back references inside bibliography. Must be specified as part of the *\usepackage{}* statement.
colorlinks	(=false,*true)*	surround the links by color frames (false) or colors the text of the links (true). The color of these links can be configured using the following options (default colors are shown):
linkcolor	=red	color of internal links (sections, pages, etc.)
linktoc	=*none*,section,*page*,*all*	defines which part of an entry in the table of contents is made into a link
citecolor	=green	color of citation links (bibliography)
filecolor	=magenta	color of file links
urlcolor	=cyan	color of URL links (mail, web)
linkbordercolor	={1 0 0}	color of frame around internal links (if colorlinks=false)
citebordercolor	={0 1 0}	color of frame around citations
urlbordercolor	={0 1 1}	color of frame around URL links

Please note, that explicite RGB specification is only allowed for the border colors (like linkbordercolor etc.), while the others may only assigned to named colors (which you can define your own, see Colors). In order to speed up your customization process, here is a list with the variables with their default value. Copy it in your document and make the changes you want. Next to the variables, there is a short explanations of their meaning:

```
\hypersetup{
    bookmarks=true,         % show bookmarks bar?
    unicode=false,          % non-Latin characters in Acrobat's
bookmarks
    pdftoolbar=true,        % show Acrobat's toolbar?
    pdfmenubar=true,        % show Acrobat's menu?
    pdffitwindow=false,     % window fit to page when opened
    pdfstartview={FitH},    % fits the width of the page to the window
    pdftitle={My title},    % title
    pdfauthor={Author},     % author
    pdfsubject={Subject},   % subject of the document
    pdfcreator={Creator},   % creator of the document
    pdfproducer={Producer}, % producer of the document
    pdfkeywords={keyword1} {key2} {key3}, % list of keywords
    pdfnewwindow=true,      % links in new window
    colorlinks=false,       % false: boxed links; true: colored links
    linkcolor=red,          % color of internal links
    citecolor=green,        % color of links to bibliography
    filecolor=magenta,      % color of file links
    urlcolor=cyan           % color of external links
}
```

If you don't need such a high customization, here are some smaller but useful examples. When creating PDFs destined for printing, colored links are not a good thing as they end up in gray in the final output, making it difficult to read. You can use color frames, which are not printed:

```
 \usepackage{hyperref}
 \hypersetup{colorlinks=false}
```

or make links black:

```
\usepackage{hyperref}
\hypersetup{
    colorlinks,%
    citecolor=black,%
    filecolor=black,%
    linkcolor=black,%
    urlcolor=black
}
```

When you just want to provide information for the Document Info section of the PDF file, as well as enabling back references inside bibliography:

```
\usepackage[pdfauthor={Author's name},%
pdftitle={Document Title},%
pagebackref=true,%
pdftex]{hyperref}
```

By default, URLs are printed using mono-spaced fonts. If you don't like it and you want them to be printed with the same style of the rest of the text, you can use this:

```
\urlstyle{same}
```

Problems with Links and Equations

Messages like the following

```
! pdfTeX warning (ext4): destination with the same identifier (name{
equation.1.7.7.30}) has been already used, duplicate ignored
```

appear, when you have made something like

```
\begin{eqnarray}a=b\nonumber\end{eqnarray}
```

The error disappears, if you use instead this form:

```
\begin{eqnarray*}a=b\end{eqnarray*}
```

Beware that the shown line number is often completely different from the erroneous line.

Problems with Links and Pages

Messages like the following:

```
! pdfTeX warning (ext4): destination with the same
identifier (name{page.1}) has been already used,
duplicate ignored
```

appear when a counter gets reinitialized, for example by using the command `\mainmatter` provided by the book document class. It resets the page number counter to 1 prior to the first chapter of the book. But as the preface of the book also has a page number 1 all links to "page 1" would not be unique anymore, hence the notice that "duplicate has been ignored." The counter measure consists of putting `plainpages=false` into the `hyperref` options. This unfortunately only helps with the page counter. An even more radical solution is to use the option `hypertexnames=false`, but this will cause the page links in the index to stop working.

The best solution is to give each page a unique name by using the `\pagenumbering` command:

```
\pagenumbering{alph}    % a, b, c, ...
... titlepage, other front matter ...
\pagenumbering{roman}   % i, ii, iii, iv, ...
... table of contents, table of figures, ...
\pagenumbering{arabic}  % 1, 2, 3, 4, ...
... beginning of the main matter (chapter 1) ...
```

Another solution is to use `\pagenumbering{alph}` before the command `\maketitle`, which will give the title page the label page.a. Since the page number is suppressed, it won't make a difference to the output.

By changing the page numbering every time before the counter is reset, each page gets a unique name. In this case, the pages would be numbered a, b, c, i, ii, iii, iv, v, 1, 2, 3, 4, 5, ...

If you don't want the page numbers to be visible (for example, during the front matter part), use `\pagestyle{empty}` ... `\pagestyle{plain}`. The important point is that although the numbers are not visible, each page will have a unique name.

Another more flexible approach is to set the counter to something negative:

```
\setcounter{page}{-100}
... titlepage, other front matter ...
```

```
\pagenumbering{roman}   % i, ii, iii, iv, ...
... table of contents, table of figures, ...
\pagenumbering{arabic}  % 1, 2, 3, 4, ...
... beginning of the main matter (chapter 1) ...
```

which will give the first pages a unique negative number.

The problem can also occur with the `algorithms` package: because each algorithm uses the same line-numbering scheme, the line identifiers for the second and follow-on algorithms will be duplicates of the first.

The problem occurs with equation identifiers if you use `\nonumber` on every line of an eqnarray environment. In this case, use the *'ed form instead, e.g. `\begin{eqnarray*} ... \end{eqnarray*}` (which is an unnumbered equation array), and remove the now unnecessary `\nonumber` commands.

If your url's are too long and running off of the page, try using the breakurl package to split the url over multiple lines. This is especially importiant in a multicolumn environment where the line with is greatly shortened.

Problems with Bookmarks

The text displayed by bookmarks does not always look like you expect it to look. Because bookmarks are "just text", much fewer characters are available for bookmarks than for normal LaTeX text. Hyperref will normally notice such problems and put up a warning:

```
Package hyperref Warning:
Token not allowed in a PDFDocEncoded string:
```

You can now work around this problem by providing a text string for the bookmarks, which replaces the offending text:

```
\texorpdfstring{''TEX text''}{''Bookmark Text''}
```

Math expressions are a prime candidate for this kind of problem:

```
\section{\texorpdfstring{$E=mc^2$}{E=mc2}}
```

which turns `\section{$E=mc^2$}` to `E=mc2` in the bookmark area. Color changes also do not travel well into bookmarks:

```
\section{\textcolor{red}{Red !}}
```

produces the string "redRed!". The command `\textcolor` gets ignored but its argument (red) gets printed. If you use:

```
\section{\texorpdfstring{\textcolor{red}{Red !}}{Red\ !}}
```

the result will be much more legible.

If you write your document in unicode and use the `unicode` option for the `hyperref` package you can use unicode characters in bookmarks. This will give you a much larger selection of characters to pick from when using `\texorpdfstring` .

Problems with tables and figures

The links created by `hyperref` point to the label created within the float environment, which, as previously described, must always be set after the caption. Since the caption is usually below a figure or table, the figure or table itself will not be visible upon clicking the link[3] . A workaround exists by using the package `hypcap` [2] with:

```
\usepackage[all]{hypcap}
```

Be sure to call this package *after* loading `hyperref`, which should otherwise be loaded last.

If you use the `wrapfig` package mentioned in the "Wrapping text around figures" section of the "Floats, Figures and Captions" chapter, or other similar packages that define their own environments, you will need to manually include **`\capstart`** in those environments, e.g.:

```
\begin{wrapfigure}{R}{0.5\textwidth}
  \capstart
  \begin{center}
    \includegraphics[width=0.48\textwidth]{filename}
  \end{center}
  \caption{\label{labelname}a figure}
\end{wrapfigure}
```

Problems with long caption and \listoffigures or long title

There is an issue when using **`\listoffigures`** with `hyperref` for long captions or long titles. This happens when the captions (or the titles) are longer than the page width (about 7-9 words depending on your settings). To fix this, you need to use the option breaklinks when first declaring:

```
\usepackage[breaklinks]{hyperref}
```

This will then cause the links in the **`\listoffigures`** to word wrap properly.

Problems with already existing .toc, .lof and similar files

The format of some of the auxilliary files generated by latex changes when you include the `hyperref` package. One can therefore encounter errors like `! Argument of \Hy@setref@link has an extra }.` when the document is typeset with `hyperref` for the first time and these files already exist. The solution to the problem is to delete all the files that latex uses to get references right and typeset again.

References

[1] "Email link with hyperref, url packages" (http://groups.google.com/group/comp.text.tex/browse_thread/thread/ae160fd2fc5680a5/71a5a7c7bfceb3cb?lnk=gst&q=email+url+hyperref#71a5a7c7bfceb3cb). *comp.text.tex User Group*. . Retrieved 2008.

[2] `Other possible values are defined in the hyperref manual` (http://mirror.switch.ch/ftp/mirror/tex/macros/latex/contrib/hyperref/doc/manual.html#TBL-7-40-1)

[3] http://www.ctan.org/tex-archive/macros/latex/contrib/hyperref/README

Packages

Add-on features for LaTeX are known as packages. Dozens of these are pre-installed with LaTeX and can be used in your documents immediately. They should all be stored in subdirectories of `texmf/tex/latex` named after each package. To find out what other packages are available and what they do, you should use the CTAN search page [1] which includes a link to Graham Williams' comprehensive package catalogue. A package is a file or collection of files containing extra LaTeX commands and programming which add new styling features or modify those already existing. Installed package files all end with `.sty` (there may be ancillary files as well). When you try to typeset a document which requires a package which is not installed on your system, LaTeX will warn you with an error message that it is missing, and you can then download the package and install it using the instructions in the installing extra packages section. You can also download updates to packages you already have (both the ones that were installed along with your version of LaTeX as well as ones you added). There is no limit to the number of packages you can have installed on your computer (apart from disk space!), but there is probably a physical limit to the number that can be used inside any one LaTeX document at the same time, although it depends on how big each package is. In practice there is no problem in having even a couple of dozen packages active.

Using an existing package

To use a package already installed on your system, insert a `\usepackage` command in your document preamble with the package name in curly braces:

```
\usepackage{package_name}
```

For example, to use the `color` package, which lets you typeset in colors, you would type:

```
\documentclass[11pt,a4paper,oneside]{report}

\usepackage{color}

\begin{document}
...
\end{document}
```

You can include several package names in one \usepackage command by separating the names with commas, like this:

```
\usepackage{package1,package2,package3}
```

and you can have more than one \usepackage command. Some packages allow optional settings in square brackets. If you use these, you must give the package its own separate \usepackage command, like geometry shown below:

```
\documentclass[11pt,a4paper,oneside]{report}

\usepackage{pslatex,palatino,avant,graphicx,color}
\usepackage[margin=2cm]{geometry}

\begin{document}
\title{\color{red}Practical Typesetting}
\author{\color{blue}Name\\ Work}
\date{\color{green}December 2005}
\maketitle
```

```
\end{document}
```

Many packages can have additional formatting specifications in optional arguments in square brackets, in the same way as `geometry` does. Read the documentation for the package concerned to find out what can be done. You can pass several options together separated by a comma:

```
\usepackage[option1,option2,option3]{''package_name''}
```

Package documentation

To find out what commands a package provides (and thus how to use it), you need to read the documentation. In the `texmf/doc` subdirectory of your installation there should be directories full of .dvi files, one for every package installed. This location is distribution-specific, but is *typically* found in:

Distribution	Path
MiKTeX	`C:\Program Files\MiKTeX 2.7\doc\latex`
teTeX	`/usr/share/texmf-tetex/doc/latex`

Generally, *most* of the packages are in the `latex` subdirectory, although other packages (such as BibTeX and font packages) are found in other subdirectories in `doc`. The documentation directories have the same name of the package (e.g. `amsmath`), which generally have one or more relevant documents in a variety of formats (`dvi`, `txt`, `pdf`, etc.). The documents generally have the same name as the package, but there are exceptions (for example, the documentation for `amsmath` is found at `latex/amsmath/amsdoc.dvi`). If your installation procedure has not installed the documentation, the DVI files can all be downloaded from CTAN. Before using a package, you should read the documentation carefully, especially the subsection usually called "User Interface", which describes the commands the package makes available. You cannot just guess and hope it will work: you have to read it and find out.

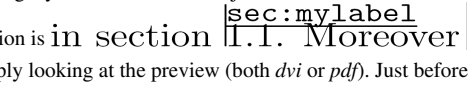
Packages list

Here is a (not complete) list of useful packages that can be used for a wide range of different kind of documents. Each package has a short description next to it and, when available, there is a link to a section describing such package in detail. All of them (unless stated) should be included in your LaTeX distribution as *package_name.sty*. For more information, refer to the documentation of the single packages, as described in the previous section. The list is in alphabetical order.

amsmath	it contains the advanced math extensions for LaTeX. The complete documentation should be in your LaTeX distribution; the file is called *amsdoc*, and can be *dvi* or *pdf*. For more information, see the chapter about Mathematics
amssymb	it adds new symbols in to be used in math mode.
amsthm	it introduces the `proof` environment and the `theoremstyle` command. For more information see the Theorems section.
array	it extends the possibility of LaTeX to handle tables, fixing some bugs and adding new features. Using it, you can create very complicated and customized tables. For more information, see the Tables section.
babel	it provides the internationalization of LaTeX. It has to be loaded in any document, and you have to give as an option the main language you are going to use in the document. For more information see the Internationalization section.
bm	allows use of bold greek letters in math mode using the `\bm{...}` command. This supersedes the `amsbsy` package.
boxedminipage	it introduces the `boxedminipage` environment, that works exactly like `minipage` but adds a frame around it
caption	allows customization of appearance and placement of captions for figures, tables, etc.

cancel	provides commands for striking out mathematical expressions. The syntax is
	`\cancel{x}`
	or
	`\cancelto{0}{x}`

changepage	to easily change the margins of pages. The syntax is
	`\changepage{textheight}{textwidth}%`
	`  {evensidemargin}{oddsidemargin}%`
	`  {columnsep}{topmargin}%`
	`  {headheight}{headsep}%`
	`  {footskip}`
	All the arguments can be both positive and negative numbers; they will be added (keeping the sign) to the relative variable.

cite	assists in citation management
color	it adds support for colored text. For more information, see the relevant section
easylist	adds support for arbitrarily-deep nested lists (useful for outlines)
esint	adds additional integral symbols, for integrals over squares, clockwise integrals over sets, etc.
eucal	other mathematical symbols
fancyhdr	to change header and footer of any page of the document. It is described in the Page Layout section
fontenc	to choose the font encoding of the output text. You might need it if you are writing documents in a language other than English. Check in the Internationalization section.
geometry	for easy management of document margins and the document page size
glossaries	for creation of glossaries and list of acronyms. For more information, see relevant chapter.
graphicx	to manage external pictures
hyperref	it gives LaTeX the possibility to manage links within the document or to any URL when you compile in PDF. For more information, see the relevant section
indentfirst	once loaded, the beginning of any chapter/section is indented by the usual paragraph indentation.
inputenc	to choose the encoding of the input text. You might need it if you are writing documents in a language other than English. Check in the Internationalization section.
latexsym	other mathematical symbols
listings	to insert programming code within the document. Many languages are supported and the output can be customized. For more information, see the relevant section
mathrsfs	other mathematical symbols
natbib	gives additional citation options and styles
pdfpages	This package simplifies the insertion of external multi-page PDF or PS documents.
rotating	It lets you rotate any kind of object. It is particularly useful for rotating tables. For more information, see the relevant section
setspace	Lets you change line spacing, e.g. provides the `\doublespacing` command for making double spaced documents. For more information, see the relevant section
showkeys	it is very useful while writing any document. If you want to reference an image or a formula, you have to give it a name using `\label{...}` and then you can recall it using `\ref{...}`. When you compile the document these will be replaced only with numbers, and you can't know which label you had used unless you take a look at the source. If you have loaded the `showkeys` package, you will see the label just next or above the relevant number in the compiled version. An example of a reference to a section is  . This way you can easily keep track of the labels you add or use, simply looking at the preview (both *dvi* or *pdf*). Just before the final version, remove it

showidx	it prints out all index entries in the left margin of the text. This is quite useful for proofreading a document and verifying the index. For more information, see the Indexing section.
subfiles	the "root" and "child" document can be compiled at the same time without making changes to the "child" document. For more information, see the Subfile package section.
subfig	it allows to define multiple floats (figures, tables) within one environment giving individual captions and labels in the form 1a, 1b.

syntonly	if you add the following code in your preamble:
	`\usepackage{syntonly}`
	`\syntaxonly`
	LaTeX skims through your document only checking for proper syntax and usage of the commands, but doesn't produce any (DVI or PDF) output. As LaTeX runs faster in this mode you may save yourself valuable time. If you want to get the output, you can simply comment out the second line.

textcomp	provides extra symbols, e.g. arrows like `\textrightarrow`, various currencies (`\texteuro`,...), things like `\textcelsius` and many other
theorem	you can change the style of newly defined theorems. For more information see the Theorems section.
todonotes	lets you insert notes of stuff to do with the syntax `\todo{Add details.}`
siunitx	helps you typeset of SI-units correctly. For example `\SI{12}{\mega\herz}`. Automatically handles the correct spacing between the number and the unit. Note that even non-SI-units are set, like dB, rad, ...
ulem	it allows to underline text (either with straight or wavy line). Few examples of usage are added to the Formatting chapter.
url	it defines the `\url{...}` command. URLs often contain special character such as _ and &, in order to write them you should *escape* them inserting a backslash, but if you write them as an argument of `\url{...}`, you don't need to escape any special character and it will take care of proper formatting for you. If you are using the `hyperref`, you don't need to load `url` because it already provides the `\url{...}` command.
verbatim	it improves the `verbatim` environment, fixing some bugs. Moreover, it provides the `comment` environment, that lets you add multiple-line comments or comment out easily big parts of the code.
wrapfig	to insert images surrounded by text. It was discussed in section Floats, Figures and Captions
xypic	is used to create trees, graphs, (commutative) diagrams, and similar things.

Creating packages

See

- LaTeX/Customizing LaTeX#Creating your own package
- LaTeX/Advanced Topics#Creating your own package

External resources

The best way to look for LaTeX packages is the already mentioned CTAN: Search [1]. Additional resources form The TeX Catalogue Online [2]:

- Alphabetic catalogue [3]
- With brief descriptions [4]
- Topical catalogue [5] with packages sorted systematically
- Hierarchical [6] mirroring the CTAN folder hierarchy

References

[1] http://tug.ctan.org/search.html
[2] http://www.ctan.org/tex-archive/help/Catalogue/catalogue.html
[3] http://www.ctan.org/tex-archive/help/Catalogue/alpha.html
[4] http://www.ctan.org/tex-archive/help/Catalogue/brief.html
[5] http://www.ctan.org/tex-archive/help/Catalogue/bytopic.html
[6] http://www.ctan.org/tex-archive/help/Catalogue/hier.html

Advanced Topics

General Guidelines

During this guide we have seen what it is possible to do and how this can be achieved, but the question is: I want to write a proper text with LaTeX, what to do then? Where should I start from? This is a short step-by-step guide about how to start a document properly, keeping a good high-level structure. This way it will be very easy to make modifications even when the document is almost finished. These are all just suggestions, but you might take inspiration from that to create your own document.

Project structure

Create a clear structure of the whole project this way:

1. create a directory only for the project. We'll refer to that in the following parts as the *root directory*
2. create two other directories inside the root, one for LaTeX documents, the other one for images. Since you'll have to write their name quite often, choose short names. A suggestion would be simply *tex* and *img*.
3. create your document (we'll call it document.tex, but you can use the name you prefer) and your own package (for example *mystyle.sty*); this second file will help you to keep the code cleaner.

If you followed all those steps, these files should be in your root directory, using "/" for each directory:

```
./document.tex
./mystyle.sty
./tex/
./img/
```

nothing else.

The file `mystyle.sty`

Instead of putting all the packages you need at the beginning of your document as you could, the best way is to load all the packages you need inside another dummy package called *mystyle* you will create just for your document. The good point of doing this is that you will just have to add one single `\usepackage` in your document, keeping your code much cleaner. Moreover, all the info about your style will be within one file, so when you will start another document you'll just have to copy that file and include it properly, so you'll have exactly the same style you have used.

Creating your own style is very simple: create a file called `mystyle.sty` (you could name it as you wish, but it has to end with ".sty"). Write at the beginning:

```
\ProvidesPackage{mystyle}
```

Then add all the packages you want with the standard command `\usepackage{...}` as you would do normally, change the value of all the variables you want, etc. It will work like the code you put here would be copied and pasted within your document.

For a list of several packages you can use, see the List of Packages section.

The main document `document.tex`

Then create a file called `document.tex`; this will be the main file, the one you will compile, even if you shouldn't need to edit it very often because you will be working on other files. It should be looking like this (it's the sample code for a *report*, but you might easily change it for an *article* or whatever else):

```
\documentclass[12pt,a4paper]{report}
\usepackage{graphicx}
\usepackage{ifpdf}
\ifpdf
   % put here packages only for the PDF:
   \DeclareGraphicsExtensions{.pdf,.png,.jpg,.mps}
   \usepackage{hyperref}
\else
   % put here packages only for the DVI:
\fi

% put all the other packages here:

\usepackage{mystyle}

\begin{document}

\input{./tex/title.tex}
%\maketitle
\tableofcontents
\listoffigures
\listoftables

\input{./tex/intro.tex}
\input{./tex/main_part.tex}
\input{./tex/conclusions.tex}

\appendix
\input{./tex/myappendix.tex}


% Bibliography:
\clearpage
\addcontentsline{toc}{chapter}{Bibliography}
\input{./tex/mybibliography.tex}

\end{document}
```

Here a lot of code expressed in previous sections has been used. At the beginning there is the header discussed in the Tips & Tricks section, so you will be able to compile in both DVI and PDF. Then you import the only package you need, that is your *mystyle.sty* (note that in the code it has to be imported without the extension), then your document starts. Then it inserts the title: we don't like the output of `\maketitle` so we created our own, the code for it will be in a file called `title.tex` in the folder called `tex` we created before. How to write it is explained in the Title

Creation section. Then tables of contents, figure and tables are inserted. If you don't want them, just comment out those lines. Then the main part of the document in inserted. As you can see, there is no text in `document.tex`: everything is in other files in the `tex` directory so that you can easily edit them. We are separating our text from the structural code, so we are improving the "What You See is What You Mean" nature of LaTeX. Then we can see the appendix and finally the Bibliography. It is in a separated file and it is manually added to the table of contents using a tip suggested in the Tips & Tricks.

Once you created your `document.tex` you won't need to edit it anymore, unless you want to add other files in the `tex` directory, but this is not going to happen very often. Now you can write your document separating it in as many files as you want and adding many pictures without getting confused: thanks to the rigid structure you gave to the project, you will be able to keep track of all your edits clearly.

A suggestion: do not call your files like "chapter_01.tex" or "figure_03.png", i.e. try to avoid using numbers in file-names: if the numbering LaTeX gives them automatically is different from the one you gave (and this will likely happen) you will get really confused. When naming a file, stop for a second, think about a short name that can fully explain what is inside the file without being ambiguous, it will let you save a lot of time as soon as the document gets larger.

Writing your document

While writing, whenever you have to take a decision about formatting, define your own command for it and add it to your `mystyle.sty`:let LaTeX work for you. If you do so, it will be very easy to change it if you change your mind. Here is an example: if you are writing a book about Mathematics and you have to use vectors, you have to decide how they will look. There are several different standards, used in many books. If a is a vector, some people like to add an arrow over it ($\vec{a}$), other people write it underlined ($\underline{a}$); another common version is to write it bold ($\mathbf{a}$). Let us assume you want to write your vectors with an arrow over them; then add the following line in your `mystyle.sty`.

```
\newcommand{\myvec}[1]{\vec{#1}}
```

and write your vectors inside the new `\myvec{...}` command. You can call it as you wish, but you'd better choose a short name because you will probably write it very often. Then, if you change your mind and you want your vectors to look differently you just have to change the definition of your `\myvec{...}`. Use this approach whenever you can: this will save you a lot of time.

Advanced Topics

Here are some topics that are not really necessary to write a proper document, but could help you making your life easier and giving you some details to modify.

Adding your own counters

In LaTeX it is fairly easy to create new counters and even counters that reset automatically when another counter is increased (think subsection in a section for example). With the command

```
\newcounter{NameOfTheNewCounter}
```

you create a new counter that is automatically set to zero. If you want the counter to be reset to zero every time another counter is increased, use:

```
\newcounter{NameOfTheNewCounter}[NameOfTheOtherCounter]
```

To increase the counter, either use

```
\stepcounter{NameOfTheNewCounter}
```

or

```
\refstepcounter{NameOfTheNewCounter} % used for labels and cross
referencing
```

or

```
\addtocounter{NameOfTheNewCounter}{number}
```

here the number can also be negative. For automatic reseting you need to use `\stepcounter`.

To set the counter value explicitly, use

```
\setcounter{NameOfTheNewCounter}{number}
```

The values of the counters can be easily found by, for example:

```
\arabic{NameOfTheNewCounter}
```

Instead of `\arabic` you could also use `\alph`, `\Alph`, `\roman`, or `\Roman`.

Here is an example for recreating something similar to a section and subsection counter that already exist in LaTeX:

```
\newcounter{mysection}
\newcounter{mysubsection}[mysection]
\addtocounter{mysection}{2} % set them to some other numbers than 0
\addtocounter{mysection}{10} % same
%
\arabic{mysection}.\arabic{mysubsection}
bla bla

\stepcounter{mysection}
\arabic{mysection}.\arabic{mysubsection}
bla bla
```

```
\stepcounter{mysubsection}
\arabic{mysection}.\arabic{mysubsection}
bla bla

\addtocounter{mysubsection}{25}
\arabic{mysection}.\arabic{mysubsection}
bla bla and more bla bla
```

Boxes

LaTeX builds up its pages by pushing around boxes. At first, each letter is a little box, which is then glued to other letters to form words. These are again glued to other words, but with special glue, which is elastic so that a series of words can be squeezed or stretched as to exactly fill a line on the page.

Admittedly, this is a very simplistic description of what really happens, but the point is that TeX operates with glue and boxes. Letters are not the only things that can be boxes. One can put virtually everything into a box, including other boxes. Each box will then be handled by LaTeX as if it were a single letter.

The past chapters have already dealt with some boxes, although they weren't described as such. The tabular environment and the `\includegraphics`, for example, both produce a box. This means that one can easily arrange two tables or images side by side. You just have to make sure that their combined width is not larger than the `\textwidth`.

You can also pack a paragraph of your choice into a box with either the

```
\parbox[pos]{width}{text}
```

command or the

```
\begin{minipage}[pos]{width} text \end{minipage}
```

environment. The *pos* parameter can take one of the letters `c`, `t` or `b` to control the vertical alignment of the box, relative to the baseline of the surrounding text. width takes a length argument specifying the width of the box. The main difference between a `minipage` and a `\parbox` is that you cannot use all commands and environments inside a parbox, while almost anything is possible in a minipage.

While `\parbox` packs up a whole paragraph doing line breaking and everything, there is also a class of boxing commands that operates only on horizontally aligned material. We already know one of them; it's called `\mbox`. It simply packs up a series of boxes into another one, and can be used to prevent LaTeX from breaking two words. As you can put boxes inside boxes, these horizontal box packers give you ultimate flexibility.

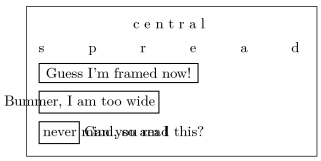
```
\makebox[width][pos]{text}
```

width defines the width of the resulting box as seen from the outside (This means it can be smaller than the material inside the box. You can even set the width to 0pt so that the text inside the box will be typeset without influencing the surrounding boxes). Besides the *length* expressions, you can also use `\width`, `\height`, `\depth`, and `\totalheight` in the width parameter. They are set from values obtained by measuring the typeset text. The *pos* parameter takes a one letter value: **c**enter, flush**l**eft, flush**r**ight, or **s**pread the text to fill the box.

The command `\framebox` works exactly the same as `\makebox`, but it draws a box around the text.

The following example shows you some things you could do with the `\makebox` and `\framebox` commands:

```
\makebox[\textwidth]{%
c e n t r a l}\par
\makebox[\textwidth][s]{%
s p r e a d}\par
\framebox[1.1\width]{Guess I'm
framed now!} \par
\framebox[0.8\width][r]{Bummer,
I am too wide} \par
\framebox[1cm][l]{never
mind, so am I}
Can you read this?
```

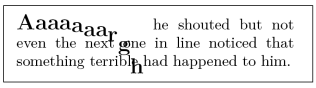


Now that we control the horizontal, the obvious next step is to go for the vertical. No problem for LaTeX. The

```
\raisebox{lift}[extend-above-baseline][extend-below-baseline]{text}
```

command lets you define the vertical properties of a box. You can use `\width`, `\height`, `\depth`, and `\totalheight` in the first three parameters, in order to act upon the size of the box inside the text argument:

```
\raisebox{0pt}[0pt][0pt]{\Large%
\textbf{Aaaa\raisebox{-0.3ex}{a}%
\raisebox{-0.7ex}{aa}%
\raisebox{-1.2ex}{r}%
\raisebox{-2.2ex}{g}%
\raisebox{-4.5ex}{h}}}
he shouted but not even the next
one in line noticed that something
terrible had happened to him.
```



An alternative to these approaches is the usage of the **framed** environment (you will need to include the "framed" package to use it). This provides an easy way to box a paragraph within a document:

```
\begin{framed}
This is an easy way to box text within a document!
\end{framed}
```

Rules and Struts

The `\rule` command in normal use produces a simple black box:

```
\rule[lift]{width}{height}
```

Here is an example:

```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```

This is useful for drawing vertical and horizontal lines.

A special case is a rule with no width but a certain height. In professional typesetting, this is called a *strut*. It is used to guarantee that an element on a page has a certain minimal height. You could use it in a tabular environment to make sure a row has a certain minimum height.

The microtype package can be used to eliminate all hyphenation.

Customizing LaTeX

Documents produced with the commands you have learned up to this point will look acceptable to a large audience. While they are not fancy-looking, they obey all the established rules of good typesetting, which will make them easy to read and pleasant to look at. However, there are situations where LaTeX does not provide a command or environment that matches your needs, or the output produced by some existing command may not meet your requirements.

In this chapter, I will try to give some hints on how to teach LaTeX new tricks and how to make it produce output that looks different from what is provided by default.

New commands

To add your own commands, use the

```
\newcommand{name}[num]{definition}
```

command. Basically, the command requires two arguments: the *name* of the command you want to create, and the *definition* of the command. The *num* argument in square brackets is optional and specifies the number of arguments the new command takes (up to 9 are possible). If missing it defaults to 0, i.e. no argument allowed.

The following two examples should help you to get the idea. The first example defines a new command called `\wbal` that will print "The Wikibook about LaTeX". Such a command could come in handy if you had to write the title of this book over and over again.

```
\newcommand{\wbal}{The Wikibook about \LaTeX}
This is ``\wbal'' \ldots{} ``\wbal''
```

This is “The Wikibook about LaTeX” … “The Wikibook about LaTeX”

The next example illustrates how to define a new command that takes one argument. The `#1` tag gets replaced by the argument you specify. If you wanted to use more than one argument, use `#2` and so on, these arguments are added in an extra set of brackets.

```
\newcommand{\wbalsup}[1] {This is the Wikibook about LaTeX supported by #1}
\newcommand{\wbalTwo}[2] {This is the Wikibook about LaTeX supported by #1 #2}
% in the document body:
\begin{itemize}
\item \wbalsup{Wikimedia}
\item \wbalsup{lots of users!}
\item \wbalTwo{John}{Doe}
\end{itemize}
```

- This is the Wikibook about LaTeX supported by Wikimedia
- This is the Wikibook about LaTeX supported by lots of users!
- This is the Wikibook about LaTeX supported by John Doe

Note: use `\wbalTwo`, not `\wbal2` (error on compiling)

LaTeX will not allow you to create a new command that would overwrite an existing one. But there is a special command in case you explicitly want this: `\renewcommand`. It uses the same syntax as the `\newcommand` command.

In certain cases you might also want to use the `\providecommand` command. It works like `\newcommand`, but if the command is already defined, LaTeX will silently ignore it.

With LaTex2e, it is also possible to add a default parameter to a command with the following syntax:

```
\newcommand{name}[num][default]{definition}
```

If the default parameter of `\newcommand` is present, then the first of the number of arguments specified by `num` is optional with a default value of `default`; if absent, then all of the arguments are required.

```
\newcommand{\wbalTwo}[2][Wikimedia]{This is the Wikibook about LaTeX
supported by {#1} and {#2}!}
% in the document body:
\begin{itemize}
\item \wbalTwo{John Doe}
\item \wbalTwo[lots of users]{John Doe}
\end{itemize}
```

- This is the Wikibook about LaTeX supported by Wikimedia and John Doe!
- This is the Wikibook about LaTeX supported by lots of users and John Doe!

NOTE: when the command is used with an explicit first parameter it is given enclosed with brackets (`"[lots of users]"`).

New Environments

Just as with the `\newcommand` command, there is a command to create your own environments. The `\newenvironment` command uses the following syntax:

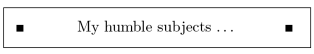
```
\newenvironment{name}[num]{before}{after}
```

Again `\newenvironment` can have an optional argument. The material specified in the *before* argument is processed before the text in the environment gets processed. The material in the *after* argument gets processed when the `\end{name}` command is encountered.

The *num* argument is used the same way as in the `\newcommand` command. LaTeX makes sure that you do not define an environment that already exists. If you ever want to change an existing command, you can use the `\renewenvironment` command. It uses the same syntax as the `\newenvironment` command.

The example below illustrates the usage of the `\newenvironment` command:

```
\newenvironment{king}
{\rule{1ex}{1ex}\hspace{\stretch{1}}}
{\hspace{\stretch{1}}\rule{1ex}{1ex}}

\begin{king}
My humble subjects \ldots
\end{king}
```



Extra space

When creating a new environment you may easily get bitten by extra spaces creeping in, which can potentially have fatal effects. For example when you want to create a title environment which suppresses its own indentation as well as the one on the following paragraph. The `\ignorespaces` command in the begin block of the environment will make it ignore any space after executing the begin block. The end block is a bit more tricky as special processing occurs at the end of an environment. With the `\ignorespacesafterend` LaTeX will issue an `\ignorespaces` after the special 'end' processing has occurred.

```
\newenvironment{simple}%
{\noindent}%
{\par\noindent}

\begin{simple}
See the space\\to the left.
\end{simple}
Same\\here.
```

```
  See the space
to the left.

  Same
here.
```

```
\newenvironment{correct}%
{\noindent\ignorespaces}%
{\par\noindent%
\ignorespacesafterend}

\begin{correct}
No space\\to the left.
\end{correct}
Same\\here.
```

```
No space
to the left.

Same
here.
```

Also, if you're still having problems with extra space being appended at the end of your environment when using the `\input` for external source, make sure there is no space between the beginning, sourcing, and end of the environment, such as:

```
\begin{correct}\input{somefile.tex}\end{correct}
```

Command-line LaTeX

If you work on a Unix-like OS, you might be using Makefiles or any kind of script to build your LaTeX projects. In that connection it might be interesting to produce different versions of the same document by calling LaTeX with command-line parameters. If you add the following structure to your document:

```
\usepackage{ifthen}
\ifthenelse{\equal{\blackandwhite}{true}}{
% "black and white" mode; do something..
}{
% "color" mode; do something different..
}
```

Now you can call LaTeX like this:

```
latex '\newcommand{\blackandwhite}{true}\input{test.tex}'
```

First the command `\blackandwhite` gets defined and then the actual file is read with input. By setting `\blackandwhite` to false the color version of the document would be produced.

Creating your own package

If you define a lot of new environments and commands, the preamble of your document will get quite long. In this situation, it is a good idea to create a LaTeX package containing all your command and environment definitions. You can then use the `\usepackage` command to make the package available in your document. Writing a package basically consists of copying the contents of your document preamble into a separate file with a name ending in `.sty`.

It is very simple, just follow the steps:

1. create a simple text file called *mypack.sty* (or any other name you like) and open it with any text editor
2. at the very beginning of the text document just write

```
\ProvidesPackage{mypack}
```

   note: it has to have the same name of the file without the extension. It tells LaTeX the name of the package and will allow it to issue a sensible error message when you try to include a package twice.
3. write whatever you want in it using all the LaTeX commands you know. Normally you should define new commands or import other packages.
4. import your new package with the known command

```
\usepackage{mypack}
```

5. or

```
\RequirePackage{mypack}
```

the file *mypack.sty* and the LaTeX source you are compiling must be in the same directory. It will be as though all you have written within your package is within the document itself.

Alternatively, it is possible to place the package within *~/texmf/tex/latex/mypack/mypack.sty* where '~' is your home directory (On Windows this is often *C:\Users\username*) and where *mypack* is the name of your package. Running `texhash` or equivalent will allow you to use your package as detailed above, but without it needing to be in the same directory as your document.

Creating your own style

It is also possible to create your own style file. The process is similar to the creation of your own package, you can call your own style file in the preamble of any document by the command:

```
\documentclass{mystyle}
```

The name of the style file is then *mystyle.cls* and can be opened with any text editor. At the beginning of this file, the following line has to be provided:

```
\ProvidesClass{mystyle}
```

again, within the style files other files or packages are imported by the requirepackage command.

Spacing

Line Spacing

If you want to use larger inter-line spacing in a document, you can change its value by putting the

```
\linespread{factor}
```

command into the preamble of your document. Use `\linespread{1.3}` for "one and a half" line spacing, and `\linespread{1.6}` for "double" line spacing. Normally the lines are not spread, so the default line spread factor is 1.

The `setspace` package allows more fine-grained control over line spacing. To set "one and a half" line spacing document-wide, but not where it is usually unnecessary (e.g. footnotes, captions):

```
\usepackage{setspace}
%\singlespacing
\onehalfspacing
%\doublespacing
%\setstretch{1.1}
```

To change line spacing within the document, the `setspace` package provides the environments `singlespace`, `onehalfspace`, `doublespace` and `spacing`:

```
This paragraph has \\ default \\ line spacing.

\begin{doublespace}
  This paragraph has \\ double \\ line spacing.
\end{doublespace}

\begin{spacing}{2.5}
  This paragraph has \\ huge gaps \\ between lines.
\end{spacing}
```

Paragraph formatting

In LaTeX, there are two parameters influencing paragraph layout. By placing a definition like:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

in the preamble of the input file, you can change the layout of paragraphs. These two commands increase the space between two paragraphs while setting the paragraph indent to zero.

The `plus` and `minus` parts of the length above tell TeX that it can compress and expand the inter paragraph skip by the amount specified, if this is necessary to properly fit the paragraphs onto the page. In continental Europe, paragraphs are often separated by some space and not indented. But beware, this also has its effect on the table of contents. Its lines get spaced more loosely now as well. To avoid this, you might want to move the two commands from the preamble into your document to some place below the command `\tableofcontents`. You may want to consider whether or not you want to use paragraph spacing. Most professional books use indenting and not spacing to separate paragraphs.

If you want to indent a paragraph that is not indented, you can use

```
\indent
```

at the beginning of the paragraph. Obviously, this will only have an effect when `\parindent` is not set to zero. If you want to indent the beginning of every section, you can use the `indentfirst` package, see the chapter about LaTeX/Packages for more information.

To create a non-indented paragraph, you can use

```
\noindent
```

as the first command of the paragraph. This might come in handy when you start a document with body text and not with a sectioning command.

Horizontal Space

LaTeX determines the spaces between words and sentences automatically. To add horizontal space, use:

```
\hspace{length}
```

If such a space should be kept even if it falls at the end or the start of a line, use `\hspace*` instead of `\hspace`. The length in the simplest case is just a number plus a unit, e.g. `\hspace{1.5 cm}`. For a list of the possible units, see the Useful Measurement Macros appendix.

The command:

```
\stretch{n}
```

generates a special rubber space. It stretches until all the remaining space on a line is filled up. If two `\hspace{\stretch{n}}` commands are issued on the same line, they grow according to the stretch factor.

```
x\hspace{\stretch{1}}    x     x               x
x\hspace{\stretch{3}} x
```

Vertical Space

The space between paragraphs, sections, subsections, etc. is determined automatically by LaTeX. If you want to customize the default paragraph spacing, it can be achieved with the following command in the preamble of your document:

```
\parskip 7.2pt
```

If necessary, additional vertical space *between two paragraphs* can be added with the command:

```
\vspace{length}
```

This command should normally be used between two empty lines. If the space should be preserved at the top or at the bottom of a page, use the starred version of the command, `\vspace*`, instead of `\vspace`. The `\stretch` command, in connection with `\pagebreak`, can be used to typeset text on the last line of a page, or to center text vertically on a page.

Additional space between two lines of the same paragraph or within a table is specified with the

```
\\[length]
```

command.

If you want to add space at the beginning of the document, without anything else written before, then you may use

```
{ \vspace*{length} }
```

It's important you use the `\vspace*` command instead of `\vspace`, otherwise latex can silently ignore the extra space.

Multiple files

Getting LaTeX to process multiple files

As your work grows, your LaTeX file can become unwieldy and confusing, especially if you are writing a long article with substantial, discrete sections, or a full-length book. In such cases it is good practice to split your work into several files. For example, if you are writing a book, it makes a lot of sense to write each chapter in a separate `.tex` file. LaTeX makes this very easy thanks to two commands:

```
\input{filename}
```

and

```
\include{filename}
```

The differences between these files will be explained below but what they have in common is that they process the contents of `filename.tex` before continuing with the rest of the base file. When the compiler processes your base file and reaches the command `\input` or `\include`, it reads `filename.tex` and processes its content in accordance with the formatting commands specified in the base file. This way you can put all the formatting options in your base file and then `input` or `include` the files which contain the actual content of your work. This means that the important part of your working process, i.e. writing, is kept largely separate from formatting choices (which is one of the main reasons why LaTeX is so good for serious writing!). You will thus be dealing solely with text and very basic commands such as `\section`, `\emph` etc. Your document will be uncluttered and much easier to work with.

The second method of including a file, `\include{filename}`, differs from the first in some important ways. You cannot nest `\include` statements within a file added via `\include`; `\input`, on the other hand, allows you to call files which themselves call other files, ad infinitum (well, nearly!). You can, however, `\include` a file which contains one or more `\input` commands. Please resist the temptation to nest files in this way simply because the system can do it: you will end up with just another kind of complexity!

A further important difference is that using `\include` will force a page break (which makes it ideal for a book's chapters), whereas the `input` command does not (which in turn makes it ideal for, say, a long article with discrete sections, which of course are not normally set on a new page).

Working on discrete parts of your documents has consequences for how the base file is compiled; these will be dealt with below.

Using different paths

When the LaTeX compiler finds a reference to an external file in the base file, it will look for it in the same directory. However, you can in principle refer to any file on your system, using both absolute and relative paths.

An *absolute* path is a full path- and filename with every element specified. So, `filename.tex` might have the full path,

```
\input{/home/user/texfiles/filename.tex}
```

If you had created the directory `myfiles` for your writing project, in your `texfiles` directory, its full path would be,

```
\input{/home/user/texfiles/myfiles/filename.tex}
```

Obviously, using absolute paths is inefficient if you are referring to a file in the current directory. If, however, you need to include a file which is always kept at a specific place in your system, you may refer to it with an absolute path, for example,

```
\input{/home/user/documents/useful/foo.tex}
```

In practice, an absolute file path is generally used when one has to refer to a file which is quite some way away in the file system (or perhaps even on a different server!). One word of warning: do not leave empty spaces in the filenames, they can cause ambiguous behaviour. Either leave no spaces or use underscores _ instead.

You may, however, need to make your source portable (to another computer or to a different location of your harddisk), in which case *relative* paths should be used if you wish to avoid unnecessary re-writing of path names. Or, a relative path may simply be a more efficient and elegant way of referring to a file. A relative path is one which is defined in relation to the current directory, in our case the one which contains the base file. LaTeX uses the standard UNIX notation: with a simple dot **.** you refer to the current directory, and by two dots **..** you refer to the previous directory, that is the one above the current directory in the file system tree. The slash **/** is used to separate the different components of a pathname: directories and filenames. So by **./** you refer to the current directory, by **../** you refer to the previous directory, by **../../** you refer to a directory which is two steps upwards in the filesystem tree. Writing

```
\input{./filename.tex}
```

will have *exactly* the same effect as writing

```
\input{filename.tex}
```

but if you found it more convenient to put all your files in a sub-directory of your current directory, called `myfiles`, you would refer to that file by specifying

```
\input{./myfiles/filename.tex}
```

Indeed, in our example of the absolute path above, you could refer to that file relatively, too:

```
\input{../../documents/useful/foo.tex}
```

Of course, all commonly used file systems – Linux, Mac OS X and Windows – also feature the UNIX **./**, **../** facility outlined above. Do note, however, that LaTeX uses forward slashes **/** even on Microsoft Windows platforms, which use backslashes **** in pathnames. LaTeX implementations for Windows systems perform this conversion for you, which ensures that your document will be valid across all installations.

This flexibility, inherent in the way in which LaTeX is integrated with modern file systems, lets you input files in a way which suits your particular set-up.

Compiling the base file

When you compile your document, page references and the like will change according to your use of the `\input` and `\include` commands. Normally LaTeX users only run the compiler on parts of the document to check that an individual chapter is syntactically correct and looks as the writer intended. A full run is generally only performed for producing a full draft or the final version. In such cases, it is invariably necessary to run LaTeX twice or more to resolve all the page numbers, references, etc. (especially if you are using bibliographic software such as BiBTeX, too).

The simplest way to check that one or more of the various components of your work is syntactically robust, is to comment out the command with a percentage sign, for example:

```
\documentclass{article}
\begin{document}
%\input{Section_1}
%\input{Section_2}
%\input{Section_3}
\input{Section_4}
%\input{Section_5}
\end{document}
```

This code will process your base file with the `article` conventions but only the material in the file `Section_4.tex` will be processed. If that was, say, the last thing you needed to check before sending off to that major journal, you would then simply remove all the percentage signs and re-run LaTeX, repeating the compiling process as necessary to resolve all references, page numbers and so on.

Using \includeonly

Using this command provides more complex, and hence more useful possibilities. If you include the following command in your preamble, i.e. before \begin{document},

```
\includeonly{filename1,filename2,...}
```

only the files specified between the curly braces will be included. Note that you can have one or more files as the argument to this command: separate them with a comma, no spaces.

This requires that there are `\include` commands in the document which specify these files. The filename should be written without the `.tex` file extension:

```
\documentclass{book}
\includeonly{Chapter_1,Chapter_4)
\begin{document}
\include{Chapter_1}
\include{Chapter_2}
\include{Chapter_3}
\include{Chapter_4}
\end{document}
```

This code would process the base file but only include the content of the author's first and fourth chapters (`Chapter_1.tex` and `Chapter_4.tex`). Importantly, this alternative retains as much of the `.aux` information as possible from the previous run, so messes up your cross-references much less than the makeshift suggestion above.

Subfiles package

A disadvantage of using `\input` and `\include` is that only the "root" document can be compiled and not the "child" documents individually. The package subfiles [1] resolves this problem.

In the "root" document the package must be loaded as:

```
\usepackage{subfiles}
```

Instead of using `\input` and `\include`, "child" documents must be loaded as follows:

```
\subfile{filename}
```

The "child" documents must start with the following statements:

```
\documentclass[rootdocument.tex]{subfiles}
\begin{document}
```

and end with:

```
\end{document}
```

In summary, root document (`main.tex`) looks like:

```
\documentclass{book}
\begin{document}
%% my document content
\subfile{chapter1}
%% more of my document content
\end{document}
```

and chapter 1 (`chapter1.tex`) looks like:

```
\documentclass[main.tex]{subfiles}
\begin{document}
%% my chapter 1 content
%%
%% more of my chapter 1 content
\end{document}
```

Some linux distributions, don't have subfiles package in their latex distributions. You can download subfiles.zip [2] to generate `subfiles.cls` and `subfiles.sty` files:

```
wget http://mirror.ctan.org/macros/latex/contrib/subfiles.zip
cd subfiles
latex subfiles.dtx
latex subfiles.ins
```

Inserting PDF files

If you need to insert an existing, possibly multi-page, PDF file into your LaTeX document, whether or not the included PDF was compiled with LaTeX or another tool, consider using the pdfpages package [3]. In the preamble, include the package:

```
\usepackage[final]{pdfpages}
```

This package also allows you to specify which pages you wish to include: for example, to insert pages 3 to 6 from some file `insertme.pdf`, use:

```
\includepdf[pages=3-6]{insertme.pdf}
```

To insert the whole of `insertme.pdf`:

```
\includepdf[pages=-]{insertme.pdf}
```

For full functionality, compile the output with `pdflatex`.

External Links

- Subfiles package documentation [4]
- pdfpages package documentation [5]

References

[1] http://ctan.org/tex-archive/macros/latex/contrib/subfiles
[2] http://tezcatl.fciencias.unam.mx/tex-archive/macros/latex/contrib/subfiles.zip
[3] http://www.ctan.org/tex-archive/macros/latex/contrib/pdfpages/
[4] http://tug.ctan.org/tex-archive/macros/latex/contrib/subfiles/subfiles.pdf
[5] http://mirror.ctan.org/macros/latex/contrib/pdfpages/pdfpages.pdf

Collaborative Writing of LaTeX Documents

Note: This Wikibook is based on the article *Tools for Collaborative Writing of Scientific LaTeX Documents* by Arne Henningsen that is published in *The PracTeX Journal* 2007, number 3 (http://www.tug.org/pracjourn/).

Abstract

Collaborative writing of documents requires a strong synchronisation among authors. This Wikibook describes a possible way to organise the collaborative preparation of LaTeX documents. The presented solution is primarily based on the version control system *Subversion* (http://subversion.apache.org/). The Wikibook describes how *Subversion* can be used together with several other software tools and LaTeX packages to organise the collaborative preparation of LaTeX documents.

Other Methods

- The online LaTeX editor ScribTeX [1] makes sharing your document with others very easy. It provides a full LaTeX environment, with all the usual features of LaTeX like bibtex, images and custom style files. It also provides full version histories of your files, essential for collaborating. The free account allows only 3 projects and only one collaborator per project.
- publications.li [2] is a real-time collaborative LaTeX editor.
- Verbosus [3] is a professional Online LaTeX Editor that supports collaboration with other users and is free to use. Merge conflicts can easily resolved by using a built-in merge tool that uses an implementation of the diff-algorithm to generate information required for a successful merge.
- The Monkey TeX [4] is free and allows team sharing.
- Another option for collaboration is dropbox [5]. It has 2Gb free storage and versioning system. Works like SVN, but more automated and therefore especially useful for beginning latex users.
- As the LaTeX system uses plain text, you can use synchronous collaborative editors like Gobby. In Gobby you can write your documents in collaboration with anyone in real time. It is strongly recommended that you use utf8 encoding (especially if there are users on multiple operating systems collaborating) and a stable network (typically wired networks).
- Google Documents [6] or LaTeX Lab [7] also allows real-time simultaneous collaborative editing of text files for anyone with a Google account (and its option to make the document available through a URL makes local download and compilation easily scriptable).
- TitanPad [8] (or other clones [9] of EtherPad). To compile use the command line :
  `wget -O filename.tex "http://titanpad.com/ep/pad/export/xxxx/latest?format=txt" && (latex filename.tex)`
  where 'xxxx' should be replaced by the pad number (something like 'z7rSrfrYcH').
- You could use more modern distributed version control systems like Mercurial or Git.

Introduction

The collaborative preparation of documents requires a considerable amount of coordination among the authors. This coordination can be organised in many different ways, where the best way depends on the specific circumstances.

In this Wikibook, I describe how the collaborative writing of LaTeX documents is organised at our department (Division of Agricultural Policy, Department of Agricultural Economics, University of Kiel, Germany). I present our software tools, and describe how we use them. Thus, this Wikibook provides some ideas and hints that will be useful for other LaTeX users who prepare documents together with their co-authors.

Interchanging Documents

There are many ways to interchange documents among authors. One possibility is to compose documents by interchanging e-mail messages. This method has the advantage that common users generally do not have to install and learn the usage of any extra software, because virtually all authors have an e-mail account. Furthermore, the author who has modified the document can easily attach the document and explain the changes by e-mail as well. Unfortunately, there is a problem when two or more authors are working at the same time on the same document. So, how can authors synchronise these files?

A second possibility is to provide the document on a common file server, which is available in most departments. The risk of overwriting each others' modifications can be eliminated by locking files that are currently edited. However, generally the file server can be only accessed from within a department. Hence, authors who are out of the building cannot use this method to update/commit their changes. In this case, they will have to use another way to overcome this problem. So, how can authors access these files?

A third possibility is to use a version control system. A comprehensive list of version control systems can be found at Wikipedia [10]. Version control systems keep track of all changes in files in a project. If many authors modify a document at the same time, the version control system tries to merge all modifications automatically. However, if multiple authors have modified the same line, the modifications cannot be merged automatically, and the user has to resolve the conflict by deciding manually which of the changes should be kept. Authors can also comment their modifications so that the co-authors can easily understand the workflow of this file. As version control systems generally communicate over the internet (e.g. through TCP/IP connections), they can be used from different computers with internet connections. A restrictive firewall policy might prevent the version control system from connecting to the internet. In this case, the network administrator has to be asked to open the appropriate port. The internet is only used for synchronising the files. Hence, a permanent internet connection is not required. The only drawback of a version control system could be that it has to be installed and configured.

Moreover, a version control system is useful even if a single user is working on a project. First, the user can track (and possibly revoke) all previous modifications. Second, this is a convenient way to have a backup of the files on other computers (e.g. on the version control server). Third, this allows the user to easily switch between different computers (e.g. office, laptop, home).

The Version Control System *Subversion*

Subversion (SVN) [11] comes as a successor to the popular version control system CVS. SVN operates on a client-server model in which a central server hosts a project repository that users copy and modify locally. A repository functions similarly to a library in that it permits users to check out the current project, make changes, and then check it back in. The server records all changes a user checks in (usually with a message summarizing what changes the user made) so that other users can easily apply those changes to their own local files.

Each user has a local *working copy* of a remote *repository*. For instance, users can *update* changes from the repository to their working copy, *commit* changes from their own working copy to the repository, or (re)view the differences between working copy and repository.

To set up a SVN version control system, the SVN server software has to be installed on a (single) computer with permanent internet access. (If this computer has no static IP address, one can use a service like DynDNS [12] to be able to access the server with a static hostname.) It can run on many Unix, modern MS Windows, and Mac OS X platforms.

Users do not have to install the SVN server software, but a SVN "client" software. This is the unique way to access the repositories on the server. Besides the basic SVN command-line client, there are several Graphical User Interface Tools (GUIs) and plug-ins for accessing the SVN server (see http:/ / subversion. tigris. org/ links. html). Additionally, there are very good manuals about SVN freely available on the internet (e.g. http://svnbook.red-bean.

com).

At our department, we run the SVN server on a GNU-Linux system, because most Linux distributions include it. In this sense, installing, configuring, and maintaining SVN is a very simple task.

Most MS Windows users access the SVN server by the TortoiseSVN [13] client, because it provides the most usual interface for common users. Linux users usually use SVN utilities from the command-line, or eSvn [14]--a GUI frontend--with KDiff3 [15] for showing complex differences.

Hosting LaTeX files in *Subversion*

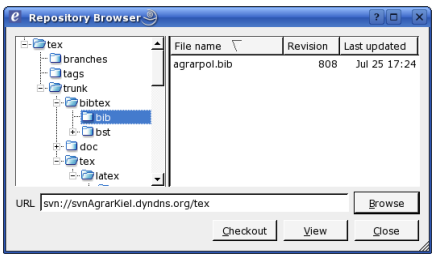


Figure 1: Common `texmf` tree shown in *eSvn*'s Repository Browser

On our *Subversion* server, we have one repository for a common `texmf` tree. Its structure complies with the **TeX Directory Structure** guidelines (TDS, http://www.tug.org/tds/tds.html, see figure 1). This repository provides LaTeX classes, LaTeX styles, and BibTeX styles that are not available in the LaTeX distributions of the users, e.g. because they were bought or developed for the internal use at our department. All users have a working copy of this repository and have configured LaTeX to use this as their personal `texmf` tree. For instance, teTeX (http://www.tug.org/tetex/) users can edit their TeX configuration file (e.g. `/etc/texmf/web2c/texmf.cnf`) and set the variable `TEXMFHOME` to the path of the working copy of the common `texmf` tree (e.g. by `TEXMF'HOME = $HOME/texmf`); MiKTeX (http://www.miktex.org/) users can add the path of the working copy of the common `texmf` tree in the 'Roots' tab of the MiKTeX Options.

If a new class or style file has been added (but not if these files have been modified), the users have to update their 'file name data base' (FNDB) before they can use these classes and styles. For instance, teTeX users have to execute `texhash`; MiKTeX users have to click on the button 'Refresh FNDB' in the 'General' tab of the MiKTeX Options.

Furthermore, the repository contains manuals explaining the specific LaTeX software solution at our department (e.g. this document).

The *Subversion* server hosts a separate repository for each project of our department. Although branching, merging, and tagging is less important for writing text documents than for writing source code for software, our repository layouts follow the recommendations of the 'Subversion book' (http://svnbook.red-bean.com). In this sense, each repository has the three directories `/trunk`, `/branches`, and `/tags`.

The most important directory is `/trunk`. If a single text document belongs to the project, all files and subdirectories of this text document are in `/trunk`. If the project yields two or more different text documents, `/trunk` contains a subdirectory for each text document. A slightly different version (a **branch**) of a text document (e.g. for presentation at a conference) can be prepared either in an additional subdirectory of `/trunk` or in a new subdirectory of `/branches`. When a text document is submitted to a journal or a conference, we create a **tag** in the directory `/tags` so that it is easy to identify the submitted version of the document at a later date. This feature has been proven very useful. When creating branches and tags, it is important always to use the *Subversion* client (and not the tools of the local file system) for these actions, because this saves disk space on the server and it preserves information about the same history of these documents.

Often the question arises, which files should be put under version control. Generally, all files that are directly modified by the user and that are necessary for compiling the document should be included in the version control system. Typically, these are the LaTeX source code (`*.tex`) files (the main document and possibly some subdocuments) and all pictures that are inserted in the document (`*.eps`, `*.jpg`, `*.png`, and `*.pdf` files). All LaTeX classes (`*.cls`), LaTeX styles (`*.sty`), BibTeX data bases (`*.bib`), and BibTeX styles (`*.bst`) generally should be hosted in the repository of the common `texmf` tree, but they could be included in the

respective repository, if some (external) co-authors do not have access to the common `texmf` tree. On the other hand, all files that are automatically created or modified during the compilation process (e.g. `*.aut`, `*.aux`, `*.bbl`, `*.bix`, `*.blg`, `*.dvi`, `*.glo`, `*.gls`, `*.idx`, `*.ilg`, `*.ind`, `*.ist`, `*.lof`, `*.log`, `*.lot`, `*.nav`, `*.nlo`, `*.out`, `*.pdf`, `*.ps`, `*.snm`, and `*.toc` files) or by the (LaTeX or BibTeX) editor (e.g. `*.bak`, `*.bib~`, `*.kilepr`, `*.prj`, `*.sav`, `*.tcp`, `*.tmp`, `*.tps`, and `*.tex~` files) generally should be **not** under version control, because these files are not necessary for compilation and generally do not include additional information. Furthermore, these files are regularly modified so that conflicts are very likely.

Subversion really makes the difference

A great feature of a version control system is that all authors can easily trace the workflow of a project by viewing the differences between arbitrary versions of the files. Authors are primarily interested in 'effective' modifications of the source code that change the compiled document, but not in 'ineffective' modifications that have no impact on the compiled document (e.g. the position of line breaks). Software tools for comparing text documents ('diff tools') generally cannot differentiate between 'effective' and 'ineffective' modifications; they highlight both types of modifications. This considerably increases the effort to find and review the 'effective' modifications. Therefore, 'ineffective' modifications should be avoided.

In this sense, it is very important not to change the positions of line breaks without cause. Hence, automatic line wrapping of the users' LaTeX editors should be turned off and line breaks should be added manually. Otherwise, if a single word in the beginning of a paragraph is added or removed, all line breaks of this paragraph might change so that most diff tools indicate the entire paragraph as modified, because they compare the files line by line. The diff tools *wdiff* (http://www.gnu.org/software/wdiff/) and *dwdiff* (http://os.ghalkes.nl/dwdiff.html) are not affected by the positions of line breaks, because they compare documents word by word. However, their output is less clear so that modifications are more difficult to track. Moreover, these tools cannot be used directly with the *Subversion* command-line switch `--diff-cmd`, but a small wrapper script has to be used (http://textsnippets.com/posts/show/1033).

A reasonable convention is to add a line break after each sentence and start each new sentence in a new line. Note that this has an advantage also beyond version control: if you want to find a sentence in your LaTeX code that you have seen in a compiled (DVI, PS, or PDF) file or on a printout, you can easily identify the first few words of this sentence and screen for these words on the left border of your editor window.

Furthermore, we split long sentences into several lines so that each line has at most 80 characters, because it is rather inconvenient to search for (small) differences in long lines. (Note: For instance, the LaTeX editor *Kile* (http://kile.sourceforge.net/) can assist the user in this task when it is configured to add a vertical line that marks the 80th column.) We find it very useful to introduce the additional line breaks at logical breaks of the sentence, e.g. before a relative clause or a new part of the sentence starts. An example LaTeX code that is formatted according to these guidelines is the source code of the article *Tools for Collaborative Writing of Scientific LaTeX Documents* by Arne Henningsen that is published (including the source code) in *The PracTeX Journal* 2007, Number 3 (http://www.tug.org/pracjourn/2007-3/henningsen/).

If the authors work on different operating systems, their LaTeX editors will probably save the files with different newline (end-of-line) characters (http://en.wikipedia.org/wiki/Newline). To avoid this type of 'ineffective' modifications, all users can agree on a specific newline character and configure their editor to use this newline character. Another alternative is to add the subversion property 'svn:eol-style' and set it to 'native'. In this case, *Subversion* automatically converts all newline characters of this file to the native newline character of the author's operating system (http://svnbook.red-bean.com/en/1.4/svn.advanced.props.file-portability.html#svn.advanced.props.special.eol-style).

There is also another important reason for reducing the number of 'ineffective' modifications: if several authors work on the same file, the probability that the same line is modified by two or more authors at the same time increases

with the number of modified lines. Hence, 'ineffective' modifications unnecessarily increase the risk of conflicts (see section Interchanging Documents).

Furthermore, version control systems allow a very effective quality assurance measure: all authors should critically review their own modifications before they commit them to the repository (see figure 2). The differences between the user's working copy and the repository can be easily inspected with a single *Subversion* command or with one or two clicks in a graphical *Subversion* client. Furthermore, authors should verify that their code can be compiled flawlessly before they commit their modifications to the repository. Otherwise, the co-authors have to pay for these mistakes when they want to compile the document. However, this directive is not only reasonable for version control systems but also for all other ways to interchange documents among authors.

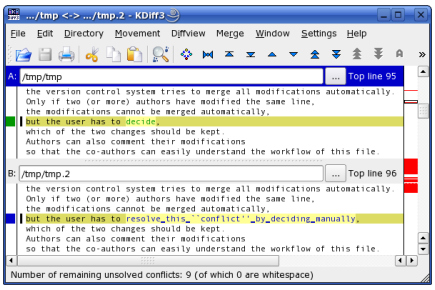


Figure 2: Reviewing modifications in *KDiff3*

Subversion has a feature called 'Keyword Substitution' that includes dynamic version information about a file (e.g. the revision number or the last author) into the contents of the file itself (see e.g. http://svnbook.red-bean.com, chapter 3). Sometimes, it is useful to include these information not only as a comment in the LaTeX source code, but also in the (compiled) DVI, PS, or PDF document. This can be achieved with the LaTeX packages *svn* (http://www.ctan.org/tex-archive/macros/latex/contrib/svn/), *svninfo* (http://www.ctan.org/tex-archive/macros/latex/contrib/svninfo/), or (preferably) *svn-multi* (http://www.ctan.org/tex-archive/macros/latex/contrib/svn-multi/).

The most important directives for collaborative writing of LaTeX documents with version control systems are summarised in the following box.

Directives for using LaTeX with version control systems

1. Avoid 'ineffective' modifications.
2. Do not change line breaks without good reason.
3. Turn off automatic line wrapping of your LaTeX editor.
4. Start each new sentence in a new line.
5. Split long sentences into several lines so that each line has at most 80 characters.
6. Put only those files under version control that are directly modified by the user.
7. Verify that your code can be compiled flawlessly before committing your modifications to the repository.
8. Use *Subversion*'s diff feature to critically review your modifications before committing them to the repository.
9. Add a meaningful and descriptive comment when committing your modifications to the repository.
10. Use the *Subversion* client for copying, moving, or renaming files and folders that are under revision control.

If the users are willing to let go of the built-in *diff* utility of SVN and use *diff* tools that are local on their workstations, they can put to use such tools that are more tailored to text documents. The *diff* tool that comes with SVN was designed with source code in mind. As such, it is built to be more useful for files of short lines. Other tools, such as **Compare It!** allows to conveniently compare text files where each line can span hundreds of characters (such as when each line represents a paragraph). When using a *diff* tool that allows convenient views of files with long lines, the users can author the TeX files without a strict line-breaking policy.

Managing collaborative bibliographies

Writing of scientific articles, reports, and books requires the citation of all relevant sources. BibTeX is an excellent tool for citing references and creating bibliographies (Markey 2005, Fenn 2006). Many different BibTeX styles can be found on CTAN (http://www.ctan.org) and on the LaTeX Bibliography Styles Database (http://jo.irisson.free.fr/bstdatabase/). If no suitable BibTeX style can be found, most desired styles can be conveniently assembled with *custombib*/*makebst* (http://www.ctan.org/tex-archive/macros/latex/contrib/custom-bib/). Furthermore,

BibTeX style files can be created or modified manually; however this action requires knowledge of the (unnamed) postfix stack language that is used in BibTeX style files (Patashnik 1988).

At our department, we have a common bibliographic data base in the BibTeX format (.bib file). It resides in our common `texmf` tree (see section 'Hosting LaTeX files in *Subversion*') in the subdirectory `/bibtex/bib/` (see figure 1). Hence, all users can specify this bibliography by only using the file name (without the full path) --- no matter where the user's working copy of the common `texmf` tree is located.

All users edit our bibliographic data base with the graphical BibTeX editor *JabRef* (http://jabref.sourceforge.net/). As *JabRef* is written in *Java*, it runs on all major operating systems. As different versions of *JabRef* generally save files in a slightly different way (e.g. by introducing line breaks at different positions), all users should use the same (e.g. last stable) version of *JabRef*. Otherwise, there would be many differences between different versions of `.bib` files that solely originate from using different version of *JabRef*. Hence, it would be hard to find the real differences between the compared documents. Furthermore, the probability of conflicts would be much higher (see section 'Subversion really makes the difference'). As *JabRef* saves the BibTeX data base with the native newline character of the author's operating system, it is recommended to add the *Subversion* property 'svn:eol-style' and set it to 'native' (see section 'Subversion really makes the difference').

JabRef is highly flexible and can be configured in many details. We make the following changes to the default configuration of *JabRef* to simplify our work. First, we specify the default pattern for BibTeX keys so that *JabRef* can automatically generate keys in our desired format. This can be done by selecting `Options` → `Preferences` → `Key pattern` and modifying the desired pattern in the field `Default pattern`. For instance, we use `[auth:lower][shortyear]` to get the last name of the first author in lower case and the last two digits of the year of the publication (see figure 3).

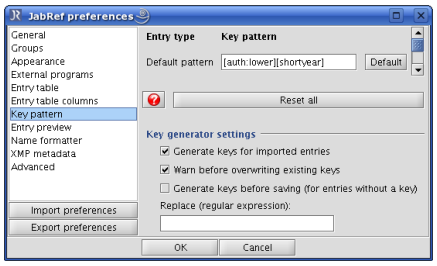


Figure 3: Specify default key pattern in *JabRef*

Second, we add the BibTeX field `location` for information about the location, where the publication is available as hard copy (e.g. a book or a copy of an article). This field can contain the name of the user who has the hard copy and where he has it or the name of a library and the shelf-mark. This field can be added in *JabRef* by selecting `Options` → `Set up general fields` and adding the word `location` (using the semicolon (`;`) as delimiter) somewhere in the line that starts with `General:` (see figure 4).

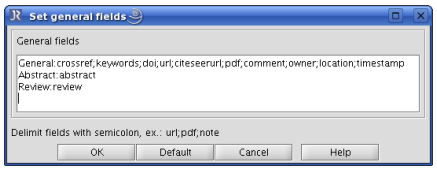


Figure 4: Set up general fields in *JabRef*

Third, we put all PDF files of publications in a specific subdirectory in our file server, where we use the BibTeX key as file name. We inform *JabRef* about this subdirectory by selecting `Options` → `Preferences` → `External programs` and adding the path of the this subdirectory in the field `Main PDF directory` (see figure 5). If a PDF file of a publication is available, the user can push the `Auto` button left of *JabRef*'s `Pdf` field to automatically add the file name of the PDF file. Now, all users who have access to the file server can open the PDF file of a publication by simply clicking on *JabRef*'s PDF icon.

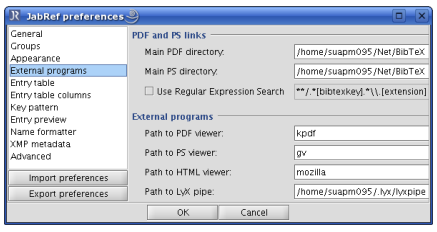


Figure 5: Specify 'Main PDF directory' in *JabRef*

If we send the LaTeX source code of a project to a journal, publisher, or somebody else who has no access to our common `texmf` tree, we do not include our entire bibliographic data base, but extract the relevant entries with the Perl script *aux2bib* (http://www.ctan.org/tex-archive/biblio/bibtex/utils/bibtools/aux2bib).

Conclusion

This wikibook describes a possible way to efficiently organise the collaborative preparation of LaTeX documents. The presented solution is based on the *Subversion* version control system and several other software tools and LaTeX packages. However, there are still a few issues that can be improved.

First, we plan that all users install the same LaTeX distribution. As the *TeX Live* distribution (http://www.tug.org/texlive/) is available both for Unix and MS Windows operating systems, we might recommend our users to switch to this LaTeX distribution in the future. (Currently, our users have different LaTeX distributions that provide a different selection of LaTeX packages and different versions of some packages. We solve this problem by providing some packages on our common `texmf` tree.)

Second, we consider to simplify the solution for a common bibliographic data base. Currently it is based on the version control system *Subversion*, the graphical BibTeX editor *JabRef*, and a file server for the PDF files of publications in the data base. The usage of three different tools for one task is rather challenging for infrequent users and users that are not familiar with these tools. Furthermore, the file server can be only accessed by local users. Therefore, we consider to implement an integrated server solution like *WIKINDX* (http://wikindx.sourceforge.net/), *Aigaion* (http://www.aigaion.nl/), or *refBASE* (http://refbase.sourceforge.net/). Using this solution only requires a computer with internet access and a web browser, which makes the usage of our data base considerably easier for infrequent users. Moreover, the stored PDF files are available not only from within the department, but throughout the world. (Depending on the copy rights of the stored PDF files, the access to the server --- or least the access to the PDF files --- has to be restricted to members of the department.) Even Non-LaTeX users of our department might benefit from a server-based solution, because it should be easier to use this bibliographic data base in (other) word processing software packages, because these servers provide the data not only in BibTeX format, but also in other formats.

All readers are encouraged to contribute to this wikibook by adding further hints or ideas or by providing further solutions to the problem of collaborative writing of LaTeX documents.

Acknowledgements

Arne Henningsen thanks Francisco Reinaldo and Géraldine Henningsen for comments and suggestions that helped him to improve and clarify this paper, Karsten Heymann for many hints and advices regarding LaTeX, BibTeX, and *Subversion*, and Christian Henning as well as his colleagues for supporting his intention to establish LaTeX and *Subversion* at their department.

References

- Fenn, Jürgen (2006): Managing citations and your bibliography with BibTeX. The PracTEX Journal, 4. http://www.tug.org/pracjourn/2006-4/fenn/.
- Markey, Nicolas (2005): Tame the BeaST. The B to X of BibTeX. http://www.ctan.org/tex-archive/info/bibtex/tamethebeast/ttb_en.pdf.Version 1.3.
- Oren Patashnik. Designing BibTeX styles. http://www.ctan.org/tex-archive/info/biblio/bibtex/contrib/doc/btxhak.pdf.
- Tools for collaborative paper-writing [16]

References

[1] http://www.scribtex.com
[2] http://www.publications.li
[3] http://www.verbosus.com
[4] http://monkeytex.bradcater.webfactional.com
[5] http://www.getdropbox.com
[6] http://docs.google.com
[7] http://docs.latexlab.org
[8] http://titanpad.com
[9] http://etherpad.org/etherpadsites.html
[10] http://en.wikipedia.org/wiki/List_of_revision_control_software
[11] http://subversion.apache.org/
[12] http://www.dyndns.com/
[13] http://tortoisesvn.tigris.org/
[14] http://zoneit.free.fr/esvn/
[15] http://kdiff3.sourceforge.net/
[16] http://mathoverflow.net/questions/3044/tools-for-collaborative-paper-writing

Internationalization

When you write documents in languages other than English, areas where LaTeX has to be configured appropriately:

1. LaTeX needs to know how to hyphenate the language(s) you are using.
2. You need to use language-specific typographic rules. In French for example, there is a mandatory space before each colon character (:).
3. You want to be able to insert all the language-specific special characters directly from your keyboard instead of using cumbersome coding (for example, type `ä` instead of `\"{a}`).

If you simply need to add a few words from another language, you may find LaTeX/Accents an easier way.

Hyphenating

The `babel` package by Johannes Braams will take care of everything. You can load it in your preamble, providing as an argument the language you want to use:

```
\usepackage[language]{babel}
```

You should place it soon after the `\documentclass` command, so that all the other packages you load afterwards will know the language you are using. A list of the languages built into your LaTeX system will be displayed every time the compiler is started. Babel will automatically activate the appropriate hyphenation rules for the language you choose. If your LaTeX format does not support hyphenation in the language of your choice, babel will still work but will disable hyphenation, which has quite a negative effect on the appearance of the typeset document. Babel also specifies new commands for some languages, which simplify the input of special characters. See the sections about languages below for more information.

If you call babel with multiple languages:

```
\usepackage[languageA,languageB]{babel}
```

then the last language in the option list will be active (i.e. languageB), and you can use the command

```
\selectlanguage{languageA}
```

to change the active language. You can also add short pieces of text in another language using the command

```
\foreignlanguage{languageB}{Text in another language}
```

Babel also offers various environments for entering larger pieces of text in another language:

```
\begin{otherlanguage}{languageB}
Text in language B. This environment switches all language-related
definitions, like the language specific names for figures, tables etc.
to the other language.
\end{otherlanguage}
```

The starred version of this environment typesets the main text according to the rules of the other language, but keeps the language specific string for ancillary things like figures, in the main language of the document. The environment `hyphenrules` switches only the hyphenation patterns used; it can also be used to disallow hyphenation by using the language name 'nohyphenation'.

Text encoding

Most of the modern computer systems allow you to input letter of national alphabets directly from the keyboard. In order to handle variety of input encoding used for different groups of languages and/or on different computer platforms LaTeX employs the `inputenc` package:

```
\usepackage[encoding]{inputenc}
```

`inputenc` package tells LaTeX what the text encoding format of your .tex files is. The encoding depends on your operating system but often a software's encoding can be changed from the settings (this happens at least with some editors, the PuTTY terminal and TeXmaker). You may choose whichever encoding you like, but you must say so in the preamble, so for example, if you prefere to use the ISO-8859-1, write

```
\usepackage[latin1]{inputenc}
```

Most modern operating systems use Unicode (utf-8) as a default encoding for text. On such system (for example Ubuntu) you can use:

```
\usepackage[utf8]{inputenc}
```

The supported encoding by the LaTeX team is `utf8` and covers a fairly specific/limited range of unicode input characters. It only defines those symbols that are known to be available with the current *font encoding*. `utf8x` is not officially supported, but covers a much broader range of input symbols.

You might encounter a situation where using `\usepackage[utf8]{inputenc}` might result in error:

```
! Package inputenc Error: Unicode char \u8:ũ not set up for use with
LaTeX.
```

This is due to the utf8 definition not necessarily having a mapping of all the character glyphs you are able to enter on your keyboard. Such characters are for example **ŷ Ŷ ũ Ũ ẽ Ẽ ĩ Ĩ**. In such case, you need to use the `utf8x` option to define more character combinations. This might break up compatibility with some packages like `csquotes`.

When using the `inputenc` package, you should consider that other people might not be able to display your input files on their computer, because they use a different encoding. For example, the German umlaut ä on OS/2 is encoded as 132, on Unix systems using ISO-LATIN 1 it is encoded as 228, while in Cyrillic encoding cp1251 for Windows this letter does not exist at all; therefore you should use this feature with care. The following encodings may come in handy, depending on the type of system you are working on:

Operating system	Encodings	
	Western Latin	Cyrillic
Mac	`applemac`	`maccyr`
Unix	`latin1`	`koi8-ru`
Windows	`ansinew`	`cp1251`
DOS, OS/2	`cp850`	`cp866nav`

Output encoding

`fontenc` package tells LaTeX how to output the text you have produced. It defines at which position inside a TeX-font each letter is stored. Multiple input encodings could be mapped into one font encoding, which reduces number of required font sets. LaTeX can produce either bitmap-fonts (usually rasterized to 300 or 600 ppi) or scalable vector fonts (such as Type 1 fonts). There are many different font sets available containing different sets of glyphs (characters).

Font encoding is set with:

```
\usepackage[encoding]{fontenc}
```

where encoding is font encoding. It is possible to load several encodings simultaneously.

The default LaTeX font encoding is OT1, the encoding of the original Computer Modern TeX text fonts. It contains only 128 characters, many from ASCII, but leaving out some others and including a number that are not in ASCII. When accented characters are required, TeX creates them by combining a normal character with an accent. While the resulting output looks perfect, this approach stops the automatic hyphenation from working inside words containing accented characters. Besides, some of Latin letters could not be created by combining a normal character with an accent, to say nothing about letters of non-Latin alphabets, such as Greek or Cyrillic.

To overcome these shortcomings, several 8-bit CM-like font sets were created. *Extended Cork* (EC) fonts in T1 encoding contains letters and punctuation characters for *most of the European languages* based on Latin script. The LH font set contains letters necessary to typeset documents in languages using Cyrillic script. Because of the large number of Cyrillic glyphs, they are arranged into four font encodings—T2A, T2B, T2C, and X2. The CB bundle contains fonts in LGR encoding for the composition of Greek text. By using these fonts you can improve/enable hyphenation in non-English documents. Another advantage of using new CM-like fonts is that they provide fonts of CM families in all weights, shapes, and optically scaled font sizes

Here is a collection of suggestions about writing a LaTeX document in a language other than English. If you have experience in a language not listed below, please add some notes about it.

Hyphenating languages

Arabic script

For languages which use the Arabic script, including Arabic, Persian, Urdu, Pashto, Kurdish, Uyghur, etc., add the following code to your preamble:

```
\usepackage{arabtex}
```

You can input text in either romanized characters or native Arabic script encodings. Use any of the following commands/environment to enter in text:

```
\< ... >
\RL{ ... }
\begin{arabtext} ... \end{arabtext}.
```

See the ArabTeX Wikipedia article for further details.

You may also use the Arabi package within babel to typeset Arabic and Persian

```
\usepackage[LAE,LFE]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[farsi,arabic]{babel}
```

You may use Arabi with Lyx, or with tex4ht to produce HTML. You may also copy and paste from PDF files produced with Arabi thanks to the support of the cmap package.

See Arabi page on CTAN [1]

Persian script

For Persian language, there is a dedicated package called XePersian which uses XeLaTeX as the typesetting engine. Just add the following code to your preamble:

```
\usepackage{xepersian}
```

See XePersian page on CTAN [2]

Moreover, Arabic script can be used to type Persian as illustrated in the previous subsection.

Cyrillic script

Please add the section "Writing in Cyrillic" from http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf . You are allowed to copy it.

See also the Bulgarian translation of the "Not so Short Introduction to LaTeX 2e" from http://www.ctan.org/tex-archive/info/lshort/bulgarian/lshort-bg.pdf

This enables you to type cyrillic letters directly via your keyboard, but with a different distribution than a standard cyrillic keyboard! To get the standard distribution, only include: \usepackage[OT1]{fontenc} \usepackage[russian]{babel}

Czech

Czech is fine using

```
\usepackage[czech]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8x]{inputenc}
```

You may use different encoding, but UTF-8 is becoming standard and it allows you to have „czech quotation marks“ directly in your text. Otherwise, there are macros *glqq* and *grqq* to produce left and right quote.

Finnish

Finnish language hyphenation is enabled with:

```
\usepackage[finnish]{babel}
```

This will also automatically change document language (section names, etc.) to Finnish.

Remember to use Unicode encoding for Finnish if you're using an editor in utf8 mode:

```
\usepackage[utf8]{inputenc}
```

The default encoding system can oftenly be changed regardless of the operating system (this happens at least with some editors, the PuTTY terminal and TeXmaker). You may choose whichever encoding you like, but you must say so in the preamble, so for example, if you prefere to use the ISO-8859-1, write instead

```
\usepackage[latin1]{inputenc}
```

The encoding is important and makes sure you can simply write the *ääkköset* ö ä å Ö Ä Å as such, which is less cumbersome than having to write

```
\"{a}
```

to get the letter ä. Actually letters like õ ô ó Ò ñ work as well so the same idea applies also to other European languages like Spanish.

If you want to use *European Computer Modern* fonts, you should use:

```
\usepackage[T1]{fontenc}
```

For creating scalable (vector) Type1 fonts instead of bitmapped fonts, you can substitute the line above with:

```
\usepackage{ae}
```

The ae-package changes encoding to T1 and also loads the scalable version of *Almost European Computer Modern* fonts.

French

Some hints for those creating French documents with LaTeX: you can load French language support with the following command:

```
\usepackage[frenchb]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
```

There are multiple options for typesetting French documents, depending on the flavor of French: `french`, `frenchb`, and `francais` for Parisian French, and `acadian` and `canadien` for new-world French. All enable French hyphenation, if you have configured your LaTeX system accordingly. All of these also change all automatic text into French: `\chapter` prints *Chapitre*, `\today` prints the current date in French and so on. A set of new commands also becomes available, which allows you to write French input files more easily. Check out the following table for inspiration:

input code	rendered output
`\og guillemets \fg{}`	« guillemets »
`M\up{me}, D\up{r}`	M^{me}, D^{r}
`1\ier{}, 1\iere{}, 1\ieres{}`	1er, 1re, 1res
`2\ieme{} 4\iemes{}`	2^{e} 4es
`\No 1, \no 2`	N° 1, n° 2
`20~\degres C, 45\degres`	20 °C, 45°
`M. \bsc{Durand}`	M. Durand
`\nombre{1234,56789}`	1 234,567 89

You will also notice that the layout of lists changes when switching to the French language. For more information on what the *frenchb* option of babel does and how you can customize its behavior, run LaTeX on file frenchb.dtx and read the produced file frenchb.pdf or frenchb.dvi.

German

You can load German language support using *either one* of the two following commands.

For old german orthography use

```
\usepackage[german]{babel}
```

or for new german orthography use

```
\usepackage[ngerman]{babel}
```

This enables German hyphenation, if you have configured your LaTeX system accordingly. It also changes all automatic text into German. Eg. “Chapter” becomes “Kapitel.” A set of new commands also becomes available, which allows you to write German input files more quickly even when you don’t use the inputenc package. Check out table 2.5 for inspiration. With inputenc, all this becomes moot, but your text also is locked in a particular encoding world.

German Special Characters.

`"a`	ä
`"s`	ß
`"` ` or `\glqq`	„
`"'` or `\grqq`	“
`"<` or `\flqq`	«
`">` or `\frqq`	»
`\flq`	‹
`\frq`	›
`\dq`	"

In German books you often find French quotation marks («guillemets»). German typesetters, however, use them differently. A quote in a German book would look like »this«. In the German speaking part of Switzerland, typesetters use «guillemets» the same way the French do. A major problem arises from the use of commands like

`\flq`: If you use the OT1 font (which is the default font) the guillemets will look like the math symbol " ≪ ", which turns a typesetter's stomach. T1 encoded fonts, on the other hand, do contain the required symbols. So if you are using this type of quote, make sure you use the T1 encoding. (`\usepackage[T1]{fontenc}`)

Greek

This is the preamble you need to write in the Greek language.

```
\usepackage[english,greek]{babel}
\usepackage[iso-8859-7]{inputenc}
```

This preamble enables hyphenation and changes all automatic text to Greek. A set of new commands also becomes available, which allows you to write Greek input files more easily. In order to temporarily switch to English and vice versa, one can use the commands `\textlatin{english text}` and `\textgreek{greek text}` that both take one argument which is then typeset using the requested font encoding. Otherwise you can use the command `\selectlanguage{...}` described in a previous section. Use `\euro` for the Euro symbol.

Hungarian

Similar to Italian, but use the following lines:

```
\usepackage[magyar]{babel}
\usepackage[latin2]{inputenc}
\usepackage[T1]{fontenc}
```

- More information in hungarian [3].

The Hungarian version of BaBeL included with standard LaTeX distribution is not perfect, a much better version can be downloaded from the previous page.

Icelandic / Faroese

The following lines can be added to write Icelandic text:

```
\usepackage[icelandic]{babel}
\usepackage[T1]{fontenc}
```

and for some users

```
\usepackage[utf8]{inputenc}
```

is needed. This allows the user to write with Icelandic characters and changes text like the *Abstract* in

```
\begin{abstract}
Þetta er útdráttur.
\end{abstract}
```

into "Útdráttur" and turns *Part*' into "Hluti".

Icelandic Special Characters.

"` or \glqq	„
\grqq	“
\TH	Þ
\th	þ
\"{O}	Ö
\"{o}	ö
\AE	Æ
\ae	æ
\DH	Ð
\dh	ð

Italian

Italian is well supported by LaTeX. Just add *\usepackage[italian]{babel}* at the beginning of your document and the output of all the commands will be translated properly. You can add letters with accents without any particular setting, just write `\`a \`e \'e \`i \`o \`u` and you will get *à è é ì ò ù* (NB: the symbol changes if the inclination of the accent changes). Anyway, if you do so, it could be quite annoying since it's time-wasting. Moreover, if you are using any spell-checking program, "città" is correct, but "citt\`a" will be seen as a mistake. If you add *\usepackage[latin1]{inputenc}* at the beginning of your document, LaTeX will include correctly all your accented letters. To sum up, just add

```
\usepackage[italian]{babel}
\usepackage[latin1]{inputenc}
```

at the beginning of your document and you can write in Italian without being worried of translations and fonts. If you are writing your document without getting any error, then don't worry about anything else. If you start getting some unknown errors whenever you use an Italian letter, then you have to worry about the encoding of your files. As known, any LaTeX source is just plain text, so you'll have to insert accented letters properly within the text file. If you write your document using always the same program on the same computer, you should not have any problem. If you are writing your document using different programs, if could start getting some strange errors from the compiler. The reason could be that the accented letters were not included properly within your source file and LaTeX can't recognize them. The reason is that an editor modified your document with a different encoding from the one that was used when creating it. Most of the operating systems use UTF-8 as default, but this could create problems if are using programs based on different libraries or different operating systems. The best way to solve this problem is to change the encoding to ISO-8859-1, that includes all the letters you need. Some text editors let you change the encoding in the settings.

Korean

To use LATEX for typesetting Korean, we need to solve three problems:

1. We must be able to edit Korean input files. Korean input files must be in plain text format, but because Korean uses its own character set outside the repertoire of US-ASCII, they will look rather strange with a normal ASCII editor. The two most widely used encodings for Korean text files are EUC-KR and its upward compatible extension used in Korean MS-Windows, CP949/Windows-949/UHC. In these encodings each US-ASCII character represents its normal ASCII character similar to other ASCII compatible encodings such as ISO-8859-x, EUC-JP, Big5, or Shift_JIS. On the other hand, Hangul syllables, Hanjas (Chinese characters as used in Korea),

Hangul Jamos, Hiraganas, Katakanas, Greek and Cyrillic characters and other symbols and letters drawn from KS X 1001 are represented by two consecutive octets. The first has its MSB set. Until the mid-1990's, it took a considerable amount of time and effort to set up a Korean-capable environment under a non-localized (non-Korean) operating system. You can skim through the now much-outdated http://jshin.net/faq to get a glimpse of what it was like to use Korean under non-Korean OS in mid-1990's. These days all three major operating systems (Mac OS, Unix, Windows) come equipped with pretty decent multilingual support and internationalization features so that editing Korean text file is not so much of a problem anymore, even on non-Korean operating systems.

2. TEX and LATEX were originally written for scripts with no more than 256 characters in their alphabet. To make them work for languages with considerably more characters such as Korean or Chinese, a subfont mechanism was developed. It divides a single CJK font with thousands or tens of thousands of glyphs into a set of subfonts with 256 glyphs each. For Korean, there are three widely used packages; HLATEX by UN Koaunghi, hLATEXp by CHA Jaechoon and the CJK package byWerner Lemberg. HLATEX and hLATEXp are specific to Korean and provide Korean localization on top of the font support. They both can process Korean input text files encoded in EUC-KR. HLATEX can even process input files encoded in CP949/Windows-949/UHC and UTF-8 when used along with Λ, Ω. The CJK package is not specific to Korean. It can process input files in UTF-8 as well as in various CJK encodings including EUC-KR and CP949/Windows-949/UHC, it can be used to typeset documents with multilingual content (especially Chinese, Japanese and Korean). The CJK package has no Korean localization such as the one offered by HLATEX and it does not come with as many special Korean fonts as HLATEX.
3. The ultimate purpose of using typesetting programs like TEX and LATEX is to get documents typeset in an 'aesthetically' satisfying way. Arguably the most important element in typesetting is a set of welldesigned fonts. The HLATEX distribution includes UHC PostScript fonts of 10 different families and Munhwabu fonts (TrueType) of 5 different families. The CJK package works with a set of fonts used by earlier versions of HLATEX and it can use Bitstream's cyberbit True-Type font.

To use the HLATEX package for typesetting your Korean text, put the following declaration into the preamble of your document:

```
\usepackage{hangul}
```

This command turns the Korean localization on. The headings of chapters, sections, subsections, table of content and table of figures are all translated into Korean and the formatting of the document is changed to follow Korean conventions. The package also provides automatic "particle selection." In Korean, there are pairs of post-fix particles grammatically equivalent but different in form. Which of any given pair is correct depends on whether the preceding syllable ends with a vowel or a consonant. (It is a bit more complex than this, but this should give you a good picture.) Native Korean speakers have no problem picking the right particle, but it cannot be determined which particle to use for references and other automatic text that will change while you edit the document. It takes a painstaking effort to place appropriate particles manually every time you add/remove references or simply shuffle parts of your document around. HLATEX relieves its users from this boring and error-prone process.

In case you don't need Korean localization features but just want to typeset Korean text, you can put the following line in the preamble, instead.

```
\usepackage{hfont}
```

For more details on typesetting Korean with HLATEX, refer to the HLATEX Guide. Check out the web site of the Korean TEX User Group (KTUG) at http://www.ktug.or.kr/.

Polish

If you plan to use Polish in your utf-8 encoded document, use the following code

```
\usepackage[utf8]{inputenc}
\usepackage{polski}
\usepackage[polish]{babel}
```

The above code merely allows to use polish letters and translates the automatic text to polish, so that "chapter" becomes "rozdział". There are a few additional things one must remember about.

Connectives

Polish has many single letter connectives: "a", "o", "w", "i", etc., grammar and typography rules don't allow for them to end a printed line. To ensure that LaTeX won't set them as last letter in the line, you have to use non breakable space:

```
Noc była sierpniowa, ciepła i~słodka, Księżyc oświecał srebrnem
światłem wgłębienie, tak,
że twarze małego rycerza i~Basi były skąpane w blasku.
Poniżej, na podwórzu zamkowem, widać było uśpione kupy żołnierzy,
a~także i~ciała zabitych
podczas dziennej strzelaniny, bo nie znaleziono dotąd czasu na ich
pogrzebanie.
```

Numerals

According to polish grammar rules, you have to put dots after numerals in chapter, section, subsection, etc. headers.

This is achieved by redefining few LaTeX macros.

For books:

```
\renewcommand\thechapter{\arabic{chapter}.}
\renewcommand\thesection{\arabic{chapter}.\arabic{section}.}
\renewcommand\thesubsection{\arabic{chapter}.\arabic{section}.\arabic{subsection}.}
\renewcommand\thesubsubsection{\arabic{chapter}.\arabic{section}.\arabic{subsection}.%

\arabic{subsubsection}.}
```

For articles:

```
\renewcommand\thesection{\arabic{section}.}
\renewcommand\thesubsection{\arabic{section}.\arabic{subsection}.}
\renewcommand\thesubsubsection{\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.}
```

Indentation

It's customary (depends on publisher) to indent first paragraph in sections and chapters:

```
\usepackage{indentfirst}
```

Hyphenation and typography

It's much more frowned upon to set pages with hyphenation between pages than it is customary in American typesetting.

To adjust penalties for hyphenation spanning pages, use this command:

```
\brokenpenalty=1000
```

To adjust penalties for leaving widows and orphans (clubs in TeX nomenclature) use those commands:

```
\clubpenalty=1000
\widowpenalty=1000
```

Further information

Refer the Słownik Ortograficzny [4] (in Polish) for additional information on polish grammar and typography rules.

Good extract is available at Zasady Typograficzne Składania Tekstu [5] (in Polish)

Portuguese

Add the following code to your preamble:

```
\usepackage[portuguese]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
```

if you are in Brazil, you can substitute the language for brazilian portuguese by choosing: `brazilian`. The first line is to get everything translated properly, the second is for being able to input text correctly and the third one to get the hyphenation of words with diacritics right. Note that we are using the latin1 input encoding here, so this will not work on a Mac or on DOS. Just use the appropriate encoding for your system. If you are using Linux, use

```
\usepackage[utf8]{inputenc}
```

Slovak

Basic settings are fine when left the same as Czech, but Slovak needs special signs for ď,ť,ľ. To be able to type them from keyboard use the following settings

```
\usepackage[slovak]{babel}
\usepackage[IL2]{fontenc}
\usepackage[utf8]{inputenc}
```

Spanish

To enable Spanish writing, besides installing the appropriate hyphenation patterns, you type:

```
\usepackage[spanish]{babel}
```

The trick is that Spanish has several options and commands to control the layout. The options may be loaded either at the call to Babel, or before, by defining the command `\spanishoptions`. Therefore, the following commands are roughly equivalent:

```
\def\spanishoptions{mexico}
\usepackage[spanish]{babel}
```

```
\usepackage[spanish,mexico]{babel}
```

On average, the former syntax should be preferred, as the latter is a deviation from standard Babel behavior, and thus may break other programs (LyX, latex2rtf2e) interacting with LaTeX.

Two particularly useful options are `es-noquoting,es-nolists`: some packages and classes are known to collide with Spanish in the way they handle active characters, and these options disable the internal workings of Spanish to allow you to overcome these common pitfalls. Moreover, these options may simplify the way LyX customizes some features of the Spanish layout from inside the GUI.

The options `mexico,mexico-com` provide support for local custom in Mexico: the former using decimal dot, as customary, and the latter allowing decimal comma, as required by the Mexican Official Norm (NOM) of the Department of Economy for labels in foods and goods. More localizations are in the making.

The other commands modify the spanish layout after loading babel. Two particularly useful commands are `\spanishoperators` and `\spanishdeactivate`.

The macro `\spanishoperators{`*list of operators*`}` contains a list of spanish mathematical operators, and may be redefined at will. For instance, the command `\def\spanishoperators{sen}` only defines sen, overriding all other definitions; the command `\let\spanishoperators\relax` disables them all. This command supports accented or spaced operators: the `\acute{`*<letter>*`}` command puts an accent, and the `\,` command adds a small space. For instance, the following operators are defined by default.

```
l\acute{i}m l\acute{i}m\,sup l\acute{i}m\,inf m\acute{a}x
\acute{i}nf m\acute{i}n sen tg arc\,sen arc\,cos arc\,tg
cotg cosec senh tgh
```

Finally, the macro `\spanishdeactivate{`*list of characters*`}` disables some active characters, to keep you out of trouble if they are redefined by other packages. The candidates for deactivation are the set `<>."'`. Please, beware that some option preempty the availability of some active characters. In particular, you should not combine the es-noquoting option with `\spanishdeactivate{<>}`, or the es-noshorthands with `\spanishdeactivate{<>."}`.

Please check the documentation for Babel or spanish.dtx for further details.

References

[1] http://tug.ctan.org/tex-archive/language/arabic/arabi/
[2] http://tug.ctan.org/tex-archive/macros/xetex/latex/xepersian/
[3] http://www.math.bme.hu/latex/
[4] http://so.pwn.pl/zasady.php
[5] http://dtp.msstudio.com.pl/typo.html

Accents

The rules for producing characters with diacritical marks, such as accents, differ somewhat depending whether you are in text mode, math mode, or the tabbing environment.

Text mode

Direct input

The Unicode character encoding UTF8 includes several special characters and characters with accents. The following code specifies that the encoding of the LaTeX document source file is UTF8. Font encoding is specified as T1, because it supports the encoding of extended character sets in fonts.

```
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

Of course, the encoding in the text editor needs to be set to utf8, as well. Depending on the used UTF8 characters, one needs to specify a font that actually includes them (and supports the T1 font enconding). E.g. for German related special characters: `\usepackage{lmodern}` .

With XeTeX and LuaTeX the inputenc and fontenc package are no longer needed and replaced by the fontspec package. Both engines support UTF-8 directly and allow the use of TTF and OpenType fonts to support Unicode characters. See the XeTeX section in the Fonts chapter of this book for more information.

Escaped codes

In addition to direct UTF8 input, LaTeX supports the composition of special characters.

The following accents may be placed on letters. Although "o" is used in most of the examples, the accents may be placed on any letter. Accents may even be placed above a "missing" letter; for example, `\~{}` produces a tilde over a blank space.

The following commands may be used only in paragraph (default) or LR (left-right) mode.

LaTeX command	Sample	Description
\`{o}	ò	grave accent
\'{o}	ó	acute accent
\^{o}	ô	circumflex
\"{o}	ö	umlaut or dieresis
\H{o}	ő	long Hungarian umlaut (double acute)
\~{o}	õ	tilde
\c{c}	ç	cedilla
\k{a}	ą	ogonek
\l	ł	l with stroke
\={o}	ō	macron accent (a bar over the letter)
\b{o}	o̲	bar under the letter
\.{o}	ȯ	dot over the letter
\d{u}	ụ	dot under the letter
\r{a}	å	ring over the letter
\u{o}	ŏ	breve over the letter
\v{s}	š	caron/hacek ("v") over the letter
\t{oo}	o͡o	"tie" (inverted u) over the two letters

To place a diacritic on top of an i or a j, its dot has to be removed. The dotless version of these letters is accomplished by typing `\i` and `\j` . For example:

- `\^{\i}` should be used for i, circumflex, î
- `\"{\i}` should be used for i, umlaut, ï

If a document is to be written completely in a language that requires particular diacritics several times, then using the right configuration allows those characters to be written directly in the document. For example, to achieve easier coding of umlauts, the babel package can be configured as `\usepackage[ngerman]{babel}` . This provides the short hand `"o` for `\"{o}`. This is very useful if one needs to use some text accents in a label, since no backslash will be accepted otherwise.

More information regarding language configuration can be found in the Internationalization section.

Math mode

Several of the above and some similar accents can also be produced in math mode. The following commands may be used only in math mode.

LaTeX command	Sample	Description	Text-mode equivalence
`\hat{o}`	$\hat{o}$	circumflex	`\^`
`\widehat{oo}`	$\widehat{oo}$	wide version of `\hat` over several letters	
`\check{o}`	$\check{o}$	vee or check	`\v`
`\tilde{o}`	$\tilde{o}$	tilde	`\~`
`\widetilde{oo}`		wide version of `\tilde` over several letters	
`\acute{o}`	$\acute{o}$	acute accent	`\'`
`\grave{o}`	$\grave{o}$	grave accent	`\``
`\dot{o}`	$\dot{o}$	dot over the letter	`\.`
`\ddot{o}`	$\ddot{o}$	two dots over the letter (umlaut in text-mode)	`\"`
`\breve{o}`	$\breve{o}$	breve	`\u`
`\bar{o}`	$\bar{o}$	macron	`\=`
`\vec{o}`	$\vec{o}$	vector (arrow) over the letter	

Tabbing environment

Some of the accent marks used in running text have other uses in the tabbing environment. In that case they can be created with the following command:

- `\a'` for an acute accent
- `\a`` for a grave accent
- `\a=` for a macron accent

Related issues

The actual entering of special characters is system dependent. For example under X-Windows umlauts are entered via `compose+caps+" o`, when the keyboard layout does not directly includes them.

Whether LaTeX is actually able to typeset certain UTF8 characters depends on the loaded packages. For example trying to typeset the euro sign (€) may yield following error message:

```
Package inputenc Error: Unicode char \u8:€ not set up for use with LaTeX.
```

Using the package `textcomp` sets this and other characters up.

The usage of the T1 font encoding influences the usability of pdf output (generated via pdflatex). For example without specifying T1, extracting the umlaut `Ä` via a PDF viewer actually extracts the two characters `"A`. Analog to that, a PDF viewer cannot find words with umlauts in a PDF document which was generated via pdflatex without the T1 font enconding. With T1 font enconding (and even with composed special characters) the PDF contains the correct text information.

The package `ae` (almost european) is obsolete. It provided some workarounds for hyphenation of words with special characters. These are not necessary any more with fonts like lmodern. Using the ae package leads to text encoding problems in PDF files generated via `pdflatex` (e.g. text extraction and searching), besides typographic issues.

External links

- A few other LaTeX accents and symbols [1]
- NASA GISS: Accents [2]

References

[1] http://spectroscopy.mps.ohio-state.edu/symposium_53/latexinstruct.html
[2] http://www.giss.nasa.gov/tools/latex/ltx-401.html

Appendix

Tips and Tricks

id est and *exempli gratia* (i.e. and e.g.)

If you simply use the forms "`i.e.`" or "`e.g.`", LaTeX will treat the periods as end of sentence periods (i.e. full stop) since they are followed by a space, and add more space before the next "sentence". To prevent LaTeX from adding space after the last period, the correct syntax is either "`i.e.\ `" or "`e.g.\ `".

Depending on style (e.g., *The Chicago Manual of Style*), a comma can be used afterwards, which is interpreted by LaTeX as part of a sentence, since the period is not followed by any space. In this case, "`i.e.,`" and "`e.g.,`" do not need any special attention.

If the command `\frenchspacing` is used in the preamble, the space between sentences is always consistent.

Grouping Figure/Equation Numbering by Section

For long documents the numbering can become cumbersome as the numbers reach into double and triple digits. To reset the counters at the start of each section and prefix the numbers by the section number, include the following in the preamble.

```
\usepackage{amsmath}
\numberwithin{equation}{section}
\numberwithin{figure}{section}
```

The same can be done with similar counter types and document units such as "subsection".

Generic header

As explained in the previous sections, a LaTeX source can be used to generate both a DVI and a PDF file. For very basic documents the source is the same but, if the documents gets more complicated, it could be necessary to make some changes in the source so that it will work for a format but it will not for the other. For example, all that is related to graphics has to be adapted according to the final format. As discussed in the section about floating objects, even if you should use different pictures according to the final format, you can override this limit putting in the same folder pictures in different formats (e.g., EPS and PNG) with the same name and link them without writing the extension. There is a simple way to solve this problem:

```
\usepackage{ifpdf}
```

or, if you don't have this package, you can add the following text just after `\documentclass[...]{...}` :

```
\newif\ifpdf
\ifx\pdfoutput\undefined
  \pdffalse
\else
  \ifnum\pdfoutput=1
    \pdftrue
  \else
```

```
    \pdffalse
  \fi
\fi
```

this is plain TeX code. The *ifpdf* package and this code, both define a new *if-else* you can use to change your code according to the compiler you are using. After you have used this code, you can use whenever you want in your document the following syntax:

```
\ifpdf
    % we are running pdflatex
\else
    % we are running latex
\fi
```

place after `\ifpdf` the code you want to insert if you are compiling with *pdflatex*, place after `\else` the code you want to insert if you are compiling with *latex*. For example, you can use this syntax to load different packages according to the compiler.

Graphics and Graph editors

Vector image editors with LaTeX support

It is often preferable to use the same font and font size in your images as in the document. Moreover, for scientific images, you may need mathematical formulae or special characters (such as Greek letters). Both things can be achieved easily if the image editor allows you to use LaTeX code in your image. Most vector image editors do not offer this option. There are, however, a few exceptions.

In early days, LaTeX users used Xfig for their drawings. The editor is still used by quite a few people nowadays because it has special 'export to LaTeX' features. It also gives you some very basic ways of encapsulating LaTeX text and math in the image (setting the text's 'special flag' to 'special' instead of 'normal'). When exporting, all LaTeX text will be put in a .tex-file, separately from the rest of the image (which is put in a .ps file).

A newer and easier-to-use vector image editor specially tailored to LaTeX use is IPE. It allows any LaTeX command, including but not limited to mathematical formulae in the image. The program saves its files as editable .eps or .pdf files, which eliminates the need of exporting your image each time you have edited it.

A very versatile vector image editor is Inkscape. It does not support LaTeX text by itself, but you can use the plugin Textext [7] for that. This allows you to put any block of LaTeX code in your image. Additionally since version 0.48 you can export to vectorgraphics with texts separated in a .tex file. Using this way text is rendered by the latex compiler itself.

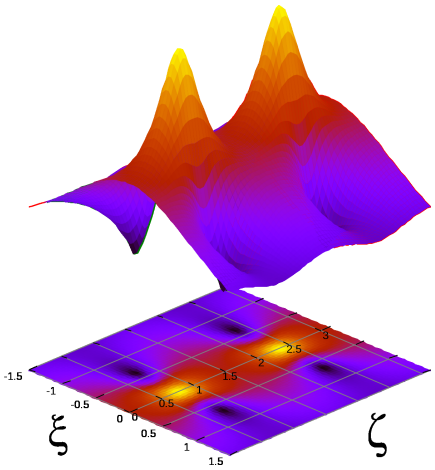
Another way to generate vectorgraphics is using the Asymptote language. It is a programming language which produces vector images in encapsulated postscript format and supports LaTeX syntax in any textlabels.

Graphs with gnuplot

A simple method to include graphs and charts in LaTeX documents is to create it within a common spreadsheet software (OpenOffice Calc or MS Office Excel etc.) and include it in the document as a cropped screenshot. However, this produces poor quality rasterized images. Calc also allows you to copy-paste the charts into OpenOffice Draw and save them as PDF files.

Using Microsoft Excel 2010, charts can be copied directly to Microsoft Expression Design 4, where they can be saved as PDF files. These PDF files can be included in LaTeX. This method produces high quality vectorized images.

An excellent method to render graphs is through **gnuplot**, a free and versatile plotting software, that has a special output filter directly for exporting files to LaTeX. We assume, that the data is in a CSV file (comma separated text) in the first and third column. A simple gnuplot script to plot the data can look like this:



gnuplot can plot various numerical data, functions, error distribution as well as 3D graphs and surfaces

```
set format "$%g$"
set title "Graph 3: Dependence of $V_p$ on $R_0$"
set xlabel "Resistance $R_0$ [$\Omega$]"
set ylabel "Voltage $V_p$ [V]"
set border 3
set xtics nomirror
set ytics nomirror
set terminal epslatex
set output "graph1.eps"
plot "graph1.csv" using 1:3   #Plot the data
```

Now gnuplot produces two files: the graph drawing in `graph.eps` and the text in `graph.tex`. The second includes the EPS image, so that we only need to include the file graph.tex in our document:

```
\input{graph1.tex}
```

The above steps can be automated by the package gnuplottex. By placing gnuplot commands inside \begin{gnuplot}\end{gnuplot}, and compiling with latex -shell-escape, the graphs are created and added into your document.

When using pdfLaTeX instead of simple LaTeX, we must convert the EPS image to PDF and to substitute the name in the `graph1.tex` file. If we are working with a Unix-like shell, it is simply done using:

```
eps2pdf graph1.eps
sed -i s/".eps"/".pdf"/g graph1.tex
```

With the included tex file we can work as with an ordinary image.

Instead of calling `eps2pdf` directly, we can also include the `epstopdf` package that automates the process. If we include a graphics now and leave out the file extension, `epstopdf` will automatically transform the .eps-file to PDF and insert it in the text.

```
\includegraphics{graph1}
```

This way, if we choose to output to PS or DVI, the EPS version is used and if we output to PDF directly, the converted PDF graphics is used. Please note that usage of `epstopdf` requires compiling with latex -shell-escape.

Note: Emacs AucTex users might want to check out Gnuplot-mode [1].

Generate png screenshots

This section describes how to generate a png screenshot of a LaTeX page using the *nix tool `dvipng` and the LaTeX package `preview`. This screenshots are useful, for example, if you want to include a LaTeX generated formula on a presentation using you favorite slideware like Powerpoint, Keynote or OpenOffice Impress. First, start by making sure you have the two required tools `dvipng` and `preview`. To check for the LaTeX package type

```
locate preview.sty
```

on a console and if locate returns a directory, then you are good. Next, type

```
which dvipng
```

to see if you have `dvipng` installed on your machine and ready to use. Again, if you get a directory, you are ready to start doing screenshot of LaTeX pages. Say you want to take a screenshot of

$$\pi = \sqrt{12}\sum_{k=0}^{\infty}\frac{(-3)^{-k}}{2k+1}.$$

Write this formula on a TeX file name `foo.tex` in this way:

```
\documentclass{article}
\usepackage[active]{preview}
\begin{document}
\begin{preview}
\[
\pi = \sqrt{12}\sum^\infty_{k=0} \frac{(-3)^{-k}}{2k+1}
\]
\end{preview}
\end{document}
```

Run LaTeX as usual to generate the dvi file `foo.dvi`. Note the `active` option in the package declaration and the preview environment around the equation's code. Without any of these two, you won't get an `dvi` output. Now, we want an X font size formula, where X is measure in pixels. You need to convert this, to dots per inch (dpi). The formula is: `<dpi> = <font_px>*72.27/10`. If you want, for instance, X = 32, then the size in dpi corresponds to 231.26. This value will be passed to `dvipng` using the flag `-D`. To generate the desired png file run the command as follows:

```
dvipng -T tight -D 231.26 -o foo.png foo.dvi
```

The flag `-T` sets the size of the image. The option `tight` will only include all ink put on the page. The option `-o` sends the output to the file name `foo.png`.

Spell-checking and Word Counting

If you want to spell-check your document, you can use command-line `aspell` (preferably) or `ispell` programs.

```
ispell yourfile.tex
aspell -c yourfile.tex
```

Both understand LaTeX and will skip LaTeX commands. You can also use LyX or Kile — graphical LaTeX editors with built-in spell checking. Last another option is to convert LaTeX source to plain text and open resulting file in a word processor like OpenOffice.org or KOffice.

If you want to count words you can, again, use LyX or convert your LaTeX source to plain text and use, for example, UNIX `wc` command:

```
detex yourfile | wc
```

An alternative to the `detex` command is the `pdftotext` command which extracts an ASCII text file from PDF:

```
1. pdflatex yourfile.tex
2. pdftotext yourfile.pdf
3. wc yourfile.txt
```

New even page

In the twoside-mode you have the ability to get a new odd-side page by:

```
\cleardoublepage
```

However, LaTeX doesn't give you the ability to get a new even-side page. The following method opens up this;

The following must be put in your document preamble:

```
\usepackage{ifthen}

\newcommand{\newevenside}{
      \ifthenelse{\isodd{\thepage}}{\newpage}{
      \newpage
        \phantom{placeholder} % doesn't appear on page
      \thispagestyle{empty} % if want no header/footer
      \newpage
      }
}
```

To active the new even-side page, type the following where you want the new even-side:

```
\newevenside
```

If the given page is an odd-side page, the next new page is subsequently an even-side page, and LaTeX will do nothing more than a regular \newpage. However, if the given page is an even page, LaTeX will make a new (odd) page, put in a placeholder, and make another new (even) page. A crude but effective method.

Hide auxiliary files

If you're using pdflatex you can create a folder in which all the output files will be stored, so your top directory looks cleaner.

```
pdflatex -output-directory tmp
```

Please note that the folder tmp should exist. However if you're using linux you can do something like this:

```
alias pdflatex='mkdir tmp; pdflatex -output-directory tmp'
```

Or for vim modify your texrc:

```
TexLet g:Tex_CompileRule_pdf = 'mkdir tmp; pdflatex -output-directory
tmp -interaction=nonstopmode $*'
```

References

[1] http://cars9.uchicago.edu/~ravel/software/gnuplot-mode.html

Useful Measurement Macros

A list of macros and their values

Units

First, we introduce the LaTeX measurement units. All LaTeX units are two-letter abbreviations. You can choose from a variety of units. Here are the most common ones.[1]

Abbreviation	Definition	Value in points (pt)
pt	a point is 1/72.27 inch, that means about 0.0138 inch or 0.3515 mm.	1
mm	a millimeter	2.84
cm	a centimeter	28.4
in	inch	72.27
ex	roughly the height of an 'x' **in the current font**	*undefined, depend on the font used*
em	roughly the width of an 'M' (uppercase) **in the current font**	*undefined, depend on the font used*

And here are some less common units.[2]

Abbreviation	Definition	Value in points (pt)
bp	a big point is 1/72 inch, that means about 0.0139 inch or 0.3527 mm.	1.00375
pc	pica	12
dd	didôt (1157 didôt = 1238 points)	1.07
cc	cîcero (12 didôt)	12.84
sp	scaled point (65536sp per point)	0.000015

Length 'macros'

Some length commands are;

\baselineskip

The normal vertical distance between lines in a paragraph

\baselinestretch

Multiplies \baselineskip

\columnsep

The distance between columns

\columnwidth

The width of the column

\evensidemargin

The margin for 'even' pages (think of a printed booklet)

\linewidth

The width of a line in the local environment

\oddsidemargin

The margin for 'odd' pages (think of a printed booklet)

\paperwidth

The width of the page

\paperheight

The height of the page

\parindent

The normal paragraph indentation

\parskip

The extra vertical space between paragraphs

\tabcolsep

The default separation between columns in a tabular environment

\textheight

The height of text on the page

\textwidth

The width of the text on the page

\topmargin

The size of the top margin

\unitlength

Units of length in Picture Environment

Length manipulation macros

You can change the values of the variables defining the page layout with two commands. With this one you can set a new value:

```
\setlength{parameter}{length}
```

with this other one, you can add a value to the existing one:

```
\addtolength{parameter}{length}
```

You can create your own length with the command:

```
\newlength{parameter}
```

You may also set a length from the size of a text with one of these commands:

```
\settowidth{parameter}{some text}
\settoheight{parameter}{some text}
\settodepth{parameter}{some text}
```

When using these commands, you may duplicate the text that you want to use as reference if you plan to also display it. But LaTeX also provides a set of commands to avoid this duplication:

```
\newsavebox{boxname}
\savebox{boxname}{some text}
\usebox{boxname}
```

You may wish to look at the example below to see how you can use these. The command `\newsavebox` creates a placeholder for storing a text; the command `\savebox` stores the specified text in this placeholder, and does not display anything in the document; and `\usebox` recalls the content of the placeholder into the document.

See LaTeX/Page_Layout for samples using these.

Samples

Resize an image to take exactly half the text width :

```
\includegraphics[width=0.5\textwidth]{mygraphic}
```

Make distance between items larger (inside an itemize environment) :

```
\addtolength{\itemsep}{0.5\baselineskip}
```

Use of *savebox* to resize an image to the height of the text:

```
% Create the holders we will need for our work
\newlength{\mytitleheight}
\newsavebox{\mytitletext}
% Create the reference text for measures
\savebox{\mytitletext}{%
  \Large\bfseries This is our title%
}
```

```
\settoheight{\mytitleheight}{\usebox{\mytitletext}}
% Now creates the actual object in our document
\framebox[\textwidth][l]{%
  \includegraphics[height=\mytitleheight]{my_image}%
  \hspace{2mm}%
  \usebox{\mytitletext}%
}
```

References

[1] http://www.uz.ac.zw/science/maths/latex/ltx-86.html
[2] http://www.uz.ac.zw/science/maths/latex/ltx-86.html

Export To Other Formats

Strictly speaking, LaTeX source can be used to directly generate two formats:

- DVI using *latex*, the first one to be supported
- PDF using *pdflatex*, more recent

Using other software freely available on Internet, you can easily convert DVI and PDF to other document formats. In particular, you can obtain the PostScript version using software which is included in your LaTeX distribution. Some LaTeX IDE will give you the possibility to generate the PostScript version directly (even if it uses internally a DVI mid-step, e.g. LaTeX → DVI → PS). It is also possible to create PDF from DVI and vice versa. It doesn't seem logical to create a file with two steps when you can create it straight away, but some users might need it because, as you remember from the first chapters, the format you can generate depends upon the formats of the images you want to include (EPS for DVI, PNG and JPG for PDF). Here you will find sections about different formats with description about how to get it.

Other formats can be produced, such as RTF (which can be used in Microsoft Word) and HTML. However, these documents are produced from software that parses and interprets the LaTeX files, and do not implement all the features available for the primary DVI and PDF outputs. Nonetheless, they do work, and can be crucial tools for collaboration with colleagues who do not edit documents with LaTeX.

Convert to PDF

Directly

```
pdflatex my_file
```

DVI to PDF

```
dvipdfm my_file.dvi
```

will create `my_file.pdf`. Another way is to pass through PS generation:

```
dvi2ps myfile.dvi
ps2pdf myfile.ps
```

you will get also a file called *my_file.ps* that you can delete.

Merging PDF

If you have created different PDF documents and you want to merge them into one single PDF file you can use the following command-line command. You need to have Ghostscript installed:

For Windows:

```
gswin32 -dNOPAUSE -sDEVICE=pdfwrite -sOUTPUTFILE=Merged.pdf -dBATCH 1.pdf 2.pdf 3.pdf
```

For Linux:

```
gs -dNOPAUSE -sDEVICE=pdfwrite -sOUTPUTFILE=Merged.pdf -dBATCH 1.pdf 2.pdf 3.pdf
```

Alternatively, PDF-Shuffler [1] is a small python-gtk application, which helps the user to merge or split pdf documents and rotate, crop and rearrange their pages using an interactive and intuitive graphical interface. This program may be avaliable in your Linux distribution's repository.

Another option to check out is pdftk [2] (or PDF toolkit), which is a command-line tool that can manipulate PDFs in many ways. To merge one or more files, use:

```
pdftk 1.pdf 2.pdf 3.pdf cat output 123.pdf
```

Note: If you are merging external PDF documents into a Latex document which is compiled with `pdflatex`, a much simpler option is to use the `pdfpages` package, e.g.:

```
\usepackage{pdfpages}
...
\includepdf[pages=-]{Document1.pdf}
\includepdf[pages=-]{Document2.pdf}
...
```

Three simple shell scripts using the pdfpages package are provided in the pdfjam bundle [3] by D. Firth. They include options for merge several pdf (pdfjoin), put several pages in one physical sheet (pdfnup) and rotate pages (pdf90).

XeTeX

You can also use XeTeX (or, more precisely, XeLaTeX), which works in the same way as pdflatex: it creates a PDF file directly from LaTeX source. One advantage of XeTeX over standard LaTeX is support for Unicode and modern typography. See its Wikipedia entry for more details.

Customization of PDF output in XeTeX (setting document title, author, keywords etc.) is done using the configuration of hyperref package.

Convert to PostScript

from PDF

```
pdf2ps my_file.pdf
```

from DVI

```
dvi2ps my_file.dvi
```

Convert to RTF

LaTeX can be converted into an RTF file, which in turn can be opened by a word processor such as OpenOffice.org Writer or Microsoft Word. This conversion is done through latex2rtf [4], which can run on any computer platform. The program operates by reading the LaTeX source, and mimicking the behaviour of the LaTeX program.

`latex2rtf` supports most of the standard implementations of LaTeX, such as standard formatting, some math typesetting, inclusion of EPS, PNG or JPG graphics, and tables. As well, it has some limited support for packages, such as varioref, and natbib. However, many other packages are not supported.

latex2rtf is simple to use. The Windows version has a GUI (`l2rshell.exe`), which is straightforward to use. The command-line version is offered for all platforms, and can be used on an example `mypaper.tex` file:

```
latex mypaper
bibtex mypaper # if you use bibtex
latex2rtf mypaper
```

Both `latex` and (if needed) `bibtex` commands need to be run *before* `latex2rtf`, because the `.aux` and `.bbl` files are needed to produce the proper output. The result of this conversion will create `myfile.rtf`, which you may open in many modern word processors such as Microsoft word or Open Office.

Convert to HTML

There are many converters to HTML. One option is the HEVEA [5] program:

LaTeX

```
hevea mylatexfile
```

BibTeX

```
bibtex2html mybibtexfile
```

TeX4ht

TeX4ht [6] is a very powerful conversion program, but its configuration is not straightforward. Basically a configuration file has to be prepared, and then the program is called.

Convert to image formats

PNG

To convert from PDF, open your file with GIMP. It will ask you which page you want to convert, whether you want to use anti-aliasing (choose *strong* if you want to get something similar to what you see on the screen). Try different resolutions to fit your needs, but 100 dpi should be enough. Once you have the image within GIMP, you can post-process it as you like and save it to any format supported by GIMP, as PNG for example. A method for DVI files is dvipng [7] (usage is the same as dvipdfm). Also, the "convert" command from the ImageMagick [8] suite can convert both DVI and PDF files to PNG.

You can optimize the resulting image using optipng [8] so that it will take up less space.

SVG

Convert it to PS as described before, then use the bash script ps2svg.sh [9] (it could be possible to write a step-by-step guide for Windows as well; all the software it uses is multiplatform).

One can also use dvisvgm [10], an open source utility that converts from DVI to SVG.

Convert to plain text

If you are thinking of converting to plain text to perform a spell-checking or count words, read Tips and Tricks first.

Most LaTeX distributions come with `detex` program, which strips LaTeX commands. It can handle multi-file projects, so all you need is to give one command:

```
detex yourfile
```

(note the omission of .tex extension). This will output result to standard output. If you want the plain text go to a file, use

```
detex yourfile > yourfile.txt
```

If the output from `detex` does not satisfy you, you can try a newer version available on Google Code [11], or use HTML conversion first and then copy text from your browser.

References

[1] http://pdfshuffler.sourceforge.net/
[2] http://www.accesspdf.com/
[3] http://www2.warwick.ac.uk/fac/sci/statistics/staff/academic/firth/software/pdfjam
[4] http://latex2rtf.sourceforge.net/
[5] http://hevea.inria.fr
[6] http://www.cse.ohio-state.edu/~gurari/TeX4ht/
[7] http://savannah.nongnu.org/projects/dvipng/
[8] http://optipng.sourceforge.net/
[9] http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Electronics/Ps2svg.sh
[10] http://dvisvgm.sourceforge.net/
[11] http://code.google.com/p/opendetex/

Command Glossary

This is a glossary of LaTeX commands—an alphabetical listing of LaTeX commands with the summaries of their effects. (Brackets "[]" are optional arguments and braces "{}" are required arguments.)

#

/
: see slash marks

\@
: following period ends sentence

\\[*][extra-space]
: new line. See Page Layout.

\,
: thin space, math and text mode

\;
: thick space, math mode

\:
: medium space, math mode

\!
: negative thin space, math mode

\-
: hyphenation; tabbing

\=
: set tab, see tabbing

\>
: tab, see tabbing

\<
: back tab, see tabbing

\+
: see tabbing

\'
: accent or tabbing

\`
: accent or tabbing

\|
: double vertical lines, math mode

\(
: start math environment

\)
: end math environment

\[

begin displaymath environment

\]

end displaymath environment

A

\addcontentsline{file}{sec_unit}{entry}

adds an entry to the specified list or table

\addtocontents{file}{text}

adds text (or formatting commands) directly to the file that generates the specified list or table

\addtocounter{counter}{value}

increments the counter

\address{Return address}

\addtolength{len-cmd}{len}

increments a length command, see Useful Measurement Macros

\addvspace

adds a vertical space of a specified height

\alph

causes the current value of a specified counter to be printed in alphabetic characters

\appendix

changes the way sectional units are numbered so that information after the command is considered part of the appendix

\arabic

causes the current value of a specified counter to be printed in Arabic numbers

\author

declares the author(s). See Document Structure

B

\backslash

prints a backslash

\baselineskip

a length command (see Useful Measurement Macros), which specifies the minimum space between the bottom of two successive lines in a paragraph

\baselinestretch

scales the value of \baselineskip

\bf

Boldface typeface

\bibitem

generates a labeled entry for the bibliography

\bigskipamount

\bigskip

: equivalent to \vspace{\bigskipamount}

\boldmath

: bold font in math mode

C

\cal

: Calligraphic style in math mode

\caption

: generate caption for figures and tables

\cdots

: Centered dots

\centering

: Used to center align LaTeX environments

\chapter

: Starts a new chapter. See Document Structure.

\circle

\cite

: Used to make citations from the provided bibliography

\cleardoublepage

\clearpage

: Ends the current page and causes any floats to be printed. See Page Layout.

\cline

: Adds horizontal line in a table that spans only to a range of cells. See \hline and ../Tables/ chapter.

\closing

: Inserts a closing phrase (e.g. \closing{yours sincerely}), leaves space for a handwritten signature and inserts a signature specified by \signature{}. Used in the *Letter* class.

\color

: Specifies color of the text. ../Colors

\copyright

: makes © sign. See Formatting.

D

\dashbox

\date

\ddots

Inserts a diagonal ellipsis (3 diagonal dots) in math mode

\documentclass[options]{style}

Used to begin a latex document

\dotfill

E

\em

Toggles italics on/off for the text inside curly braces with the command. Such as {\em This is in italics \em but this isn't \em and this is again}. This command allows nesting.

\emph

Toggles italics on/off for the text in curly braces following the command e.g. \emph{This is in italics \emph{but this isn't} and this is again}.

\ensuremath (LaTeX2e)

\euro

Prints euro € symbol. Requires `eurosym` package.

F

\fbox

\flushbottom

\fnsymbol

\footnote

Creates a footnote.

\footnotemark

\footnotesize

Sets font size. See Formatting.

\footnotetext

\frac

inserts a fraction in mathematics mode. The usage is \frac{numerator}{denominator}.

\frame

\framebox

Like \makebox but creates a frame around the box. See LaTeX/Advanced Topics#Boxes.

\frenchspacing

Instructs LaTex to abstain from inserting more space after a period (´.´) than is the case for an ordinary character. In order to untoggle this functionality resort to the command \nonfrenchspacing.

H

\hfill

Abbreviation for \hspace{\fill}.

\hline

adds a horizontal line in a tabular environment. See also \cline, Tables chapter.

\hrulefill

\hspace

Produces horizontal space.

\huge

Sets font size. See Formatting.

\Huge

Sets font size. See Formatting.

\hyphenation{word list}

Overrides default hyphenation algorithm for specified words. See Hyphenation

I

\include

This command is different from `\input` in that it's the output that is added instead of the commands from the other files. For more see LaTex/Basics

\includegraphics

Inserts an image. Requires `graphicx` package.

\includeonly

\indent

\input

Used to read in LaTex files. For more see LaTex/Basics.

\it

Italicizes the text which is inside curly braces with the command. Such as {\it This is in italics}. \em is generally preferred since this allows nesting.

\item

Creates an item in a list. Used in list structures.

K

\kill

L

\label

Used to create label which can be later referenced with `\ref`. See Labels and Cross-referencing.

\large

Sets font size. See Formatting.

\Large

Sets font size. See Formatting.

\LARGE

Sets font size. See Formatting.

\LaTeX

Prints LaTeX logo. See Formatting.

\LaTeXe

Prints current LaTeX version logo. See Formatting.

\ldots

Prints sequence of three dots. See Formatting.

\left

\lefteqn

\line

\linebreak

Suggests LaTeX to break line in this place. See Page Layout.

\linethickness

\linewidth

\listoffigures

Inserts a list of the figures in the document. Similar to Document_Structure#Table_of_contents

\listoftables

Inserts a list of the tables in the document. Similar to Document_Structure#Table_of_contents

\location

M

\makebox

Defines a box that has a specified width, independent from its content. See LaTeX/Advanced Topics#Boxes.

\maketitle

Causes the title page to be typeset, using information provided by commands such as \title{} and \author{}.

\markboth \markright

\mathcal

\mathop

\mbox

\medskip

\multicolumn

\multiput

N

\newcommand

Defines a new command. See New Commands.

\newcounter

\newenvironment

Defines a new environment. See New Environments.

\newfont

\newlength

\newline

Ends current line and starts a new one. See Page Layout.

\newpage

Ends current page and starts a new one. See Page Layout.

\newsavebox

\newtheorem

\nocite

Adds a reference to the bibliography without an inline citation. \nocite{*} causes all entries in a bibtex database to be added to the bibliography.

\noindent

\nolinebreak

\nonfrenchspacing

Setting the command untoggles the command \frenchspacing and activates LaTeX standards to insert more space after a period (´.´) than after an ordinary character.

\normalsize

Sets default font size. See Formatting.

\nopagebreak

Suggests LaTeX not to break page in this place. See Page Layout.

\not

O

\onecolumn

\opening

Inserts an opening phrase when using the *letter* class, for example \opening{Dear Sir}

\oval

\overbrace

Draws a brace over the argument. Can be used in displaystyle with superscript to label formulae. See Advanced Mathematics.

\overline

Draws a line over the argument.

P

\pagebreak

Suggests LaTeX breaking page in this place. See Page Layout.

\pagenumbering

\pageref

Used to reference to number of page where a previously declared `\label` is located. See Floats, Figures and Captions.

\pagestyle

See Page Layout.

\par

Starts a new paragraph

\paragraph

Starts a new paragraph. See Document Structure.

\parbox

Defines a box whose contents are created in paragraph mode. See Advanced Topics.

\parindent

Normal paragraph indentation. See Useful Measurement Macros.

\parskip

\part

Starts a new part of a book. See Document Structure.

\protect

\providecommand (LaTeX2e)

See Customizing LaTeX.

\put

R

\raggedbottom

Command used for top justified within other environments.

\raggedleft

Command used for right justified within other environments.

\raggedright

Command used for left justified within other environments.

\raisebox

Creates a box and raises its content. See LaTeX/Advanced Topics#Boxes.

\ref

Used to reference to number of previously declared `\label`. See Labels and Cross-referencing.

\renewcommand

\right

\rm

\roman

\rule

Creates a line of specified width and height. See LaTeX/Advanced Topics#Rules and Struts.

S

\savebox

Makes a box and saves it in a named storage bin.

\sbox

The short form of \savebox with no optional arguments.

\sc

\scriptsize

Sets font size. See Formatting.

\section

Starts a new section. See Document Structure.

\setcounter

\setlength

\settowidth

\sf

\shortstack

\signature

In the *Letter* class, specifies a signature for later insertion by \closing.

\sl

\slash

See slash marks

\small

Sets font size. See Formatting.

\smallskip

\sout

Strikes out text. Requires `ulem` package. See Formatting.

\space

force ordinary space

\sqrt

Creates a root (default square, but magnitude can be given as an optional parameter).

\stackrel

Takes two arguments and stacks the first on top of the second.

\subparagraph

Starts a new subparagraph. See Document Structure.

\subsection

Starts a new subsection. See Document Structure.

\subsubsection

Starts a new sub-subsection. See Document Structure.

T

\tableofcontents

Inserts a table of contents (based on section headings) at the point where the command appears.

\telephone

In the *letter* class, specifies the sender's telephone number.

\TeX

Prints TeX logo. See Formatting.

\textbf{}

Sets bold font style. See Formatting.

\textcolor{}{}

Creates colored text. See Entering colored text.

\textit{}

Sets italic font style. See Formatting.

\textmd{}

Sets medium weight of a font. See Formatting.

\textnormal{}

Sets normal font. See Formatting.

\textrm{}

Sets roman font family. See Formatting.

\textsc{}

Sets font style to small caps. See Formatting.

\textsf{}

Sets sans serif font family. See Formatting.

\textsl{}

Sets slanted font style. See Formatting.

\texttt{}

Sets typewriter font family. See Formatting.

\textup{}

Sets upright shape of a font. See Formatting.

\textwidth

\textheight

\thanks

\thispagestyle

\tiny

Sets font size. See Formatting.

\title

\today

Writes current day. See Formatting.

\tt

\twocolumn

\typeout

\typein

U

\uline

Underlines text. Requires `ulem` package. See Formatting.

\underbrace

\underline

\unitlength

\usebox

\usecounter

\uwave

Creates wavy underline. Requires `ulem` package. See Formatting.

V

\value

\vbox{text}

Encloses a paragraph's text to prevent it from running over a page break

\vdots

Creates vertical dots. See Mathematics.

\vector

\verb

Creates inline verbatim text. See Formatting.

\vfill

\vline

\vphantom

\vspace

This page uses material from Dr. Sheldon Green's Hypertext Help with LaTeX [1].

References

[1] http://www.giss.nasa.gov/tools/latex/

Index

This is an alphabetical index of the book.

: Top - 0–9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

B

C

D

E

F

G

H

I

L

M

P

Q

R

S

T

U

V

W

X

Article Sources and Contributors

Introduction *Source*: http://en.wikibooks.org/w/index.php?oldid=2152387 *Contributors*: 3mta3, Alejo2083, Anarchyboy, Chazz, Conrad.Irwin, Dan Polansky, Derbeth, E.lewis1, Elwikipedista, Goodgerster, Graemeg, Gronau, Iamunknown, Igjimh, Jraregris, Kernigh, Latexing, Mckay, Nbrouard, Orderud, PAC2, Pdelong, Pi zero, Piksi, Polytropos Technikos, QuiteUnusual, Rehoot, Sgenier, Sjlegg, Tomato86, Urhixidur, Vadik wiki, Whiteknight, Withinfocus, Ævar Arnfjörð Bjarmason, 85 anonymous edits

Absolute Beginners *Source*: http://en.wikibooks.org/w/index.php?oldid=2111620 *Contributors*: ABCD, Adrignola, Alejo2083, Atallcostsky, Bilbo1507, BrettMontgomery, Chisophugis, Chuckhoffmann, CrazyTerabyte, Derbeth, Dilaudid, E.lewis1, Igjimh, Immae, Janskalicky, Jguk, Jtwdog, Krischik, Mwtoews, Pi zero, Qeny, Rehoot, Snaxe920, Thenub314, Tomato86, TomyDuby, Tpr, Vadik wiki, Withinfocus, 49 anonymous edits

Basics *Source*: http://en.wikibooks.org/w/index.php?oldid=2096013 *Contributors*: 3mta3, Alejo2083, Derbeth, Guyrobbie, Jonathan Webley, Mwtoews, PAC2, Pi zero, Thenub314, Tomato86, Vadik wiki, Waldir, Withinfocus, 26 anonymous edits

Document Structure *Source*: http://en.wikibooks.org/w/index.php?oldid=2164286 *Contributors*: Adrignola, Alejo2083, Astrophizz, BiT, ConditionalZenith, Derbeth, Derwaldrandfoerster, Dilaudid, Echeban, FlashSheridan, Frap, Hosszuka, Igjimh, JECompton, Jan Winnicki, Jtwdog, Keplerspeed, Kovianyo, Lucasreddinger, Mwtoews, Neet, Nkour, Nux, Orderud, Pi zero, QuiteUnusual, Recent Runes, Snaxe920, Spag85, Spelemann, Stephan Schneider, Thenub314, Tomato86, Tully, Waldir, Wdcf, Withinfocus, 60 anonymous edits

Errors and Warnings *Source*: http://en.wikibooks.org/w/index.php?oldid=2093263 *Contributors*: Adrignola, Alejo2083, LR, Mwtoews, Nothing1212, Pi zero, Rogerbrent, Thenub314, Waldir, Wkdurfee, 15 anonymous edits

Title Creation *Source*: http://en.wikibooks.org/w/index.php?oldid=2086267 *Contributors*: Adrignola, Alejo2083, Anarchyboy, Cfailde, Derbeth, Ftravers, Infenwe, JECompton, Jomegat, Jonathan Webley, Mwtoews, Pi zero, Recent Runes, Sargas, Spirosdenaxas, Thenub314, Withinfocus, 23 anonymous edits

Page Layout *Source*: http://en.wikibooks.org/w/index.php?oldid=2164811 *Contributors*: Adouglass, Adrignola, Alejo2083, Derbeth, Drewbie, Herbythyme, Igjimh, JenVan, Jomegat, Jtwdog, LQST, Mwtoews, Neet, Pi zero, Rajkiran g, Sabalka, Spook, Tomato86, Vaffelkake, Waldir, Withinfocus, 56 anonymous edits

Formatting *Source*: http://en.wikibooks.org/w/index.php?oldid=2172169 *Contributors*: ABCD, Adrignola, Alejo2083, Andyr, C3l, Cdecoro, ChrisHodgesUK, ConditionalZenith, Crasshopper, Derbeth, EvanKroske, Ffangs, Fishpi, GPHemsley, Gmh04, Hannes Röst, Harrikoo, Henrybissonnette, Hjsb, Igjimh, Ish ishwar, Jonathan Webley, Jtwdog, Keplerspeed, Kovianyo, LR, Listdata, MartinSpacek, Mcld, Mouselb, Mwtoews, Neet, Neoriddle, Nixphoeni, PatrickDevlin21, Pi zero, Pstar, QuantumEleven, QuiteUnusual, Recent Runes, Robbiemorrison, Rror, Smobbl Bobbl, Tazquebec, Thenub314, ToematoeAdmn, Tomato86, TorfusPolymorphus, Tully, Waldir, Withinfocus, Yez, Zrisher, Zzo38, 157 anonymous edits

Fonts *Source*: http://en.wikibooks.org/w/index.php?oldid=2129425 *Contributors*: Derbeth, Ish ishwar, Jacho, Joaospam, Pi zero, Waldir, Wikieditoroftoday, 9 anonymous edits

List Structures *Source*: http://en.wikibooks.org/w/index.php?oldid=2147600 *Contributors*: Cerniagigante, Henrybissonnette, Kejia, 3 anonymous edits

Tables *Source*: http://en.wikibooks.org/w/index.php?oldid=2170248 *Contributors*: Abonnema, Adrignola, Alejo2083, Arided, Avila.gas, Benjaminevans82, Bumbulski, Byassine52, ChrisHodgesUK, Chuaprap, Collinpark, Crissov, Derbeth, Dilaudid, Dporter, Drewbie, Dubbaluga, Erylaos, Fmccown, Gallen01, Gelbukh, Grenouille, Helptry, HenrikMidtiby, Hosszuka, Igjimh, Ish ishwar, JV, Jevon, Jonathan Webley, Jotomicron, Jtwdog, Klusinyan, Maratonda, Mcld, Mwtoews, PAC2, Petter Strandmark, Pi zero, PsyberS, Quaristice, Rdg nz, Rogal, Sandrobt, Sargas, SciYann, Skou, SteveM82, Thenub314, Tomxlawson, Tweenk, Vesal, Willy james, Withinfocus, Xonqnopp, Yanuzz, Ysnikraz, 122 anonymous edits

Bibliography Management *Source*: http://en.wikibooks.org/w/index.php?oldid=2163726 *Contributors*: Alejo2083, Amamory, Blaisorblade, Braindrain0000, Derbeth, Drewbie, Eyliu, Helder.wiki, Igjimh, Jevon, John1923, Jtwdog, Kazkaskazkasako, Kevang, Kovianyo, Lbailey45, MartinSpacek, Mhue, Mwtoews, Nbrouard, Norbert.beckers, Oderbolz, Pi zero, Qwertyus, Rafaelgr, Rbonvall, Rondenaranja, Rossdub, Sandman10000, Sargas, Silca678, Spag85, Spelemann, Stephan Schneider, StevenJohnston, Tdomhan, Thenub314, TorfusPolymorphus, Waldir, Withinfocus, Xonqnopp, ZeroOne, 136 anonymous edits

Mathematics *Source*: http://en.wikibooks.org/w/index.php?oldid=2172341 *Contributors*: 3mta3, Adam majewski, Adrignola, Alejo2083, Ans, Asmeurer, Atiq ur Rehman, Avila.gas, Basenga, BiT, Bonuama, Cameronc, Clebell, Conrad.Irwin, Cícero, DavidMcKenzie, Derbeth, Dilaudid, Dncarley, Elliptic1, Fsart, Geminatea, Germanzs, Greenbreen, Hagindaz, Igjimh, Insaneinside, InverseHypercube, Jodi.a.schneider, Joe Schmedley, Jomegat, Jonathan Webley, Jtwdog, Juliusross, Justin W Smith, Karcih, Karthicknainar, Kejia, Krishnavedala, Krst, Kubieziel, Kwetal, Lucasreddinger, Marra, Mcld, Mecanismo, Merciadriluca, Mhue, Ms2ger, Mwtoews, Neet, Netheril96, Nigels, Pi zero, Prispartlow, Recent Runes, Robin, Sargas, Swift, Tgwizard, Tom.marcik, Tomato86, TomyDuby, Unco, Waldir, Winfree, Withinfocus, Wmheric, Xnn, 175 anonymous edits

Advanced Mathematics *Source*: http://en.wikibooks.org/w/index.php?oldid=2165501 *Contributors*: 3mta3, Adrignola, Arthurvogel, Avila.gas, BiT, ChrisHodgesUK, Inductiveload, Jonathan Webley, Kcho, Krst, Leyo, LinuxChristian, MagnusPI, Pi zero, Silca678, Simonjtyler, Thenub314, Tomato86, TorfusPolymorphus, 27 anonymous edits

Theorems *Source*: http://en.wikibooks.org/w/index.php?oldid=2016767 *Contributors*: 3mta3, Alejo2083, Derbeth, Juliabackhausen, Kovianyo, Mhue, Pi zero, Raylu, Tomato86, Tosha, 18 anonymous edits

Linguistics *Source*: http://en.wikibooks.org/w/index.php?oldid=2171951 *Contributors*: DmitriyZotikov, 1 anonymous edits

Labels and Cross-referencing *Source*: http://en.wikibooks.org/w/index.php?oldid=2171282 *Contributors*: Adrignola, Alejo2083, DavidMcKenzie, Derbeth, InverseHypercube, Jomegat, Kovianyo, MartinSpacek, Mcld, Pi zero, Sargas, TomyDuby, Waldir, 52 anonymous edits

Indexing *Source*: http://en.wikibooks.org/w/index.php?oldid=2168359 *Contributors*: Adrignola, Alejo2083, Dan Polansky, Derbeth, Itai, Kovianyo, Morelight, Mwtoews, Neet, Neoptolemus, Nsuwan, Petter Strandmark, Pi zero, Sargas, Semperos, Stephan Schneider, Tomato86, Tully, Uluboz, 12 anonymous edits

Algorithms and Pseudocode *Source*: http://en.wikibooks.org/w/index.php?oldid=2170991 *Contributors*: Adrignola, Alejo2083, Brevity, Dan Polansky, Debejyo, Derbeth, Fishpi, Gkc, JenVan, Maartenweyn, Mcld, Mrt doulaty, Nemti, Pi zero, Sanderd17, Sargas, Tauriel-1, Thefrankinator, Tomato86, Webinn, Writalnaie, ZeroOne, 51 anonymous edits

Importing Graphics *Source*: http://en.wikibooks.org/w/index.php?oldid=2168609 *Contributors*: 3mta3, Adrignola, Alejo2083, Bcmpinc, Bsander, Cengique, ConditionalZenith, Crasic, Dan Polansky, Danielstrong52, Derbeth, Dilaudid, Ffangs, Harrywt, Hippasus, Igjimh, Insaneinside, Jflycn, Jtwdog, KlausFoehl, MarSraM, Martin scharrer, Mathieu Perrin, Matthias M., Mwtoews, NavarroJ, Nemoniac, Pi zero, Piksi, QuiteUnusual, Snoopy67, Spag85, TWiStErRob, Tomato86, Tuka, Waldir, Withinfocus, Wn202, Wysinwygaa, ZeroOne, 99 anonymous edits

Creating Graphics *Source*: http://en.wikibooks.org/w/index.php?oldid=2163905 *Contributors*: 3mta3, Adrignola, Alejo2083, Atulya1988, Dan Polansky, Franzl aus tirol, Jacobrothstein, Jflycn, Jomegat, Jonathan Webley, Kenyon, Keplerspeed, Krisrose, Mwtoews, Niel.Bowerman, Pi zero, Pmillerrhodes, Rajkiran g, Rnddim, Simeon, Thefrankinator, Tomato86, Tosha, Unbitwise, Wxm29, Пика Пика, 28 anonymous edits

Floats, Figures and Captions *Source*: http://en.wikibooks.org/w/index.php?oldid=2167023 *Contributors*: Adrignola, Alejo2083, Basenga, Borgg, Caesura, DavidMcKenzie, Derbeth, Dilaudid, Hansfn, HenrikMidtiby, Hosszuka, Icc97, Igjimh, Janltx, Jer789, Joe Schmedley, Jtwdog, Martin von Wittich, Mcld, Mwtoews, Neet, Orderud, Pi zero, Rafopar, Robert Borkowski, Robin, Spag85, Tomato86, Tosha, Tully, Waldir, Wdcf, Withinfocus, Wxm29, Ypey, 87 anonymous edits

Colors *Source*: http://en.wikibooks.org/w/index.php?oldid=2167297 *Contributors*: Adrignola, Alejo2083, Benjaminevans82, ChristianGruen, Conighion, Dan Polansky, Gms, JackPotte, Kazkaskazkasako, Mietchen, PAC2, Pi zero, Sanderd17, Scorwin, Tomato86, Velociostrich, Waldir, White gecko, 23 anonymous edits

Glossary *Source*: http://en.wikibooks.org/w/index.php?oldid=2159369 *Contributors*: Dan Polansky, Pmlineditor, Redirect fixer, Tomato86, 10 anonymous edits

Letters *Source*: http://en.wikibooks.org/w/index.php?oldid=2160418 *Contributors*: Derbeth, E.lewis1, Edudobay, Garoth, GavinMcGimpsey, Gms, Jld, Neoptolemus, Pi zero, QuiteUnusual, Tomato86, 21 anonymous edits

Teacher's Corner *Source*: http://en.wikibooks.org/w/index.php?oldid=2134012 *Contributors*: PAC, PAC2, Tomato86, 10 anonymous edits

Presentations *Source*: http://en.wikibooks.org/w/index.php?oldid=2166892 *Contributors*: 3mta3, Avila.gas, Dan Polansky, Derbeth, Dreaven3, Eyliu, Flal, Grj23, Helptry, Ish ishwar, Jayk, Kovianyo, Neoriddle, NqpZ, PAC, PAC2, Pi zero, Ramac, Sanderd17, Sargas, Stuples, Xonqnopp, 23 anonymous edits

Hyperlinks *Source*: http://en.wikibooks.org/w/index.php?oldid=2157332 *Contributors*: Adrignola, Alejo2083, Bajrangkhichi96, Belteshazzar, Blacktrumpeter, Bytecrook, Calimo, Courcelles, Dan Polansky, Derbeth, Joeyboi, Jomegat, Juliabackhausen, Jwchong, Kevinfiesta, LaTeX, Lnkbuildingservices4u, MartinSpacek, Mimo, Modest Genius, Neet, Pamputt, PeterAllen, Pi zero, Prawojazdy, Qwertyus, Recent Runes, Roarbakk, SiriusB, Teles, Thietkeweb, Tomato86, Urhixidur, Waldir, Wdcf, Xonqnopp, 43 anonymous edits

Packages *Source*: http://en.wikibooks.org/w/index.php?oldid=2032595 *Contributors*: Adamcrume, Alejo2083, Derbeth, HJMills, HenrikMidtiby, Hosszuka, Javalenok, Jtwdog, Kovianyo, Kwetal, LR, Listdata, Mwtoews, Neet, Orderud, Paxinum, Pi zero, TFTD, Tomato86, Tully, Withinfocus, 17 anonymous edits

General Guidelines *Source*: http://en.wikibooks.org/w/index.php?oldid=2060835 *Contributors*: Alejo2083, Derbeth, Mike.lifeguard, Pi zero, 6 anonymous edits

Advanced Topics *Source*: http://en.wikibooks.org/w/index.php?oldid=2149203 *Contributors*: Adrignola, Alejo2083, Crasshopper, DavidMcKenzie, Everlong, Fongs, Jacobrothstein, Jguk, LR, MartinSpacek, Neet, Pi zero, Raphael Ackermann, Sargas, Tomato86, Waldir, Withinfocus, 29 anonymous edits

Customizing LaTeX *Source*: http://en.wikibooks.org/w/index.php?oldid=2155415 *Contributors*: Aadornellesf, Adrignola, Alejo2083, Go.pbam., Hosszuka, LR, Mwtoews, Pi zero, Risk, Tully, 32 anonymous edits

Multiple files *Source*: http://en.wikibooks.org/w/index.php?oldid=2160397 *Contributors*: Ediahist, MaBoehm, Neet, Neoriddle, Zyqqh, 4 anonymous edits

Collaborative Writing of LaTeX Documents *Source*: http://en.wikibooks.org/w/index.php?oldid=2169817 *Contributors*: Arnehe, Bamgooly, Derbeth, Glosser.ca, Hermine potter, Jason barrington, Jmcdon10, Keplerspeed, Kpym, Madskaddie, Pi zero, QuiteUnusual, Schaber, Tosha, 28 anonymous edits

Internationalization *Source*: http://en.wikibooks.org/w/index.php?oldid=2170653 *Contributors*: Adrignola, Alejo2083, Alzahrawi, BiT, Derbeth, Drevicko, Eudoxos, Harrikoo, Hroobjartr, ILubeMyCucumbers20, Jlrn, Jomegat, Kovianyo, Louabill, Louisix, Mijikenda, Pi zero, Piksi, Skarakoleva, Tomato86, 36 anonymous edits

Accents *Source*: http://en.wikibooks.org/w/index.php?oldid=2172175 *Contributors*: ChrisHodgesUK, Drevicko, Gms, Mwtoews, Pi zero, Silverpie, Steindani, Waldir, Wikieditoroftoday, Zvika, Zwiebelleder, Zxx117, 20 anonymous edits

Tips and Tricks *Source*: http://en.wikibooks.org/w/index.php?oldid=2152465 *Contributors*: Alejo2083, Basenga, Brendanarnold, Bumbulski, Derbeth, Filip Dominec, Jamoroch, Jguk, Mwtoews, Pi zero, Sargas, Tomato86, Towsonu2003, Vaffelkake, Winniehell, Withinfocus, 43 anonymous edits

Useful Measurement Macros *Source*: http://en.wikibooks.org/w/index.php?oldid=2027114 *Contributors*: Adrignola, Alejo2083, Dendik, Derbeth, Hokiehead, Igjimh, Panic2k4, Pi zero, Scruss, Withinfocus, Xonqnopp, 23 anonymous edits

Export To Other Formats *Source*: http://en.wikibooks.org/w/index.php?oldid=2149630 *Contributors*: Alejo2083, Bakken, Bpsullivan, Derbeth, Jomegat, Keplerspeed, Mwtoews, Pi zero, 20 anonymous edits

Command Glossary *Source*: http://en.wikibooks.org/w/index.php?oldid=2165928 *Contributors*: Adrignola, ChrisHodgesUK, Dan Polansky, Derbeth, Dmb, Igjimh, IrfanAli, Orderud, Paxinum, Pi zero, Pstar, Robert Horning, Spelemann, Stoettner, TomyDuby, Tuetschek, Withinfocus, Xeracles, 21 anonymous edits

Index *Source*: http://en.wikibooks.org/w/index.php?oldid=2165970 *Contributors*: Avila.gas, Dan Polansky, Derbeth, Ffangs, Jonathan Webley, Pi zero, Ramac, TomyDuby, 2 anonymous edits

Image Sources, Licenses and Contributors

Image:LaTeX_diagram.svg *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_diagram.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:Latex wikibook test title.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_wikibook_test_title.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:Latex_layout.svg *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_layout.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:quote1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Quote1.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:quote2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Quote2.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:quote4.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Quote4.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:Latex_quote_3.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_quote_3.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Thenub314

Image:Latex_quote_4.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_quote_4.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:LaTeX sloppypar.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_sloppypar.png *License*: Public Domain *Contributors*: Derbeth

Image:Latex example ligatures.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_ligatures.png *License*: GNU Free Documentation License *Contributors*: Tobias Oetiker

Image:emph.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Emph.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

File:Latex sizes table.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_sizes_table.png *License*: GNU Free Documentation License *Contributors*: Jtwdog at en.wikibooks

Image:Ammonium sulphate mhchem.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Ammonium_sulphate_mhchem.png *License*: GNU Free Documentation License *Contributors*: Mike.lifeguard, Mwtoews

Image:Latex dashes example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_dashes_example.png *License*: GNU Free Documentation License *Contributors*: Tobias Oetiker

Image:dashes.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Dashes.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:Latex example text dots.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_text_dots.png *License*: GNU Free Documentation License *Contributors*: Tobias Oetiker

Image:Latex ready-made strings.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_ready-made_strings.png *License*: GNU Free Documentation License *Contributors*: Tobias Oetiker

Image:symbols.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Symbols.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:LaTeX-dingbats.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-dingbats.png *License*: GNU Free Documentation License *Contributors*: Andrew Roberts

Image:verbatim.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Verbatim.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:alltt.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Alltt.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:LaTeX-footnote.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-footnote.png *License*: GNU Free Documentation License *Contributors*: Andrew Roberts

Image:LaTeX marginpar.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_marginpar.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Derbeth

Image:itemize.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Itemize.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:enum.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Enum.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:desc.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Desc.png *License*: GNU Free Documentation License *Contributors*: Derbeth, Jtwdog

Image:LaTeX desc-newline.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_desc-newline.png *License*: Public Domain *Contributors*: b:en:User:LRJulian Verdurmen

Image:nested.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Nested.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:Latex example paralist.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_paralist.png *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Alessio Damato 20:55, 24 August 2007 (UTC)

Image:basic_table1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Basic_table1.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:basic_table2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Basic_table2.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:basic_table3.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Basic_table3.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:basic_table4.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Basic_table4.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:Latex example tabular cline.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_tabular_cline.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex example wrapped table.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_wrapped_table.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Andy Roberts and Alessio Damato

Image:Latex example defining multiple columns.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_defining_multiple_columns.png *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Alessio Damato

Image:align.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Align.png *License*: GNU Free Documentation License *Contributors*: Original uploader was Jtwdog at en.wikibooks

Image:specifier1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Specifier1.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:specifier2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Specifier2.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:specifier3.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Specifier3.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:specifier4.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Specifier4.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:multicolumn.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Multicolumn.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:multirow.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Multirow.png *License*: GNU Free Documentation License *Contributors*: Jtwdog

Image:multirowandcolumnexample.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Multirowandcolumnexample.png *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*:

Image:Latex-tables-double-dichotomy-example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex-tables-double-dichotomy-example.png *License*: Public Domain *Contributors*: Arided

File:LaTeXAlternateRowTable.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeXAlternateRowTable.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Maratonda

Image:Partial-vertical-line-add.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Partial-vertical-line-add.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jevon

Image:Partial-vertical-line-remove.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Partial-vertical-line-remove.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jevon

Image:LaTeX TabWidth1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_TabWidth1.png *License*: Public Domain *Contributors*: Derbeth

Image:LaTeX TabWidth2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_TabWidth2.png *License*: Public Domain *Contributors*: Derbeth

Image:LaTeX TabXWidth1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_TabXWidth1.png *License*: Public Domain *Contributors*: Neteler

Image:LaTeX TabXWidth2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_TabXWidth2.png *License*: Public Domain *Contributors*: Neteler

Image:LaTeX animal table.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_animal_table.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Mwtoews

Image:LaTeX animal table with booktabs.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_animal_table_with_booktabs.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Mwtoews

Image:LaTeX bibliography plain.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_bibliography_plain.png *License*: Public Domain *Contributors*: Derbeth

Image:LaTeX bibliography abbrv.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_bibliography_abbrv.png *License*: Public Domain *Contributors*: Derbeth

Image:LaTeX bibliography alpha.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_bibliography_alpha.png *License*: Public Domain *Contributors*: Derbeth

Image:Jabref-2.2-screenshot.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Jabref-2.2-screenshot.png *License*: GNU Free Documentation License *Contributors*: Mwtoews, Ö

Image:BibDesk-1.3.10-screenshot.png *Source*: http://en.wikibooks.org/w/index.php?title=File:BibDesk-1.3.10-screenshot.png *License*: Public Domain *Contributors*: Mij

Image:LaTeX-xfrac-example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-xfrac-example.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:Latex_new_squareroot.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_new_squareroot.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:bordermatrix.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Bordermatrix.png *License*: Public Domain *Contributors*: Winfree

Image:LaTeX-smallmatrix.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-smallmatrix.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

File:LaTeX Dotsc.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_Dotsc.png *License*: Public Domain *Contributors*: Neet

File:LaTeX Dotsb.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_Dotsb.png *License*: Public Domain *Contributors*: Neet

File:LaTeX Dotsm.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_Dotsm.png *License*: Public Domain *Contributors*: Neet

File:LaTeX Dotsi.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_Dotsi.png *License*: Public Domain *Contributors*: Neet

File:LaTeX Dotso.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_Dotso.png *License*: Public Domain *Contributors*: Neet

Image:LaTeX-mathclap-example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-mathclap-example.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:LaTeX-mathtools-brackets.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-mathtools-brackets.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:LaTeX-mathtools-arrows.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-mathtools-arrows.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:LaTeX-mathtools-harpoons.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-mathtools-harpoons.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

File:LaTeX - Indented Equations.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_-_Indented_Equations.png *License*: Public Domain *Contributors*: Inductiveload

Image:LaTeX-displaybreak-in-math.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-displaybreak-in-math.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:LaTeX-boxed-equation.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-boxed-equation.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:LaTeX-boxed-formula-minipage.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-boxed-formula-minipage.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:Latex-Aboxed-example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex-Aboxed-example.png *License*: Creative Commons Zero *Contributors*: ChrisHodgesUK

Image:Latex-intertext.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex-intertext.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Tomato86

Image:Latex example referencing section.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_referencing_section.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:Latex example figure referencing.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_figure_referencing.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato 13:31, 12 January 2007 (UTC)

Image:Latex example math referencing.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_math_referencing.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

File:Latex-algorithmic-if-else.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex-algorithmic-if-else.png *License*: Public Domain *Contributors*: Nemti

File:LaTeX_program_package_example01.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_program_package_example01.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: MyName (Gkc (talk))

File:Latex_sample_code.PNG *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_sample_code.PNG *License*: GNU General Public License *Contributors*: LaTeX

Image:chick1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Chick1.png *License*: GNU Free Documentation License *Contributors*: Original uploader was Jtwdog at en.wikibooks

Image:chick2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Chick2.png *License*: GNU Free Documentation License *Contributors*: Original uploader was Jtwdog at en.wikibooks

Image:chick3.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Chick3.png *License*: GNU Free Documentation License *Contributors*: Original uploader was Jtwdog at en.wikibooks

Image:chick4.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Chick4.png *License*: GNU Free Documentation License *Contributors*: Original uploader was Jtwdog at en.wikibooks

Image:Latex_example_line_segments.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_line_segments.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_arrows.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_arrows.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_circles.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_circles.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_text_formulas.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_text_formulas.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_multiput.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_multiput.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_ovals.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_ovals.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_multiple_pics.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_multiple_pics.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_bezier.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_bezier.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_catenary.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_catenary.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_rapidity.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_rapidity.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_basic.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_basic.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_arrows_1.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_arrows_1.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_arrows_2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_arrows_2.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_arrows_3.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_arrows_3.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_arrows_labels.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_arrows_labels.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_inarrow_labels.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_inarrow_labels.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_arrow_list.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_arrow_list.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex_example_xypics_standard_arrow.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_standard_arrow.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex example xypics curved arrow.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_xypics_curved_arrow.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Chemfig_angles.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Chemfig_angles.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Chemfig_bonds.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Chemfig_bonds.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Methane_chemfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Methane_chemfig.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Skeletondiagram.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Skeletondiagram.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Skeletondiagram2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Skeletondiagram2.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Ring_chemfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Ring_chemfig.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Ring2_chemfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Ring2_chemfig.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Ring3_chemfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Ring3_chemfig.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Ring4_chemfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Ring4_chemfig.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Acetate-ion2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Acetate-ion2.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Acetate-ion.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Acetate-ion.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Pmillerrhodes

Image:Ion-example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Ion-example.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Edgar181, Jahobr, Pmillerrhodes

Image:Latex caption example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_caption_example.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:LaTeX figure caption with lof entry.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX_figure_caption_with_lof_entry.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Mwtoews

Image:Latex example sidecap.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_sidecap.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: User:Mwtoews

Image:Latex example wrapfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_wrapfig.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:Latex example wrapfig vspace.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_wrapfig_vspace.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:Latex example subfig.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_subfig.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:LaTeX-letter.png *Source*: http://en.wikibooks.org/w/index.php?title=File:LaTeX-letter.png *License*: GNU Free Documentation License *Contributors*: Derbeth

Image:Envelope.jpg *Source*: http://en.wikibooks.org/w/index.php?title=File:Envelope.jpg *License*: Public Domain *Contributors*: Jld

Image:Koma_env.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Koma_env.png *License*: GNU Free Documentation License *Contributors*: gms

Image:Frametitle_keyword_example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Frametitle_keyword_example.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Israel Buitron

Image:Latex_showkeys_example.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_showkeys_example.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

Image:Latex example box test.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_box_test.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex example box test 2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_box_test_2.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex example rule.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_rule.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:Latex example newenvironment.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Latex_example_newenvironment.png *License*: GNU Free Documentation License *Contributors*: Alessio Damato

Image:ESvn-texmf.png *Source*: http://en.wikibooks.org/w/index.php?title=File:ESvn-texmf.png *License*: unknown *Contributors*: Original uploader was Arnehe at en.wikibooks

Image:Kdiff3-modification.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Kdiff3-modification.png *License*: unknown *Contributors*: Original uploader was Arnehe at en.wikibooks

Image:JabRef-KeyPattern.png *Source*: http://en.wikibooks.org/w/index.php?title=File:JabRef-KeyPattern.png *License*: unknown *Contributors*: Original uploader was Arnehe at en.wikibooks

Image:JabRef-GeneralFields.png *Source*: http://en.wikibooks.org/w/index.php?title=File:JabRef-GeneralFields.png *License*: unknown *Contributors*: Original uploader was Arnehe at en.wikibooks

Image:JabRef-ExternalPrograms.png *Source*: http://en.wikibooks.org/w/index.php?title=File:JabRef-ExternalPrograms.png *License*: unknown *Contributors*: Original uploader was Arnehe at en.wikibooks

Image:Soliton_2nd_order.svg *Source*: http://en.wikibooks.org/w/index.php?title=File:Soliton_2nd_order.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Alessio Damato

License

Creative Commons Attribution-Share Alike 3.0 Unported
http://creativecommons.org/licenses/by-sa/3.0/