LaTeX Wikibook

Contents

Chapter 1

Getting Started

1.1 Introduction

1.1.1 What is TeX?

TeX is a low-level markup and programming language created by Donald Knuth to typeset documents attractively and consistently. 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. With the release of 8-bit character support in 1989, TeX development has been essentially frozen with only bug fixes released periodically. 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 over document structure and formatting makes it a powerful—and formidable—tool. TeX is renowned for being extremely stable, for running on many different kinds of computers, and for being virtually bug free. The version numbers of TeX are converging toward π , with a current version number of 3.1415926.

The name TeX is intended by its developer to be /'tɛx/, with the final consonant of *loch* or *Bach*. (Donald E. Knuth, *The TeXbook*) The letters of the name are meant to represent the capital Greek letters tau, epsilon, and chi, as TeX is an abbreviation of $\tau\acute{\epsilon}\chi\nu\eta$ (TEXNH – technē), Greek for both "art" and "craft", which is also the root word of *technical*. English speakers often pronounce it /'tɛk/, like the first syllable of *technical*.

Programming in TeX generally progresses along a very gradual learning curve, requiring a significant investment of time to build custom macros for text formatting. Fortunately, document preparation systems based on TeX, consisting of collections of pre-built macros, do exist. These pre-built macros are time saving, and automate certain repetitive tasks and help reduce user introduced errors; however, this convenience comes at the cost of complete design flexibility. One of the most popular macro packages is called **LaTeX**.

1.1.2 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. Within the typesetting system, its name is formatted as LaTeX.

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).

Since LaTeX comprises a group of TeX commands, LaTeX document processing is essentially programming. You create a text file in LaTeX markup, which LaTeX reads 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, or text flow for readability, as LaTeX takes care of them automatically.

- In LaTeX the document structure 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.

- Since the document source is plain text, tables, figures, equations, etc. can be generated programmatically with any language.

- You are forced to structure your documents correctly.

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, JPEG, etc.

1.1.3   Philosophy of use

Flexibility and modularity

One of the most frustrating things beginners and even advanced users might encounter using LaTeX is the lack of flexibility regarding the document design and layout. If you want to design your document in a very specific way, you may have trouble accomplishing this. Keep in mind that LaTeX *does* the formatting for you, and mostly the right way. If it is not exactly what you desired, then the LaTeX way is at least not worse, if not better. One way to look at it is that LaTeX is a bundle of macros for TeX that aims to carry out everything regarding document formatting, so that the writer only needs to care about content. If you really want flexibility, use plain TeX instead.

One solution to this dilemma is to make use of the modular possibilities of LaTeX. You can build your own macros, or use macros developed by others. You are likely not the first person to face some particular formatting problem, and someone who encountered a similar problem before may have published their solution as a package.

CTAN is a good place to find many resources regarding TeX and derivative packages. It is the first place where you should begin searching.

Questions and documentation

Besides internet resources being plentiful, the best documentation source remains the official manual for every specific package, and the reference documentation, i.e., the *TeXbook* by D. Knuth and *LaTeX: A document preparation system* by L. Lamport.

Therefore before rushing on your favorite web search engine, we really urge you to have a look at the package documentation that causes troubles. This official documentation is most commonly installed along your TeX distribution, or may be found on CTAN.

1.1.4   Terms regarding TeX

Document preparation systems

LaTeX is a document preparation system based on TeX. So the system is the combination of the language and the macros.

Distributions

TeX distributions are collections of packages and programs (compilers, fonts, and macro packages) that enable you to typeset without having to manually fetch files and configure things.

Engines

An engine is an executable that can turn your source code into a printable output format. The engine by itself only handles the syntax, it also needs to load fonts and macros to fully understand the source code and generate output properly. The engine will determine what kind of source code it can read, and what format it can output (usually DVI or PDF).

All in all, distributions are an easy way to install what you need to use the engines and the systems you want. Distributions usually target specific operating systems. You can use different systems on different engines, but sometimes there are restrictions. Code written for TeX, LaTeX or ConTeXt are (mostly) not compatible. Additionally, engine-specific code (like font for XeTeX) may not be compiled by every engine.

When searching for information on LaTeX, you might also stumble upon XeTeX, ConTeXt, LuaTeX or other names with a -TeX suffix. Let's recap most of the terms in this table.

1.1.5   What next?

In the next chapter we will proceed to the installation. Then we will compile our first LaTeX file.

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 Q&A,

- the TeX FAQ,

- and the TeXample.net Community.

1.2 Installation

If this is the first time you are trying out LaTeX, you don't even need to install anything. For quick testing purpose you may just create a user account with an online LaTeX editor and continue this tutorial in the next chapter. These websites offer collaboration capabilities while 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.

LaTeX is not a program by itself; it is a language. *Using LaTeX* requires a bunch of tools. Acquiring them manually would result in downloading and installing multiple programs in order to have a suitable computer system that can be used to create LaTeX output, such as PDFs. **TeX Distributions** help the user in this way, in that it is a single step installation process that provides (almost) everything.

At a minimum, you'll need a TeX distribution, a good text editor and a DVI or PDF viewer. More specifically, the basic requirement is to have a TeX compiler (which is used to generate output files from source), fonts, and the LaTeX macro set. Optional, and recommended installations include an attractive editor to write LaTeX source documents (this is probably where you will spend most of your time), and a bibliographic management program to manage references if you use them a lot.

1.2.1 Distributions

TeX and LaTeX are available for most computer platforms, since they were programmed to be very portable. They are most commonly installed using a distribution, such as teTeX, MiKTeX, or MacTeX. TeX distributions are collections of packages and programs (compilers, fonts, and macro packages) that enable you to typeset without having to manually fetch files and configure things. LaTeX is just a set of macro packages built for TeX.

The recommended distributions for each of the major operating systems are:

- TeX Live is a major TeX distribution for *BSD, GNU/Linux, Mac OS X and Windows.

- MiKTeX is a Windows-specific distribution.

- MacTeX is a Mac OS-specific distribution based on TeX Live.

These, however, do not necessarily include an editor. You might be interested in other programs that are not part of the distribution, which will help you in writing and preparing TeX and LaTeX files.

*BSD and GNU/Linux

In the past, the most common distribution used to be **teTeX**. As of May 2006 teTeX is no longer actively maintained and its former maintainer Thomas Esser recommended TeX Live as the replacement.[1]

The easy way to get TeX Live is to use the package manager or portage tree coming with your operating system. Usually it comes as several packages, with some of them being essential, other optional. The *core* TeX Live packages should be around 200-300 MB.

If your *BSD or GNU/Linux distribution does not have the TeX Live packages, you should report a wish to the bug tracking system. In that case you will need to download TeX Live yourself and run the installer by hand.

You may wish to install the content of TeX Live more selectively. See below.

Mac OS X

Mac OS X users may use MacTeX, a TeX Live-based distribution supporting TeX, LaTeX, AMSTeX, ConTeXt, XeTeX and many other core packages. Download MacTeX.mpkg.zip on the MacTeX page, unzip it and follow the instructions. Further information for Mac OS X users can be found on the TeX on Mac OS X Wiki.

Since Mac OS X is also a Unix-based system, TeX Live is naturally available through MacPorts and Fink. Homebrew users should use the official MacTeX installer because of the unique directory structure used by TeX Live. Further information for Mac OS X users can be found on the TeX on Mac OS X Wiki.

Microsoft Windows

Microsoft Windows users can install MiKTeX onto their computer. It has an easy installer that takes care of setting up the environment and downloading core packages. This distribution has advanced features, such as automatic installation of packages, and simple interfaces to modify settings, such as default paper sizes.

There is also a port of TeX Live available for Windows.

1.2.2 Custom installation with TeX Live

This section targets users who want fine-grained control over their TeX distribution, like an installation with a minimum of disk space usage. If it is none of your concern, you may want to jump to the next section.

Picky users may wish to have more control over their installation. Common distributions might be tedious for the user caring about disk space. In fact, MikTeX and Mac-TeX and packaged TeX Live features hundreds of LaTeX packages, most of them which you will never use. Most Unix with a package manager will offer TeX Live as a set of several big packages, and you often have to install 300–400 MB for a functional system.

TeX Live features a manual installation with a lot of possible customizations. You can get the network installer at tug.org. This installer allows you to select precisely the packages you want to install. As a result, you may have everything you need for less than 100 MB. TeX Live is then managed through its own package manager, *tlmgr*. It will let you configure the distributions, install or remove extra packages and so on.

You will need a Unix-based operating system for the following. Mac OS X, GNU/Linux or *BSD are fine. It may work for Windows but the process must be quite different.

TeX Live groups features and packages into different concepts:

- *Collections* are groups of packages that can always be installed individually, except for the *Essential programs and files* collection. You can install collections at any time.

- *Installation Schemes* group collections and packages. Schemes can only be used at installation time. You can select only one scheme at a time.

Minimal installation

We will give you general guidelines to install a minimal TeX distribution (*i.e.*, only for plain TeX).

1. Download the installer at http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz and extract it to a temporary folder.

2. Open a terminal in the extracted folder and log in as root.

3. Change the umask permissions to 022 (user read/write/execute, group/others read/execute only) to make sure other users will have read-only access to the installed distribution.

umask 022

1. Launch install-tl.

2. Select the *minimal scheme (plain only)*.

3. You may want to change the directory options. For example you may want to hide your personal macro folder which is located at TEXMFHOME. It is ~/texmf by default. Replace it by ~/.texmf to hide it.

4. Now the options:

   (a) **use letter size instead of A4 by default:** mostly for users from the USA.

   (b) **execution of restricted list of programs:** it is recommended to select it for security reasons. Otherwise it allows the TeX engines to call any external program. You may still configure the list afterwards.

   (c) **create format files:** targetting a minimal disk space, the best choice depends on whether there is only one user on the system, then deselecting it is better, otherwise select it. From the help menu: *"If this option is set, format files are created for system-wide use by the installer. Otherwise they will be created automatically when needed. In the latter case format files are stored in user's directory trees and in some cases have to be re-created when new packages are installed."*

   (d) **install font/macro doc tree:** useful if you are a developer, but very space consuming. Turn it off if you want to save space.

   (e) **install font/macro source tree:** same as above.

   (f) Symlinks are fine by default, change it if you know what you are doing.

5. Select portable installation if you install the distribution to an optical disc, or any kind of external media. Leave to default for a traditional installation on the system hard drive.

At this point it should display

1 collections out of 85, disk space required: 40 MB

or a similar space usage.

You can now proceed to installation: *start installation to hard disk*.

Don't forget to add the binaries to your PATH as it's noticed at the end of the installation procedure.

First test

In a terminal write

$ tex '\empty Hello world!\bye' $ pdftex '\empty Hello world!\bye'

You should get a DVI or a PDF file accordingly.

Configuration

Formerly, TeX distributions used to be configured with the texconfig tool from the teTeX distribution. TeX Live still features this tool, but recommends using its own tool instead: tlmgr. Note that as of January 2013 not all texconfig features are implemented by tlmgr. Only use texconfig when you cannot do what you want with tlmgr.

List current installation options:

tlmgr option

You can change the install options:

tlmgr option srcfiles 1 tlmgr option docfiles 0 tlmgr paper letter

See the TLMGR(1) man page for more details on its usage. If you did not install the documents as told previously, you can still access the tlmgr man page with

tlmgr help

Installing LaTeX

Now we have a running plain TeX environment, let's install the base packages for LaTeX.

tlmgr install latex latex-bin latexconfig latex-fonts

In this case you can omit latexconfig latex-fonts as they are auto-resolved dependencies to LaTeX. Note that tlmgr resolves some dependencies, but not all. You may need to install dependencies manually. Thankfully this is rarely too cumbersome.

Other interesting packages:

tlmgr install amsmath babel carlisle ec geometry graphics hyperref lm marvosym oberdiek parskip pdftex-def url

If you installed a package you do not need anymore, use

tlmgr remove <package>

Hyphenation

If you are using Babel for non-English documents, you need to install the hyphenation patterns for every language you are going to use. They are all packaged individually. For instance, use

tlmgr install hyphen-{finnish,sanskrit}

for finnish and sanskrit hyphenation patterns.

Note that if you have been using another TeX distribution beforehand, you may still have hyphenation cache stored in you home folder. You need to remove it so that the new packages are taken into account. The TeX Live cache is usually stored in the ~/.texliveYYYY folder (YYYY stands for the year). You may safely remove this folder as it contains only generated data. TeX compilers will re-generate the cache accordingly on next compilation.

Uninstallation

By default TeX Live will install in /usr/local/texlive. The distribution is quite proper as it will not write any file outside its folder, except for the cache (like font cache, hyphenation patters, etc.). By default,

- the system cache goes in /var/lib/texmf;
- the user cache goes in ~/.texliveYYYY.

Therefore TeX Live can be installed and uninstalled safely by removing the aforementioned folders.

Still, TeX Live provides a more convenient way to do this:

tlmgr uninstall

You may still have to wipe out the folders if you put untracked files in them.

1.2.3 Editors

TeX and LaTeX source documents (as well as related files) are all text files, and can be opened and modified in almost any text editor. You should use a text editor (e.g. Notepad), not a word processor (Word, OpenOffice). Dedicated LaTeX editors are more useful than generic plain text editors, because they usually have autocompletion of commands, spell and error checking and handy macros.

Cross-platform

BaKoMa TeX   BaKoMa TeX is an editor for Windows and Mac OS 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. You can take an evaluation copy for 28 days.

Emacs   Emacs is a general purpose, extensible text processing system. Advanced users can program it (in elisp) to make Emacs the best LaTeX environment that will fit their needs. In turn beginners may prefer using it in combination with AUCTeX and Reftex (extensions that may be installed into the Emacs program). Depending on your configuration, Emacs can provide a complete LaTeX editing environment with auto-completion, spell-checking, a complete set of keyboard shortcuts, table of contents view, document preview and many other features.

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

Gummi   Gummi is a LaTeX editor for Linux, which compiles the output of pdflatex in realtime and shows it on the right half of the screen[2].
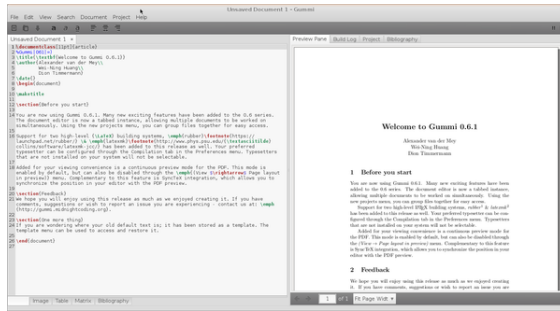
Screenshot of Gummi.
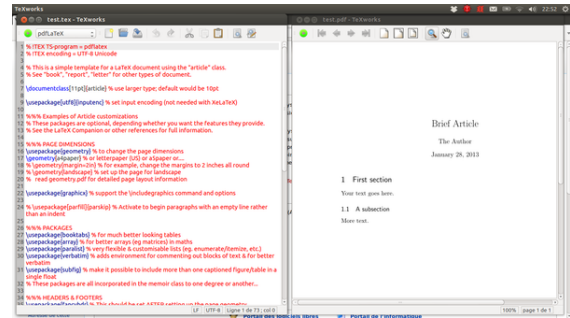
Screenshot of TeXworks on Ubuntu 12.10.
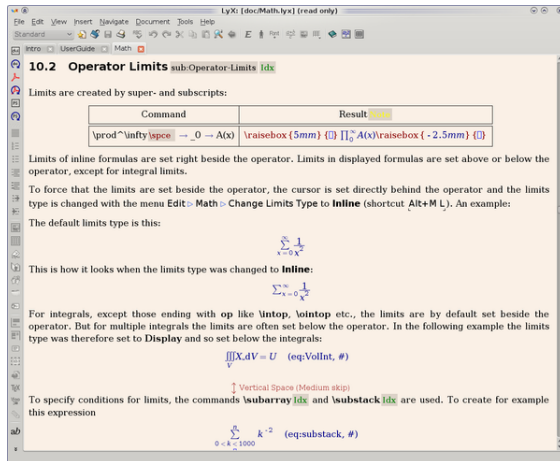
LyX1.6.3

LyX   LyX is a popular document preparation system for Windows, Linux and Mac OS. It provides a graphical interface to LaTeX, including several popular packages. 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. LyX calls this a What You See Is What You Mean (WYSIWYM) approach.[3]

LyX saves its documents in their own markup, and generates LaTeX code based on this. The user is mostly isolated from the LaTeX code and not in complete control of it, and as such LyX is not a normal LaTeX editor. However, as LaTeX is underlying system, knowledge of how that works is useful also for a LyX user. In addition, if one wants to do something that is not supported in the GUI, using LaTeX code may be required.

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

TeXstudio   TeXstudio is a cross-platform open source LaTeX editor forked from Texmaker.

TeXworks   TeXworks 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.

Vim   Vim is another general purpose text editor for a wide variety of platforms including UNIX, Mac OS X and Windows. A variety of extensions exist including LaTeX Box and Vim-LaTeX.

*BSD and GNU/Linux-only
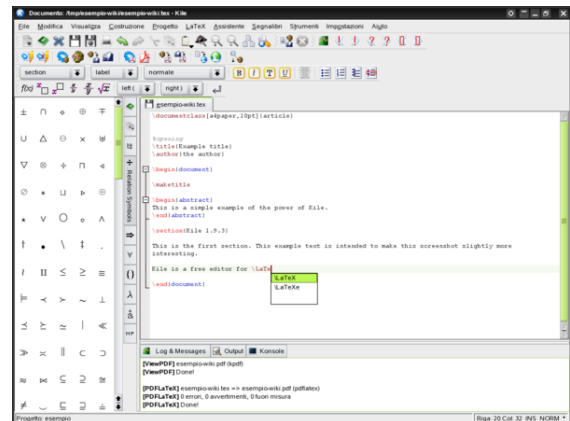
Screenshot of Kile.

Kile   Kile is a LaTeX editor for KDE (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.

LaTeXila   LaTeXila is another text editor for Linux

(Gnome).

Mac OS X-only

TeXShop   TeXShop is a TeXworks-like editor and pre-viewer for Mac OS that is bundled with the MacTeX distribution. It uses multiple windows, one for editing the source, one for the preview, and one as a console for error messages. It offers one-click updating of the preview and allows easy crossfinding between the code and the preview by using CMD-click.

TeXnicle   TeXnicle is a free editor for Mac OS that includes the ability to perform live updates. It includes a code library for the swift insertion of code and the ability to execute detailed word counts on documents. It also performs code highlighting and the editing window is customisable, permitting the user to select the font, colour, background colour of the editing environment. It is in active development.

Archimedes   Archimedes is an easy-to-use LaTeX and Markdown editor designed from the ground up for Mac OS X. It includes a built-in LaTeX library, code completion support, live previews, macro support, integration with sharing services, and PDF and HTML export options. Archimedes's Magic Type feature lets users insert mathematical symbols just by drawing them on their MacBook's trackpad or Magic Trackpad.

Texpad   Texpad is an integrated editor and viewer for Mac OS with a companion app for iOS devices. Similar to TeXShop, Texpad requires a working MacTeX distribution to function, however it can also support other distributions side-by-side with MacTex. It offers numerous features including templates, outline viewing, auto-completion, spell checking, customizable syntax highlighting, to-do list integration, code snippets, Markdown integration, multi-lingual support, and a Mac OS native user interface. Although Texpad offers a free evaluation period, the unlocked version is a paid download.

Windows-only

LEd   LEd

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

WinEdt   WinEdt is a powerful and versatile text editor with strong predisposition towards creation of LaTeX/TeX documents for Windows. It has been designed

and configured to integrate with TeX Systems such as MiTeX or TeX Live. Its built-in macro helps in compiling the LaTeX source to the WYSIWYG-like DVI or PDF or PS and also in exporting the document to other mark-up languages as HTML or XML.

WinShell   WinShell

Online solutions

To get started without needing to install anything, you can use a web-hosted service featuring a full TeX distribution and a web LaTeX editor.

- Authorea is an integrated online framework for the creation of technical documents in collaboration. Authorea's frontend allows you to enter text in LaTeX or Markdown, as well as figures, and equations (in LaTeX or MathML). Authorea's versioning control system is entirely based on Git (every article is a Git repository).

- Overleaf is a secure, easy to use online LaTeX editor with integrated rapid preview - like EtherPad for LaTeX. Start writing with one click (no signup required) and share the link. It supports real time preview, figures, bibliographies and custom styles.

- publications.li is a real-time collaborative LaTeX editor running the open-source editor \BlueLatex.

- ShareLaTeX.com is a secure cloud-based LaTeX editor offering unlimited free projects. Premium accounts are available for extra features such as collaborative editing, version control and Dropbox integration.

- SimpleLaTeX is an online editor and previewer for short LaTeX notes, which can be optionally cached or shared. Previews are available in SVG, PNG, and PDF. It also includes a simple GUI for editing tables.

- Verbosus 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.

1.2.4   Bibliography management

Bibliography files (*.bib) are most easily edited and modified using a management system. These graphical user interfaces all feature a database form, where information is entered for each reference item, and the resulting text file can be used directly by BibTeX.

Cross-platform
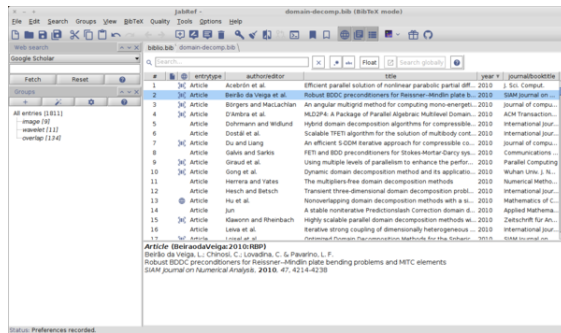
Screenshot of JabRef.

- JabRef

- Mendeley

- Zotero

Mac OS X-only
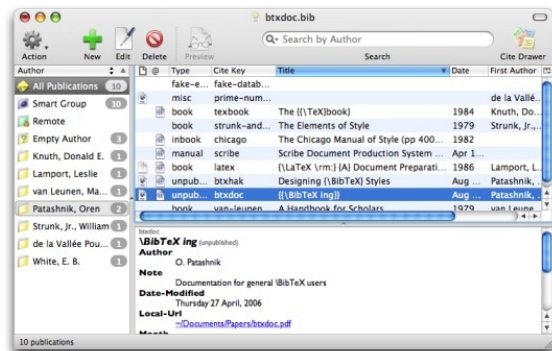
Screenshot of BibDesk

- BibDesk is a bibliography manager based on a Bib-TeX file. It imports references from the internet and makes it easy to organize references using tags and categories[4].

1.2.5   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 any PDF viewer.

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.

Here follows a list of various PDF viewers.

- PDF.js (built-in modern browsers)

- Evince (Linux GNOME)

- Foxit (Windows)

- Okular (Linux KDE)

- Preview (built-in Mac OS X)

- Skim (Mac OS X)

- Sumatra PDF (Windows)

- Xpdf (Linux)

- Zathura (Linux)

1.2.6   Tables and graphics tools

LaTeX is a document preparation system, it does not aim at being a spreadsheet tool nor a vector graphics tool.

If LaTeX can render beautiful tables in a dynamic and flexible manner, it will not handle the handy features you could get with a spreadsheet like dynamic cells and calculus. Other tools are better at that. The ideal solution is to combine the strength of both tools: build your dynamic table with a spreadsheet, and export it to LaTeX to get a beautiful table seamlessly integrated to your document. See Tables for more details.

The graphics topic is a bit different since it is possible to write procedural graphics from within your LaTeX document. Procedural graphics produce state-of-the-art results that integrates perfectly to LaTeX (*e.g.* no font change), but have a steep learning curve and require a lot of time to draw.

For easier and quicker drawings, you may want to use a WYSIWYG tool and export the result to a vector format like PDF. The drawback is that it will contrast in style with the rest of your document (font, size, etc.). Some tools have the capability to export to LaTeX, which will partially solve this issue. See Importing Graphics for more details.

1.2.7   References

[1] teTeX Home Page (Retrieved January 31, 2007)

[2] Gummi

[3] LyX

[4] BibDesk

1.3 Installing Extra 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. The directory name "texmf" stands for "TEX and METAFONT". To find out what other packages are available and what they do, you should use the CTAN search page 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. There are two main file types: class files with .cls extension, and style files with .sty extension. 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. You can 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 a configurable 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.

Most LaTeX installations come with a large set of pre-installed style packages, so you can use the package manager of the TeX distribution or the one on your system to manage them. See the automatic installation. But many more are available on the net. The main place to look for style packages on the Internet is CTAN. Once you have identified a package you need that is not in your distribution, use the indexes on any CTAN server to find the package you need and the directory where it can be downloaded from. See the manual installation.

1.3.1 Automatic installation

If on an operating system with a package manager or a portage tree, you can often find packages in repositories.

With MikTeX there is a package manager that allows you to pick the package you want individually. As a convenient feature, upon the compilation of a file requiring non-installed packages, MikTeX will automatically prompt to install the missing ones.

With TeX Live, it is common to have the distribution packed into a few big packages. For example, to install something related to internationalization, you might have to install a package like texlive-lang. With TeX Live manually installed, use tlmgr to manage packages individually.

tlmgr install <package1> <package2> ... tlmgr remove <package1> <package2> ...

The use of tlmgr is covered in the Installation chapter.

If you cannot find the wanted package with any of the previous methods, see the manual installation.

1.3.2 Manual installation

Downloading packages

What you need to look for is usually two files, one ending in .dtx and the other in .ins. The first is a DOCTeX file, which combines the package program and its documentation in a single file. The second is the installation routine (much smaller). You must always download both files. If the two files are not there, it means one of two things:

- *Either* the package is part of a much larger bundle which you shouldn't normally update unless you change LaTeXversion of LaTeX;

- *or* it's an older or relatively simple package written by an author who did not use a .dtx file.

Download the package files to a temporary directory. There will often be a readme.txt with a brief description of the package. You should of course read this file first.

Installing a package

There are five steps to installing a LaTeX package. (These steps can also be used on the pieces of a complicated package you wrote yourself; in this case, skip straight to Step 3.)

1. **Extract the files** Run LaTeX on the .ins file. That is, open the file in your editor and process it as if it were a LaTeX document (which it is), or if you prefer, type latex followed by the .ins filename in a command window in your temporary directory. This will extract all the files needed from the .dtx file (which is why you must have both of them present in the temporary directory). Note down or print the names of the files created if there are a lot of them (read the log file if you want to see their names again).

2. **Create the documentation** Run LaTeX on the .dtx file. You might need to run it twice or more, to get the cross-references right (just like any other LaTeX document). This will create a .dvi file of documentation explaining what the package is for and how to use it. If you prefer to create PDF then run pdfLaTeX instead. If you created a .idx as well, it means that the document contains an index, too. If you want the index to be created properly, follow the steps in the indexing section. Sometimes you will see that a .glo (glossary) file has been produced. Run the following command instead:

makeindex -s gglo.ist -o name.gls name.glo

3. **Install the files** While the documentation is printing, move or copy the package files from your temporary directory to the right place[s] in your TeX local installation directory tree. Packages installed by hand should always be placed in your "local" directory tree, *not* in the directory tree containing all the pre-installed packages. This is done to *a)* prevent your new package accidentally overwriting files in the main TeX directories; and *b)* avoid your newly-installed files being overwritten when you next update your version of TeX.

For a TDS(TeX Directory Structure)-conformant system, your "local installation directory tree" is a folder and its subfolders. The outermost folder should probably be called texmf-local/ or texmf/. Its location depends on your system:

- MacTeX:
  Users/*username*/Library/texmf/.
- Unix-type systems: Usually ~/texmf/.
- MikTeX: Your local directory tree can be any folder you like, as long as you then register it as a user-managed texmf directory (see http://docs.miktex.org/manual/localadditions.html#id573803)

The "right place" sometimes causes confusion, especially if your TeX installation is old or does not conform to the TeX Directory Structure(TDS). For a TDS-conformant system, the "right place" for a LaTeX .sty file is a suitably-named subdirectory of texmf/tex/latex/. "Suitably-named" means sensible and meaningful (and probably short). For a package like paralist, for example, I'd call the directory texmf/tex/latex/paralist.

Often there is just a .sty file to move, but in the case of complex packages there may be more, and they may belong in different locations. For example, new BibTeX packages or font packages will typically have several files to install. This is why it is a good idea to create a subdirectory for the package rather than dump the files into misc along with other unrelated stuff. If there are configuration or other files, read the documentation to find out if there is a special or preferred location to move them to.

For most fonts on CTAN, the *foundry* is public.

4. **Update your index** Finally, run your TeX indexer program to update the package database. This program comes with every modern version of TeX and has various names depending on the LaTeX distribution you use. (Read the documentation that came with your installation to find out which it is, or consult http://www.tug.org/fonts/fontinstall.html#fndb):

- teTeX, TeX Live, fpTeX: texhash
- web2c: mktexlsr

- MacTeX: MacTeX appears to do this for you.
- MikTeX: initexmf --update-fndb (or use the GUI)
- MiKTeX 2.7 or later versions, installed on Windows XP through Windows 7: Start -> All Programs -> MikTex -> Settings. In Windows 8 use the keyword *Settings* and choose the option of *Settings* with the MiKTex logo. In *Settings* menu choose the first tab and click on *Refresh FNDB*-button (MikTex will then check the Program Files directory and update the list of File Name DataBase). After that just verify by clicking 'OK'.

5. **Update font maps** If your package installed any TrueType or Type 1 fonts, you need to update the font mapping files *in addition* to updating the index. Your package author should have included a .map file for the fonts. The map updating program is usually some variant on updmap, depending on your distribution:

- TeX Live and MacTeX: updmap --enable Map=*mapfile*.map (if you installed the files in a personal tree) or updmap-sys --enable Map=*mapfile*.map (if you installed the files in a system directory).
- MikTeX: Run initexmf --edit-config-file updmap, add the line "Map *mapfile*.map to the file that opens, then run initexmf --mkmaps.

See http://www.tug.org/fonts/fontinstall.html.

The reason this process has not been automated widely is that there are still thousands of installations which do not conform to the TDS, such as old shared Unix systems and some Microsoft Windows systems, so there is no way for an installation program to guess where to put the files: you have to know this. There are also systems where the owner, user, or installer has chosen not to follow the recommended TDS directory structure, or is unable to do so for political or security reasons (such as a shared system where the user cannot write to a protected directory). The reason for having the texmf-local directory (called texmf.local on some systems) is to provide a place for local modifications or personal updates, especially if you are a user on a shared or managed system (Unix, Linux, VMS, Windows NT/2000/XP, etc.) where you may not have write-access to the main TeX installation directory tree. You can also have a personal texmf subdirectory in your own login directory. Your installation must be configured to look in these directories first, however, so that any updates to standard packages will be found there before the superseded copies in the main texmf tree. All modern TeX installations should do this anyway, but if not, you can edit texmf/web2c/texmf.cnf yourself.

1.3.3 Checking package status

The universal way to check if a file is available to TeX compilers is the command-line tool kpsewhich.

$ kpsewhich tikz /usr/local/texlive/2012/texmf-dist/tex/plain/pgf/frontendlayer/tikz.tex

kpsewhich will actually search for files only, not for packages. It returns the path to the file. For more details on a specific package use the command-line tool tlmgr (TeX Live only):

tlmgr info <package>

The tlmgr tool has lot more options. To consult the documentation:

tlmgr help

1.3.4 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:

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.

You can usually automatically open any installed package documentation with the *texdoc* command:

texdoc <package-name>

1.3.5 External resources

The best way to look for LaTeX packages is the already mentioned CTAN: Search. Additional resources form The TeX Catalogue Online:

- Alphabetic catalogue
- With brief descriptions

- Topical catalogue with packages sorted systematically
- Hierarchical mirroring the CTAN folder hierarchy

1.3.6 See Also

- LaTeX/Package Reference

1.4 Basics

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 will first have a look at the LaTeX syntax.
- We will create our first LaTeX document.
- Then we will take you through how to feed this file through the LaTeX system to produce quality output, such as postscript or PDF.
- Finally we will have a look at the file names and types.

1.4.1 The LaTeX syntax

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 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.

A minimal example looks something like the following (the commands will be explained later):

Spaces

The LaTeX compiler normalises whitespace so that whitespace characters, such as [space] or [tab], are treated uniformly as "space": several consecutive "spaces" are treated as one, "space" opening a line is generally ignored, and a single line break also yields "space". A double line break (an empty line), however, defines the end of a paragraph; multiple empty lines are also treated as the end of a paragraph. An example of applying these rules is presented below: the left-hand side shows the

user's input (.tex), while the right-hand side depicts the rendered output (.dvi/.pdf/.ps).

Reserved 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:

In some circumstances, the bracket characters [and] can also be considered as reserved characters, as they are used to give optional parameters to some commands. If you want to print these directly after some command, like in this situation: \command [text] it will fail, as [text] will be considered as an option given to \command. You can achieve the correct output this way: \command {} [text].

The backslash character \ *cannot* 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 commands \~ and \^ produce respectively a tilde and a hat which is placed over the next letter. For example \~n gives ñ. That's why you need braces to specify there is no letter as argument. You can also use \textasciitilde and \textasciicircum to enter these characters; or other commands .

If you want to insert text that might contain several particular symbols (such as URIs), you can consider using the \verb command, which will be discussed later in the section on formatting. For source code, see Source Code Listings

The 'less than' (<) and 'greater than' (>) characters are the only visible ASCII characters (not reserved) that will not print correctly. See Special Characters for an explanation and a workaround.

Non-ASCII characters (*e.g.* accents, diacritics) can be typed in directly for most cases. However you must *configure* the document appropriately. The other symbols and many more can be printed with special commands as in mathematical formulae or as accents. We will tackle this issue in Special Characters.

LaTeX groups

Sometimes a certain state shall be kept local, *i.e.* limiting its scope. This can be done by enclosing the part to be changed locally in curly braces. In certain occasions, using braces won't be possible. LaTeX provides \bgroup and \egroup to begin and end a group, respectively.

Environments form an implicit group.

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:

Between the \begin and the \end you can put other commands and nested environments. The internal mechanism of environments defines a group, which makes its usage safe (no influence on the other parts of the document). 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.

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:

Most standard LaTeX commands have a *switch* equivalent. Switches have no arguments but apply on the rest of the scope, *i.e.* the current group or environment. A switch should (almost) never be called outside of any scope, otherwise it will apply on the rest of the document.

Example:

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.

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

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

1.4.2 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.

- Save your file as hello.tex.

When picking a name for your file, make sure it bears a .tex extension.

What does it all mean?

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, provided there is no white space before the percent sign.

1.4.3 Compilation

Compilation process

The general concept is to transform a plain text document into a publishable format, mostly a DVI, PS or PDF file. This process is called *compilation*, which is done by an executable file called a *compiler*.

There are two main compilers.

- tex compiler reads a TeX .tex file and creates a .dvi.

- pdftex compiler reads a TeX .tex file and creates a .pdf.

These compilers are basically used to compile Plain TeX, not LaTeX. There is no such LaTeX compiler since LaTeX is just a bunch of macros for TeX. However, there are two executables related to the previous compilers:

- latex executable calls tex with LaTeX initialization files, reads a LaTeX .tex file and creates a .dvi

- pdflatex executable calls pdftex with LaTeX initialization files, reads a LaTeX .tex file and creates a .pdf

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.

As a matter of fact, following your operating system latex and pdflatex are simple scripts or symbolic links.

Most of the programs should be already within your LaTeX distribution; the others come with Ghostscript, which is a free and multi-platform software as well. Here are common programs you expect to find in any LaTeX distribution:

- dvi2ps converts the .dvi file to .ps (postscript).

- dvi2pdf converts the .dvi file to .pdf (dvi2pdfm is an improved version).

and with Ghostscript:

- ps2pdf and pdf2ps converts the .ps file to .pdf and vice-versa.

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 features of your document. Therefore, 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

pdftex   TEX   tex

pdflatex   LATEX   latex

Source formats

Output formats

PDF   ps2pdf / pdf2ps   PostScript   dvi2ps   DVI

(pdf, png, jpg)   (eps)

dvipdfm
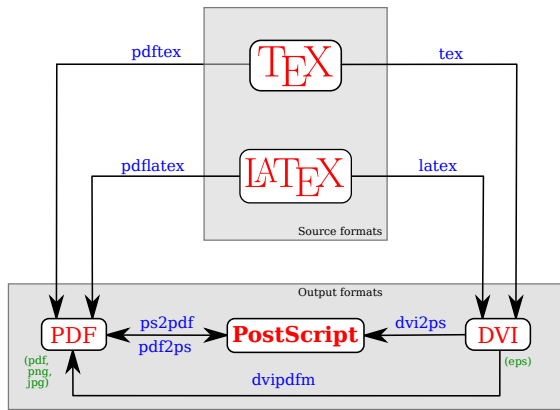
a LaTeX source, the best way is to use only *latex* for a DVI output or *pdflatex* for a PDF output, converting to PostScript only when it is necessary to print the document.

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

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. 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. 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. 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.)

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.

Autobuild Systems

Compiling using only the latex binary can be quite tricky as soon as you start working on more complex documents

as previously stated. A number of programs exist to automatically read in a TeX document and run the appropriate compilers the appropriate number of times. For example, latexmk can generate a PDF from most TeX files simply:

$ latexmk -pdf file.tex

Note that most editors will take care of it for you.

Compressed PDF

For a PDF output, you may have noticed that the output PDF file is not always the same size depending on the engine you used to compile the file. So latex → dvips → ps2pdf will usually be much smaller than pdflatex. If you want pdflatex features along with a small output file size, you can use the Ghostscript command:

$ gs -dBATCH -dNOPAUSE -q -sDEVICE=pdfwrite -sOutputFile="Compressed.pdf" "Original.pdf"

1.4.4   Files

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 only one full point or period (.) to separate the file extension (somewhat similar to the conventions for a good 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.

Ancillary files

The TeX compilers are single-pass processes. It means that there is no way for a compiler to *jump* around the document, which would be useful for the table of contents and references. Indeed the compiler cannot guess at which page a specific section is going to be printed, so when the table of contents is printed before the upcoming sections, it cannot set the page numbers.

To circumvent this issue, many LaTeX commands which need to *jump* use ancillary files which usually have the same file name as the current document but a different extension. It stores temporary data into these files and use them for the next compilation. So to have an up-to-date table of contents, you need to compile the document twice. There is no need to re-compile if no section moved.

For example, the temporary file for the table of contents data is filename.toc.

None of these files contains unrecoverable information. It means you can delete them safely, compiling will regenerate them automatically.

When you work with various capabilities of LaTeX (index, glossaries, bibliographies, etc.) 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:

1.4.5   And what now?

Common Elements

See Document Structure and the **Common Elements** part for all the common features that belong to every type of document.

Non-English documents and special characters

LaTeX has some nice features for most languages in the world. You can tell LaTeX to follow typography rules of the target language, ease special characters input, and so on. See Special Characters and Internationalization.

Modular document

See Modular Documents for good recommendations about the way to organize big projects into multiple files.

Questions and Issues

We highly urge you to read the FAQ if you have issues about basic features, or if you want to read essential recommendations. For the more specific questions and issues, refer to the Tips and Tricks page.

Macros for the utmost efficiency

The full power of LaTeX resides in macros. They make your documents very dynamic and flexible. See the dedicated part.

Working in a team

See chapter Collaborative Writing of LaTeX Documents.

Chapter 2

Common Elements

2.1 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 hierarchical constructs, including chapters, sections, subsections and paragraphs.

2.1.1 Global structure

When LaTeX processes an input file, it expects it to follow a certain structure. Thus every input file must contain the commands

The area between \documentclass{...} and \begin{document} is called the *preamble*. It normally contains commands that affect the entire document.

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

2.1.2 Preamble

Document classes

When processing an input file, LaTeX needs to know the type of document the author wants to create. This is specified with the \documentclass command. It is recommended to put this declaration at the very beginning.

Here, class specifies the type of document to be created. The LaTeX distribution provides additional classes for other documents, including letters and slides. 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. 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

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:

The standard document classes that are a part of LaTeX are built to be fairly generic, which is why they have a lot of options in common. Other classes may have different options (or none at all). Normally, third party classes come with some documentation to let you know. The most common options for the standard document classes are listed in the following table:

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

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. Some packages come with the LaTeX base distribution. Others are provided separately. Modern TeX distributions come with a large

number of packages pre-installed. The command to use a package is pretty simple: \usepackage:

command, where package is the name of the package and options is a list of keywords that trigger special features in the package. For example, to use the color package, which lets you typeset in colors, you would type:

You can pass several options to a package, each separated by a comma.

2.1.3 The *document* environment

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:

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 title will never be typeset.

Using this approach, you can only create a title with a fixed layout. If you want to create your title freely, see the Title Creation section.

Abstract

As most research papers have an abstract, there are pre-defined 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*.

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 before you begin the abstract environment:

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 are some of the structure commands found in *simple.tex*.

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 use \begin and \end commands to indicate which content belongs to a given block.

LaTeX provides 7 levels of depth for defining sections (see table below). Each section in this table is a subsection of the one above it.

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 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 3. 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 you wish. For example, if you want to change it to "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:

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:

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 package unnumberedtotoc [1]. It provides the command

which will take care of a proper header as well. \addpart and addchap are also available. KOMA classes provide those commands by default.

If you don't want to use package unnumberedtotoc, you have to do everything by hand using \addcontentsline and \markright{} (or even \markboth{}{}).

Note that if you use PDF bookmarks you will need to add a phantom section so that hyperlinks will lead to the correct place in the document. The \phantomsection command is defined in the hyperref package, and is Commonly used like this:

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

Section number style   See Counters.

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 Paragraph 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:

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 pack-

age is used and the link does not point to the correct chapter, the command \phantomsection in combination with \clearpage or \cleardoublepage can be used (see also Labels and Cross-referencing):

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:

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.

You can change the depth of specific section type, which could be useful for PDF bookmarks (if you are using the hyperref package) :

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, doc).

2.1.4   Book structure

The standard LaTeX book class follows the same layout described above with some additions. By default a book will be two-sided, *i.e.* left and right margins will change according to the page number parity. Furthermore current chapter and section will be printed in the header.

If you do not make use of chapters, it is barely useful to use the book class.

Additionally the class provides macros to change the formatting of some places of the document. We will give you some advice on how to use them properly.[2]

- The frontmatter chapters will not be numbered. Page numbers will be printed in roman numerals. Frontmatter is not supposed to have sections, so they will be numbered 0.n because there is no chapter numbering. Check the Counters chapter for a fix.

- The mainmatter chapters works as usual. The command resets the page numbering. Page numbers will be printed in arabic numerals.

- The \appendix macro can be used to indicate that following sections or chapters are to be numbered as appendices. Appendices can be used for the article class too:

Only use the \appendix macro once for all appendices.

- The backmatter behaves like the frontmatter. It has the same issue with section numbering.

As a general rule you should avoid mixing the command order. Nonetheless all commands are optional, so you might consider using only a few.

Note that the special content like the table of contents is considered as an unnumbered chapter.

Page order

This is one traditional page order for books.

Frontmatter

1. Half-title
2. Empty
3. Title page
4. Information (copyright notice, ISBN, etc.)
5. Dedication if any, else empty
6. Table of contents
7. List of figures (can be in the backmatter too)
8. Preface chapter

Mainmatter

1. Main topic

Appendix

1. Some subordinate chapters

Backmatter

1. Bibliography
2. Glossary / Index

2.1.5 Special pages

Comprehensive papers often feature special pages at the end, like indices, glossaries and bibliographies. Since this is a quite complex topic, we will give you details in the dedicated part *Special Pages*.

Bibliography

Any good research paper will have a complete list of references. LaTeX has two ways of inserting your references into a document:

- 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 and then link them via a command to your current document and use a Bibtex style 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.

To learn how to add a bibliography to your document, see the Bibliography Management section.

2.1.6 Notes and references

[1] unnumberedtotoc

[2] http://tex.stackexchange.com/questions/20538/what-is-the-right-order-when-using-frontmatter-tableofcontents-mainmatter

2.2 Text Formatting

This section will guide you through the formatting techniques of the text. *Formatting* tends to refer to most things to do with appearance, so it makes the list of possible topics quite eclectic: text style, spacing, etc. If *formatting* may also refer to paragraphs and to the page layout, we will focus on the customization of words and sentences for now.

A lot of 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.

LaTeX is so flexible that we will actually only skim 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!

2.2.1 Spacing

Line Spacing

If you want to use larger inter-line spacing in a document, you can change its value by putting the

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. This may not be ideal in all situations: see http://tex.stackexchange.com/questions/30073/why-is-the-linespread-factor-as-it-is .

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):

To change line spacing within the document, the setspace package provides the environments singlespace, onehalf-space, doublespace and spacing:

Non-breaking spaces

This *essential* feature is a bit unknown to newcomers, although it is available on most WYSIWYG document processors. A non-breaking space between two tokens (e.g. words, punctuation marks) prevents the processors from inserting a line break between them. Besides a non-breaking space cannot be enlarged. It is very important for a consistent reading.

LaTeX uses the '~' symbol as a non-breaking space. You would usually use non-breaking spaces for punctuation marks in some languages, for units and currencies, for initials, etc. In French typography, you would put a non-breaking space before all two-parts punctuation marks.

Examples:

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

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 spac-

ing, 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 non-breaking space. 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.)

Stretched spaces

You can insert a horizontal stretched space with \hfill in a line so that the rest gets "pushed" toward the right margin. For instance this may be useful in the header.

Similarly you can insert vertical stretched space with \vfill. It may be useful for special pages.

See Lengths for more details.

Manual spacing

The spaces between words and sentences, between paragraphs, sections, subsections, etc. is determined automatically by LaTeX. It is against LaTeX philosophy to insert spaces manually and will usually lead to bad formatting. Manual spacing is a matter of macro writing and package creation.

See Lengths for more details.

2.2.2   Hyphenation

LaTeX hyphenates words whenever necessary. Hyphenation rules will vary for different languages. LaTeX only supports English by default, so if you want to have correct hyphenation rules for your desired language, see Internationalization.

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

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. (2012 list: https://www.tug.org/TUGboat/tb33-1/

tb103hyf.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:

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.

LaTeX does not hyphenate compound words that contain a dash[1]. There are two packages that can add back flexibility. The hyphenat package supplies the \hyp command. This command typesets the dash and then subjects the constituent words to automatic hyphenation. After loading the package:

one should write, instead of electromagnetic-endioscopy:

The extdash package also offers features for controlling the hyphenation of compound words containing dashes — as opposed to the words themselves which it leaves to LaTeX. The shortcuts option enables a more compressed syntax:

Typical usage is as follows, assuming the compressed syntax. In both cases, LaTeX can break and hyphenate the constituent words, but in the latter case, it will not break after the L:

One or more words can be kept together on the **one line** with the standard LaTeX command:

This prevents hyphenation and causes its argument to be kept together under all circumstances. For example:

\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:

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. Also note that using a higher text width will decrease the probability of encountering badly hyphenated

word. For example adding

will widen the text width and reduce the amount of margin overruns.

2.2.3 Quote-marks

LaTeX treats left and right quotes as different entities. For single quotes, a grave accent, ` (on American keyboards, this symbol is found on the tilde key; adjacent to the number 1 key on most keyboards) gives a left quote mark, and an apostrophe, ' gives a right. For double quotes, simply double the symbols, and LaTeX will interpret them accordingly. (Don't use the " for right double quotes: when the babel package is used for some languages (e.g. German), the " is redefined to produce an umlaut accent; using " for right double quotes will either lead to bad spacing or it being used to produce an umlaut). 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.

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:

See Fonts for more details on font encoding.

The package csquotes offers a multilingual 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.

2.2.4 Diacritics and accents

Most accents and diacritics may be inserted with direct keyboard input by configuring the preamble properly. For symbols unavailable on your keyboard, diacritics may be added to letters by placing special escaped metacharacters before the letter that requires the diacritic.

See Special Characters.

2.2.5 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.

Another solution is to edit the text to avoid long words, numbers or URLs approaching the side margin.

2.2.6   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:

Ligatures can interfere with some text-search tools (a search for "finally" wouldn't find the string "finally"). The \DisableLigatures from the microtype package can disable ligatures in the whole document to increase accessibility.

Note that this will also disable ligatures such as "--" to "–", "---" to "—", 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.

Another solution is to use the cmap package, which will help the reader to interpret the ligatures:

2.2.7   Slash marks

The normal typesetting of the / character in LaTeX does not allow following characters to be "broken" onto 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.

A word after / or \slash is not automatically hyphenated. This is a similar problem to non-hyphenation of words with a dash described under Hyphenation. One way to have both a line break and automatic hyphenation in both words is

Both / and \slash can be used with a zero \hspace like this. \slash includes a penalty to make a line break there less desirable. This combination can be made into a new slash macro if desired. The hyphenat package includes an \fshyp which will add a hyphen after the slash like "input/-output" if the line breaks there.

2.2.8   Fonts

To change the font family, emphasize text, and other font-related issues, see Fonts.

2.2.9   Formatting macros

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.

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 \textit, without having to wade through your document, identifying all the occurrences of \textit and then figuring out for each one whether it was used for pointing out danger or for some other reason.

See Macros for more details.

2.2.10   Text mode superscript and sub-script

Sub and superscripting can be done quite easily using \textsubscript{} and \textsuperscript{}.

Note: A current LaTeX version is needed to use sub-scripts that way.

2.2.11   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:

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:

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:

2.2.12 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:

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

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

The commands \textendash and \textemdash are also used to produce en-dash (–), and em-dash (—), respectively.

2.2.13 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:

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

2.2.14 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 2$_\varepsilon$	The current incarnation

2.2.15 Notes and References

[1] hyphenat package documentation, p3

2.3 Paragraph Formatting

Altering the paragraph formatting is rarely necessary in academic writing. It is primarily used for formatting text in floats or for more exotic documents.

2.3.1 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.

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.

However, if you *really* need to disable one of the above commands locally (for example because you have to use some broken package), you can use the command \justifying from package ragged2e.

2.3.2 Paragraph indent and break

By default, the first paragraph after a heading follows the standard Anglo-American publishers' practice of no indentation. The size of subsequent paragraph indents is 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 it by using the \setlength command. This will set paragraph indents to 1cm:

Whitespace in LaTeX can also be made flexible (what Lamport calls "rubber" lengths). This means that values such as extra vertical space inserted before a paragraph \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:

If you want to indent a paragraph that is not indented, you can use

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: once loaded, the beginning of any chapter/section is indented by the usual paragraph indentation.

To create a non-indented paragraph, you can use

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.

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. If you want both indent and break, use

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.

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.

2.3.3   \paragraph line break

Default style for \paragraph may seem odd in the first place, as it writes the following text next to the title. If you do not like it, use a class other than the traditional article/book, or use ConTeXt or PlainTeX. Hacking of the class in use is really not the way LaTeX is intended to be used, and you may encounter a lot of frustrating issues.

Anyway, let's analyse the problem. If you add a manual line break with \\, LaTeX will complain that

There's no line here to end.

Simply adding an empty space will do it:

Alternatively you can use the shorter, yet not completely equivalent syntax:

2.3.4   Line spacing

To change line spacing in the whole document use the command \linespread covered in Text Formatting.

Alternatively, you can use the \usepackage{setspace} package, which is also covered in Text Formatting. This package provides the commands \dou-

blespacing, \onehalfspacing, \singlespacing and \setstretch{baselinestretch}, which will specify the line spacing for all sections and paragraphs until another command is used. Furthermore, the package provides the following environments in order to change line spacing within the document but not document-wide:

- doublespace: lines are double spaced;

- onehalfspace: line spacing set to one-and-half spacing;

- singlespace: normal line spacing;

- spacing: customizable line spacing, e.g.   \begin{spacing}{\baselinestretch} ... \end{spacing}.

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

2.3.5   Manual breaks

LaTeX takes care of formatting, breaks included. You should avoid manual breaking as much as possible, for it could lead to very bad formatting.

Controlling the breaks should be reserved to macro and package writers. Here follows a quick reference.

The page breaks are covered in Page Layout. More details on manual spaces between paragraphs (such as \bigskip) can be found in Lengths.

2.3.6   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:

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 a problem, you can use the alltt package instead, providing an environment with the same name:

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:

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

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.

To use verbatim in beamer, the frame needs to be made fragile: \begin{frame}[fragile] .

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

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 midway 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{}. (Alternately pdfborder = {0 0 0} might work if the four zeroes do not.)

You can put the following code into your preamble to change the style, how URLs are displayed to the normal font:

See also Hyperlinks

***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: \be-

gin{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.

Multiline 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, unless you're using an editor that automates this process. Alternatively, you can use the verbatim package, to be loaded in the preamble as usual:

(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:

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:

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

This approach can, however, produce unwanted spaces in the document, so it may work better to use

Then if you supply only one argument to \comment{}, this has the desired effect without producing extra spaces.

Another drawback is that content is still parsed and possibly expanded, so you cannot put anything you want in it (such as LaTeX commands).

Skipping parts of the source

A more robust way of making the TeX engine skip some part of the source is to use the TeX \iffalse-conditional. The typical use is

The \iffalse-conditional is always false.

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. See Document Structure.

2.3.7   Notes and References

This page uses material from Andy Roberts' Getting to grips with LaTeX with permission from the author.

2.4   Colors

Adding colors to your text is supported by the color package. Using this package, you can set the font color, text background, or page background. You can choose from predefined colors or define your own colors using RGB, Hex, or CMYK. Mathematical formulas can also be colored.

2.4.1   Adding the color package

To make use of these features, the color package must be imported.

Alternatively, one can write:

The \usepackage is obvious, but the initialization of additional commands like usenames allows you to use names of the default colors, the same 16 base colors as used in HTML. The dvipsnames allows you access to more colors, another 64, and svgnames allows access to about 150 colors. The initialization of "table" allows colors to be added to tables by placing the color command just before the table. The package loaded here is the xcolor package.

If you need more colors, then you may also want to look at adding the x11names to the initialization section as well,

this offers more than 300 colors, but you need to make sure your xcolor package is the most recent you can download.

2.4.2   Entering colored text

The simplest way to type colored text is by:

where declared-color is a color that was defined before by \definecolor.

Another possible way by

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:

The difference between \textcolor and \color is the same as that between \texttt and \ttfamily, you can use the one you prefer. The \color environment allows the text to run over multiple lines and other text environments whereas the text in \textcolor must all be one paragraph and not contain other environments.

You can change the background color of the whole page by:

2.4.3   Entering colored background for the text

If the background color and the text color is changed, then:

There is also \fcolorbox to make framed background color in yet another color:

2.4.4   Predefined colors

The predefined color names are

black, blue, brown, cyan, darkgray, gray, green, lightgray, lime, magenta, olive, orange, pink, purple, red, teal, violet, white, 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 or pstricks package you must declare the xcolor package before that, otherwise it will not work.

2.4.5 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 the listings package.)

Method

You need to include the xcolor package in your preamble to define new colors. In the abstract, the colors are defined following this scheme:

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*, *RGB*, *HTML*, 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):

Examples

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:

The following code should give a similar results to the last code chunk.

If you loaded the xcolor package, you can define colors upon previously defined ones.

The first specifies 20 percent blue and 80 percent white; the second is a mixture of 20 percent blue and 80 percent black; and the last one is a mixture of (20*0.3) percent blue, ((100-20)*0.3) percent black and (100-30) percent green.

xcolor also feature a handy command to define colors from color mixes:

Using color specifications directly

Normally one would predeclare all the colors as above, but sometimes it is convenient to directly use a color without naming it first. To achieve this, \color and \textcolor have an alternative syntax specifying the model in square brackets, and the color specification in curly braces. For example:

Creating / Capturing colors

You may want to use colors that appear on another document, web pages, pictures, etc. Alternatively, you may want to play around with rgb values to create your own custom colors.

Image processing suites like the free GIMP suite for Linux/Windows/Mac offer color picker facilities to capture any color on your screen or synthesize colors directly from their respective rgb / hsv / hexadecimal values.

Smaller, free utilities also exist:

- Linux/BSD: The gcolor2 tool (usually also available in repositories)

- Microsoft Windows: The open-source Color Selector tool.

- Apple Macs: Hex Color Picker for creating custom colors and the built-in DigitalColor Meter for capturing colors on screen.

- Online utilities: See here for a Wikipedia article with several external links

Spot colors

Spot colors are customary in printing. They usually refer to pre-mixed inks based on a swatchbook (like Pantone, TruMatch or Toyo). The package colorspace extends xcolor to provide real spot colors. They are defined with, say:

2.4.6 Sources

- The xcolor manual

- The color package documentation

2.5 Fonts

Fonts are a complex topic. For common documents, only Font families, Emphasizing text, and Font encoding are really needed. The other sections are more useful to macro writers or for very specific needs.

2.5.1   Introduction

The digital fonts have a long and intricate history. See Adobe Font Metrics for some more details.

Originally TeX was conceived to use its own font system, MetaFont, designed by D. Knuth. The default font family for TeX and friends is called Computer Modern. These high quality fonts are scalable, and have a wide range of typographical fine tuning capabilities.

Standard tex compilers will let you use other fonts. There are many different font types, such as PostScript Type1/Type3 fonts and bitmap fonts. Type1 are outline fonts (vector graphics) which are commonly used by pdf-tex. Bitmap fonts are raster graphics, and usually have very poor quality, which can easily be seen when zooming or printing a document. Type3 is a superset of Type1 and has more functionalities from Postscript, such as embedding raster graphics. In the TeX world, Type3 fonts are often used to embed bitmap fonts.

It should be noticed that fonts get generated the first time they are required, hence the long compilation time.

However, MetaFont is internally a quite complex font system, and the most popular font systems as of this day are Truetype font (ttf) and OpenType font (otf). With modern TeX compilers such as xetex and luatex it is possible to make use of such fonts in LaTeX documents. If you want/have to stick with the standard compilers, the aforementioned font types must first be converted and made available to LaTeX (*e.g.* converted to Type1 fonts). The external links section below has some useful resources.

In LaTeX, there are many ways to specify and control fonts. It is a very complex matter in typography.

2.5.2   Font families

There are many font families e.g. Computer Modern, Times, Arial, and Courier. Those families can be grouped into three main categories: roman (rm) or serif, sans serif (sf) and monospace (tt) (see Typeface for more details). Each font family comes with the default design which falls into one of those categories; however, it is interchangeable among them. Computer Modern Roman is the default font family for LaTeX. Fonts in each family also have different properties (size, shape, weight, etc.). Families are meant to be consistent, so it is highly discouraged to change fonts individually rather than the whole family.

The three families are defined by their respective variables:

- \rmdefault
- \sfdefault
- \ttdefault

The default family is contained in the \familydefault variable, and it is meant to have one of the three aforementioned variables as value. The default is defined (in the preamble) like the following assignment:

This will turn all the part of the document using the default font to the default sans serif, which is Computer Modern Sans Serif if you did not change the default font.

Changing font families usually works in two steps:

1. First specify which family you want to change (rm, sf or tt).

2. Second specify the new default family if it is not rm.

Mathematical fonts is a more complex matter. Fonts may come with a package that will take care of defining all three families plus the math fonts. You can do it by yourself, in which case you do not have to load any package.

Below is an example[1] that demonstrates how to change a specific family.

The three default family font variables and the \familydefault variable should not be confused with their respective switch:

- \normalfont
- \rmfamily
- \sffamily
- \ttfamily

2.5.3   Available LaTeX Fonts [2]

To choose a font of your liking, please visit http://www.tug.dk/FontCatalogue/. Here are some common examples.

Below are some fonts which are installed by default.

Serif Fonts

Sans Serif Fonts

Typewriter Fonts   Furthermore, the Bera Mono (BitStream Vera Mono) and LuxiMono fonts were designed to look good when used in conjunction with the Computer Modern serif font.

\usepackage[scaled=0.85]{beramono}

Cursive Fonts   Since LaTeX has no generic family group for cursive fonts, these fonts are usually assigned to the roman family.

Mathematical Formula Fonts

2.5.4 Emphasizing text

In order to add some emphasis to a word or a phrase, the simplest way is to use the \emph{text} command, which usually italicizes the text. Italics may be specified explicitly with \textit{text}.

Note that the \emph command is dynamic: if you emphasize a word which is already in an emphasized sentence, it will be reverted to the upright font.

Text may be emphasized more heavily through the use of boldface, particularly for keywords the reader may be trying to find when reading the text. As bold text is generally read before any other text in a paragraph or even on a page, it should be used sparingly. It may also be used in place of italics when using sans-serif typefaces to provide a greater contrast with unemphasized text. Bold text can be generated with the \textbf{text} command.

2.5.5 Font encoding

A *character* is a sequence of bytes, and should not be confused with its representation, the *glyph*, which is what the reader sees. So the character 'a' has different representations following the used font, for example the upright version, the italic version, various weights and heights, and so on.

Upon compilation, tex will have to choose the right font glyph for every character. This is what is called *font encoding*. 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 has some caveats.

- It stops the automatic hyphenation from working inside words containing accented characters.

- Searches for words with accents in PDFs will fail.

- Extracting ('e.g.' copy paste) the umlaut 'Ä' via a PDF viewer actually extracts the two characters '"A'.

- 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.

All this is not possible with *OT1*; that's why you may want to change the font encoding of your document.

Note that changing the font encoding will have some requirements over the fonts being used. The default Computer Modern font does not support T1. You will need Computer Modern Super (cm-super) or Latin Modern (lmodern), which are Computer Modern-like fonts with T1 support. If you have none of these, it is quite frequent (depends on your TeX installation) that tex chooses a Type3 font such as the Type3 EC, which is a bitmap font. Bitmap fonts look rather ugly when zoomed or printed.

The fontenc package tells LaTeX what font encoding to use. Font encoding is set with:

where encoding is the font encoding. It is possible to load several encodings simultaneously.

There is nothing to change in your document to use CM Super fonts (assuming they are installed), they will get loaded automatically if you use T1 encoding. For lmodern, you will need to load the package after the T1 encoding has been set:

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.

2.5.6 Font styles

Each family has its own font characteristics (such as italic and bold), also known as font styles, or font properties.

Font styles are usually implemented with different font files. So it is possible to build a new font family by specifying the font styles of different font families.

Shapes

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

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 recom-

mended.

You may have noticed the absence of underline. This is because underlining is not recommended for typographic reasons (it weighs the text down). You should use *emph* instead. However underlining text provides a useful extra form of emphasis during the editing process, for example to draw attention to changes. Although underlining 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 \emph formatting, add \normalem straight after the document environment begins. Alternatively, use \usepackage[normalem]{ulem}.

- To underline, use \uline{...} along with \usepackage[normalem]{ulem}..

- To add a wavy underline, use \uwave{...} along with \usepackage[normalem]{ulem}..

- For a strike-out (strikethrough), use \sout{...} along with \usepackage[normalem]{ulem}..

- For a slash through each individual character \xout{...} along with \usepackage[normalem]{ulem}.

Some font styles are not compatible one with the other. But some extra packages will fill this hole. For bold small capitals, you might want to use:

Sizing text

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

These commands change the size within a given scope, so for instance {\Large some words} will change the size of only some words, and does not affect the font in the rest of the document. It will work for most parts of the text.

These commands cannot be used in math mode. However, part of a formula may be set in a different size by using an \mbox command containing the size command. The new size takes effect immediately after the size command; if an entire paragraph or unit is set in a certain size, the size command should include the blank line or the \end{...} which delimits the unit.

The default for \normalsize is 10 point (option 10pt), but it may differ for some Document Styles or their options.

The actual size produced by these commands also depends on the Document Style and, in some styles, more than one of these size commands may produce the same actual size.

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.

As a technical note, points in TeX follow the standard American point size in which 1 pt is approximately 0.35136 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 0.3527 mm while the standard European point size (known as the *Didot point*) had 1 pt equal to approximately 0.37597151 mm (see: point (typography)).

2.5.7   Local font selection

You can change font for a specific part of the text. There are four font properties you can change.

\fontencoding The font encoding, such as OT1 (TeX default) or T1 (extended characters support, better PDF support, widely used).

\fontfamily The font family.

\fontseries The series: l=light, m=medium, b=bold, bx=very bold.

\fontshape The shape: it=italic, n=normal, sl=slanted, sc=small capitals.

The \selectfont command is mandatory, otherwise the font will not be changed. It is highly recommended to enclose the command in a group to cleanly return to the previous font selection when desired.

You can use all these commands in a row:

The default values are stored in \encodingdefault, \familydefault, \seriesdefault and \shapedefault. Setting back the default font properties can be done with

For short, you can use the \usefont{<encoding>}{<family>}{<series>}{<shape>} command.

2.5.8   Arbitrary font size

The \tiny...\Huge commands are often enough for most contents. These are fixed sizes however. In most document processors, you can usually choose any size for any font. This is because the characters actually get magnified. If it usually looks correct for medium sizes, it will look odd at extreme sizes because of an unbalanced thickness. In TeX it is possible to change the magnification of

anything, but highly discouraged for the aforementioned reason. Changing the font size is made by changing the font file. Yes, there is a file for every size: cmr10 for Computer Modern Roman 10pt, cmr12 for Computer Modern Roman 12pt, etc. This ensure the characters are correctly balanced and remain readable at all defined sizes.

You may choose a particular font size with the \font-size{<size>}{<line space>} command. Example:

If you are using the default Computer Modern font encoding, you may get the following message:

LaTeX Font Warning: Font shape `OT1/cmr/m/n' in size <142.26378> not available (Font) size <24.88> substituted on input line 103.

In that case you will notice that the font size is by default restricted to a set of fixed sizes as noted above. You can use the fix-cm or type1cm packages to allow computer modern fonts to be scaled to arbitrary values.

2.5.9   Finding fonts

You will find a huge font directory along examples and configurations at TUG Font Catalogue.

2.5.10   Using arbitrary system fonts

If you use the XeTeX or LuaTeX engine and the fontspec 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:

Then compile the document with xelatex or lualatex. Note that you can only generate .pdf files, and that you need a sufficiently recent 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. Instead make sure that your document is encoded as UTF-8 and load fontspec, which will take care of the font encoding. To make your document support both pdflatex and xelatex/lualatex you can use the \ifxetex/ \ifluatex macro from the ifxetex/ ifluatex package. For example for xelatex

2.5.11   PDF fonts and properties

PDF documents have the capability to embed font files. It makes them portable, hence the name *Portable Document Format.*

Many PDF viewers have a *Properties* feature to list embedded fonts and document metadata.

Many Unix systems make use of the *poppler* tool set which features pdfinfo to list PDF metadata, and pdffonts to list embedded fonts.

2.5.12   Useful websites

- The Latex Font Catalogue
- LaTeX font commands
- How to change fonts in Latex
- Understanding the world of TEX fonts and mastering the basics of fontinst
- Font installation the shallow way *"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
- A bash script for installing a LaTeX font family (MikTeX / TeXLive)
- LaTeX And TrueType Font
- True Type Fonts with LaTeX under Linux + MiK-TeX 2.5
- Unicode Truetype font installer for LaTeX under Windows + MikTeX
- Using TrueType fonts with TeX (LaTeX) and pdf-TeX (pdfLaTeX) (for MikTeX)

2.5.13   References

[1] found at the Google discussion group *latexlovers*

[2] Taken from http://www.macfreek.nl/memory/Fonts_in_LaTeX

2.6   List Structures

Convenient and predictable list formatting is one of the many advantages of using LaTeX. Users of WYSIWYG word processors can sometimes be frustrated by the software's attempts to determine when they intend lists to begin and end. As a mark-up language, LaTeX gives more control over the structure and content of lists.

2.6.1   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 types:

- itemize for a bullet list

- enumerate for an enumerated list and

- description for a descriptive list.

All lists follow the basic format:

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.

Try out the examples below, to see what the lists look like in a real document.

LaTeX will happily allow you to insert a list environment into an existing one (up to a depth of four, more levels are available using packages). Simply begin the appropriate environment at the desired point within the current list. Latex will sort out the layout and any numbering for you.

2.6.2   Some special lists

Sometimes you feel the need to better align the different list items. If you are using a KOMA-script class (or package scrextend), the labeling environment is handy. It takes a mandatory argument that contains the longest of your labels.

If you are on tight space limitations and only have short item descriptions, you may want to have the list inline. Please note that the example also shows how to change the font.

Need some details on Colors?

If you want a horizontal list, package tasks can be handy. In combination with a package like exsheets, you can prepare exam papers for students.

2.6.3   Customizing lists

Especially when dealing with lists containing of just a few words per item, the standard lists take up too much space and you want to customize the appearance. Package enumitem helps you by providing a simple interface.

You can change the appearance of lists globally in the preamble, or just for single lists using the optional argument of the environment. Have a look at the following example where the list on the right is more compact using noitemsep.

An example for alignment and the width of the label.

The documentation of package enumitem goes into more detail with respect to what can be changed and how. You can even define your own lists. Environments like labeling and tasks can be changed differently, details can be found in the package documentation respectively.

2.6.4   Easylist package

The easylist package allows you to create list using a more convenient syntax and with infinite nested levels. It is also very customizable.

Load the package with the control character as optional argument:

The easylist environment will default to enumerations.

It features predefined styles which you can set as optional argument.

Available styles:

- tractatus

- checklist - All items have empty check boxes next to them

- booktoc - Approximately the format used by the table of contents of the book class

- articletoc - Approximately the format used by the table of contents of the article class

- enumerate - The default

- itemize

You can customize lists with the \ListProperties(...) command and revert back the customization with \NewList. Yes, that's parentheses for \ListProperties parameters.

The Style parameter sets the style of counters and text, the Style* parameter sets the style of counters, and the Style** parameter sets the style of text. The parameter Numbers determines the way that the numbers are displayed and the possible values are r or R (for lower and upper case Roman numerals), l or L (for lower and upper case letters), a (for Arabic numbers, the default), and z (for Zapf's Dingbats).

The FinalMark parameter sets the punctuation of the final counter (Ex: FinalMark3={)}) while FinalSpace sets the amount of space between the item and the item's text. The Margin parameter sets the distance from the left margin (Ex: FinalSpace2=1cm). The Progressive parameter sets the distance from the left margin of all items in proportion to their level.

The Hide = n parameter prevents the first n counters from appearing in all levels. If there is a number after a parameter (Ex: Style3*) then this numbers indicates the level that it will affect (Ex: Style3=\color{red}).

Example of custom enumerate:

Note that we put the FinalMark argument between { } to avoid LaTeX understanding it as the end of the properties list. Now we change the default properties to print a custom itemize:

Spaces in Style parameters are important. The Style* parameter acts as a default value and easylist will use a medium dash for level 1, 5 and onward.

You can also define custom styles using LaTeX macros:

Important note: easylist has some drawbacks. First if you need to put an easylist inside an environment using the same control character as the one specified for easylist, you may get an error. To circumvent it, use the following commands provided by easylist:

Besides using easylist along with figures may cause some trouble to the layout and the indentation. LaTeX lists do not have this problem.

To use easylist with Beamer, each frame that uses easylist must be marked as fragile:

2.7 Special Characters

In this chapter we will tackle matters related to input encoding, typesetting diacritics and special characters.

In the following document, we will refer to *special characters* for all symbols other than *A-Za-z0-9* and English punctuation marks.

This chapter is tightly linked with the font encoding issue. You should have a look at Fonts on the topic.

Some languages usually need a dedicated input system to ease document writing. This is the case for Arabic, Chinese, Japanese, Korean and others. This specific matter will be tackled in Internationalization.

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.

2.7.1 Input encoding

A technical matter

Most of the modern computer systems allow you to input letters of national alphabets directly from the keyboard. If you tried to input these special characters in your LaTeX source file and compiled it, you may have noticed that they do not get printed at all.

A LaTeX source document is a plain text file. A computer stores data in a binary format, that is a sequence of bits (0 and 1). To display a plain text file, we need a code which tells which sequence of bits corresponds to which sequence of characters. This association is called *input*

encoding, *character encoding*, or more informally *charset*.

For historical reasons, there are many different input encodings. There is an attempt to unify all the encoding with a specification that contains all existent symbols that are known from human history. This specification is *Unicode*. It only defines *code points*, which is a number for a symbol, but not the way symbols are represented in binary value. For that, unicode encodings are in charge. There are also several unicode encodings available, UTF-8 being one of them.

The ASCII encoding is an encoding which defines 128 characters on 7 bits. Its widespread use has led the vast majority of encodings to have backward compatibility with ASCII, by defining the first 128 characters the same way. The other characters are added using more bits (8 or more).

This is actually a big issue, since if you do not use the right encoding to display a file, it will show weird characters. What most programs try to do is guess statistically the encoding by analyzing the frequent sequences of bits. Sadly, it is not 100% safe. Some text editors may not bother guessing the encoding and will just use the OS default encoding. You should consider that other people might not be able to display directly your input files on their computer, because the default encoding for text file is different. It does not mean that the user cannot use another encoding, besides the default one, only that it has to be configured. For example, the German umlaut ä on OS/2 is encoded as 132, with Latin1 it is encoded as 228, while in Cyrillic encoding cp1251 this letter does not exist at all. Therefore you should *consider encoding with care*.

The following table shows the default encodings for some operating systems.

UTF-8 and Latin1 are not compatible. It means that if you try to open a Latin1-encoded file using a UTF-8 decoding, it will display odd symbols only if you used accents in it, since both encoding are ASCII superset they encode the *classic* letters the same way. There aren't many advantages in using Latin1 over UTF-8, which is technically superior. UTF-8 is also becoming the most widely used encoding (on the Web, in modern Unices, etc.).

Dealing with LaTeX

TeX uses ASCII by default. But 128 characters is not enough to support non-english languages. TeX has its own way to do that with commands for every diacritical marking (see Escaped codes). But if we want accents and other special characters to appear directly in the source file, we have to tell TeX that we want to use a different encoding.

There are several encodings available to LaTeX:

- ASCII: the default. Only bare english characters are supported in the source file.

- ISO-8859-1 (a.k.a. Latin 1): 8-bits encoding. It supports most characters for latin languages, but that's it.

- UTF-8: a Unicode multi-byte encoding. Supports the complete Unicode specification.

- Others...

In the following we will assume you want to use UTF-8.

There are some *important steps* to specify encoding.

- Make sure your text editor decodes the file in UTF-8.

- Make sure it saves your file in UTF-8. Most text editors do not make the distinction, but some do, such as Notepad++.

- If you are working in a terminal, make sure it is set to support UTF-8 input and output. Some old Unix terminals may not support UTF-8. PuTTY is not set to use UTF-8 by default, you have to configure it.

- Tell LaTeX that the source file is UTF-8 encoded.

inputenc [2] package tells LaTeX what the text encoding format of your .tex files is.

The inputenc package allows as well the user to change the encoding *within the document* by means of the command \inputencoding{'encoding name'}.

Extending the support

The LaTeX support of UTF-8 is fairly specific: it includes only a limited range of unicode input characters. It only defines those symbols that are known to be available with the current *font encoding*. You might encounter a situation where using UTF-8 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 may try need to use the utf8x option to define more character combinations. utf8x is not officially supported, but can be viable in some cases. However it might break up compatibility with some packages like csquotes.

Another possiblity is to stick with utf8 and to define the characters yourself. This is easy:

where codepoint is the unicode codepoint of the desired character. TeX sequence is what to print when the character matching the codepoint is met. You may find codepoints on this site. Codepoints are easy to find on the web. Example:

Now inputting 'ŷ' will effectively print 'ŷ'.

With XeTeX and LuaTeX the inputenc package is no longer needed. Both engines support UTF-8 directly and allow the use of TTF and OpenType fonts to support Unicode characters. See the Fonts section for more information.

2.7.2 Escaped codes

In addition to direct UTF-8 input, LaTeX supports the composition of special characters. This is convenient if your keyboard lacks some desired accents and other diacritics.

The following accents may be placed on letters. Although 'o' letter 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.

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[german]{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.

2.7.3 *Less than* < and *greater than* >

The two symbols '<' and '>' are actually ASCII characters, but you may have noticed that they will print '¡' and '¿' respectively. This is a font encoding issue. If you want them to print their real symbol, you will have to use another font encoding such as T1, loaded with the fontenc package. See Fonts for more details on font encoding.

Alternatively, they can be printed with dedicated commands:

2.7.4 Euro € currency symbol

When writing about money these days, you need the euro sign. The textcomp package features a \texteuro command which gives you the euro symbol as supplied by your current text font. Depending on your chosen font this may be quite far from the official symbol.

An official version of the euro symbol is provided by eurosym. Load it in the preamble (optionally with the official option):

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.) you can use a different option:

again you can insert the euro symbol with \euro{}.

Alternatively you can use the marvosym package which also provides the official euro symbol.

Now that you have succeeded in printing a euro sign, you may want the '€' on your keyboard to actually print the euro sign as above. There is a simple method to do that. You must make sure you are using UTF-8 encoding along with a working \euro{} or \EUR{}command.

Complete example:

2.7.5 Degree symbol for temperature and math

The easiest way to print temperature and angle values is to use the \SI{value}{unit} command from the siunitx package, which works both in text and math mode:

For more information, see the documentation of the siunitx package.

A common mistake is to use the \circ command. It will not print the correct character (though $^\circ$ will). Use the textcomp package instead, which provides a \textdegree command.

For temperature, you can use the same command or opt for the gensymb package and write

Some keyboard layouts feature the degree symbol, you can use it directly if you are using UTF-8 and textcomp. For better results (font quality) we recommend the use of an appropriate font, like lmodern:

2.7.6 Other symbols

LaTeX has many 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:

Not mentioned in above table, tilde (~) is used in LaTeX code to produce non-breakable space. To get printed tilde sign, either write \~{} or \textasciitilde{}. And a visible

space ␣ can be created with \textvisiblespace.

For some more interesting symbols, the Postscript ZipfDingbats font is available thanks to the pifont package. Add the declaration to your preamble: \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	✄
40	✈	41	✉	42	☛	43	☞	44	✌
48	✏	49	✐	50	✑	51	✓	52	✔
56	✘	57	✚	58	✛	59	✜	60	✝
64	✠	65	✡	66	✢	67	✣	68	✤
72	★	73	☆	74	✧	75	✩	76	✪
80	✫	81	✱	82	✲	83	✳	84	✴
88	✶	89	✷	90	✸	91	✹	92	✺
96	✿	97	❀	98	❁	99	✽	100	❂
104	✻	105	✼	106	✾	107	❄	108	●
112	❒	113	❏	114	❐	115	▲	116	▼
120	❘	121	❙	122	❚	123	❛	124	❜
		161	❡	162	❢	163	❣	164	❤
168	♣	169	♦	170	♥	171	♠	172	①
176	⑤	177	⑥	178	⑦	179	⑧	180	⑨
184	⑩	185	❶	186	❷	187	❸	188	❹
192	❶	193	❷	194	❸	195	❹	196	❺
200	❾	201	❿	202	❶	203	❷	204	❸
208	❼	209	❽	210	❾	211	❿	212	→
216	↘	217	→	218	↗	219	➣	220	➤
224	➥	225	➡	226	➢	227	➣	228	➤
232	➦	233	⇨	234	⇨	235	⇦	236	⇨
		241	⇨	242	⊃	243	➪	244	➚
248	➨	249	➷	250	➺	251	↔	252	➼

2.7.7 In special environments

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.

When applying accents to letters i and j, you can use \imath and \jmath to keep the dots from interfering with the accents:

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

2.7.8   Unicode keyboard input

Some operating systems provide a keyboard combination to input any Unicode code point, the so-called *unicode compose key*.

Many X applications (*BSD and GNU/Linux) support the Ctrl+Shift+u combination. A 'u' symbol should appear. Type the code point and press enter or space to actually print the character. Example:

<Ctrl+Shift+u> 20AC <space>

will print the euro character.

Desktop environments like GNOME and KDE may feature a customizable compose key for more memorizable sequences.

Xorg features advanced keyboard layouts with variants that let you enter a lot of characters easily with combination using the aprioriate modifier, like Alt Gr. It highly depends on the selected layout+variant, so we suggest you to play a bit with your keyboard, preceeding every key and dead key with the Alt Gr modifier.

2.7.9   External links

- A few other LaTeX accents and symbols

- NASA GISS: Accents

- The Comprehensive LATEX Symbol List

- PDF document with a lengthy list of symbols provided by various packages

2.7.10   Notes and References

[1] For a quick explanation on character sets, see this article on Joel Spolski's blog.

[2] For a detailed information on the package, see complete specifications written by the package's authors.

2.8   Internationalization

LaTeX has to be configured and used appropriately when it is used to write documents in languages other than English. This has to address three main areas:

1. LaTeX needs to know how to hyphenate the language(s) to be used.

2. The user needs to use language-specific typographic rules. In French for example, there is a mandatory space before each colon character (:).

3. The input of special characters, especially for languages using an input system (Arab, Chinese, Japanese, Korean).

It is convenient to be able to insert language-specific special characters directly from the keyboard instead of using cumbersome coding (for example, by typing ä instead of \"{a}). This can be done by configuring input encoding properly. We will not tackle this issue here: see the Special Characters chapter.

Some languages require special fonts with the proper font encoding set. See Font encoding.

Some of the methods described in this chapter may be useful when dealing with non-English author names in bibliographies.

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.

2.8.1   Prerequisites

Most non-english language will need to input special characters very often. For a convenient writing you will need to set the input encoding and the font encoding properly.

The following configuration is optimal for many languages (most latin languages). Make sure your document is saved using the UTF-8 encoding.

For more details check Font encoding and Special Characters.

2.8.2   Babel

The babel package by Johannes Braams and Javier Bezos will take care of everything (with XeTeX and LuaTeX you should consider polyglossia). You can load it in your preamble, providing as an argument name of the language you want to use (usually its English name, but not always):

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. 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:

then the last language in the option list will be active (i.e.

languageB), and you can use the command

to change the active language. You can also add short pieces of text in another language using the command

Babel also offers various environments for entering larger pieces of text in another language:

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' (but note selectlanguage* is preferred).

The babel manual provides much more information on these and many other options.

2.8.3 Multilingual versions

It is possible in LaTeX to typeset the content of one document in several languages and to choose upon compilation which language to output. This might be convenient to keep a consistent sectioning and formatting across the different languages. It is also useful if you make use of multiple proper nouns and other untranslated content. Using the commands above in multilingual documents can be cumbersome, and therefore babel provides a way to define shorter names. With

You can write:

Alternative choice using iflang

The current language can also be tested by using the iflang package by Heiko Oberdiek (the built-in feature from the babel package is not reliable). Here comes a simple example:

\IfLanguageName{ngerman}{Hallo}{Hello}

This allows to easily distinguish between two languages without the need of defining own commands. The babel language is changed by setting

\selectlanguage{english}

2.8.4 Specific 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:

You can input text in either romanized characters or native Arabic script encodings. Use any of the following commands and environments to enter in text:

See the ArabTeX Wikipedia article for further details.

You may also use the Arabi package within Babel to typeset Arabic and Persian

You may also copy and paste from PDF files produced with Arabi thanks to the support of the cmap package. You may use Arabi with LyX, or with tex4ht to produce HTML.

See Arabi page on CTAN

Armenian

The Armenian script uses its own characters, which will require you to install a text editor that supports Unicode and will allow you to enter UTF-8 text, such as Texmaker or WinEdt. These text editors should then be configured to compile using XeLaTeX.

Once the text editor is set up to compile with XeLaTeX, the fontspec package can be used to write in Armenian:

or

The Sylfaen font lacks italic and bold, but DejaVu Serif supports them.

See Armenian Wikibooks for further details, especially on how to configure the Unicode supporting text editors to compile with XeLaTeX.

Cyrillic script

Version 3.7h of babel includes support for the T2* encodings and for typesetting Bulgarian, Russian and Ukrainian texts using Cyrillic letters[1]. Support for Cyrillic is based on standard LaTeX mechanisms plus the fontenc and inputenc packages. AMS-LaTeX packages should be loaded before fontenc and babel[(Why?)]. If you are going to use Cyrillics in mathmode, you also need to load mathtext package before fontenc:

Generally, babel will automatically choose the default font encoding, for the above three languages this is T2A. However, documents are not restricted to a single font encoding. For multilingual documents using Cyrillic and Latin-based languages it makes sense to include Latin font encoding explicitly. Babel will take care of switching to the appropriate font encoding when a different language is selected within the document.

On modern operating systems it is beneficial to use Unicode (utf8 or utf8x) instead of KOI8-RU (koi8-ru) as an input encoding for Cyrillic text.

In addition to enabling hyphenations, translating automatically generated text strings, and activating some language specific typographic rules (like \frenchspacing), babel provides some commands allowing typesetting according to the standards of Bulgarian, Russian, or Ukrainian languages.

For all three languages, language specific punctuation is provided: the Cyrillic dash for the text (it is little nar-

rower than Latin dash and surrounded by tiny spaces), a dash for direct speech, quotes, and commands to facilitate hyphenation:

The Russian and Ukrainian options of babel define the commands

which act like \Alph and \alph (commands for turning counters into letters, *e.g.* a, b, c...), but produce capital and small letters of Russian or Ukrainian alphabets (whichever is the active language of the document).

The Bulgarian option of babel provides the commands

which make \Alph and \alph produce letters of either Bulgarian or Latin (English) alphabets. The default behaviour of \Alph and \alph for the Bulgarian language option is to produce letters from the Bulgarian alphabet.

See the Bulgarian translation of "The Not So Short Introduction to LaTeX" [2] for a method to type Cyrillic letters directly from the keyboard using a different distribution.

Chinese

One possible Chinese support is made available thanks to the CJK package collection. If you are using a package manager or a portage tree, the CJK collection is usually in a separate package because of its size (mainly due to fonts).

Make sure your document is saved using the UTF-8 character encoding. See Special Characters for more details. Put the parts where you want to write chinese characters in a CJK environment.

The last argument specifies the font. It must fit the desired language, since fonts are different for Chinese, Japanese and Korean. Possible choices for Chinese include:

- gbsn (⬚⬚⬚⬚, simplified Chinese)
- gkai (⬚⬚⬚⬚, simplified Chinese)
- bsmi (⬚⬚⬚⬚⬚⬚⬚, traditional Chinese)
- bkai (⬚⬚⬚⬚⬚, traditional Chinese)

Czech

Czech is fine using

UTF-8 allows you to have „czech quotation marks" directly in your text. Otherwise, there are macros \clqq and \crqq to produce left and right quote. You can place quoted text inside \uv.

Finnish

Finnish language hyphenation is enabled with:

This will also automatically change document language (section names, etc.) to Finnish.

French

You can load French language support with the following command:

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. If you do not know or do not really care, we would recommend using frenchb.

However, as of version 3.0 of babel-french, it is advised to choose the language as a global option with the following command[3]:

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:

You may want to typeset guillemets and other French characters directly if your keyboard have them. Running Xorg (*BSD and GNU/Linux), you may want to use the oss variant which features some nice shortcuts, like

You will need the T1 font encoding for guillemets to print properly.

For the degree character you will get an error like

! Package inputenc Error: Unicode char \u8:° not set up for use with LaTeX.

The textcomp package will fix it for you.

The great advantage of Babel for French is that it will handle some elements of French typography for you, especially non-breaking spaces before all two-parts punctuation marks. So now you can write:

The non-breaking space before the euro symbol is still necessary because currency symbols and other units or not supported in general (that's not specific to French).

You can use the numprint package along Babel. It will let you print numbers the French way.

You will also notice that the layout of lists changes when switching to the French language. This is customizable using the \frenchbsetup command. 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. You can get the PDF version on CTAN.

German

You can load German language support using *either one* of the two following commands.

For traditional ("old") German orthography use

or for reform ("new") German orthography use

This enables German hyphenation, if you have configured your LaTeX system accordingly. It also changes all automatic text into German, e.g. "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 the table below for inspiration. With inputenc, all this becomes moot, but your text also is locked in a particular encoding world.

In German books you sometimes 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 encoding (which is the default) 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.

Decimal numbers usually have to be written like 0{,}5 (not just 0,5). Packages like ziffer enable input like 0,5. Alternatively, one can use the \num command from the babel and (globally) set the decimal marker using

Greek

This is the preamble you need to write in the Greek language. Note the particular input encoding.

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

Use the following lines:

More information in hungarian.

Icelandic and Faroese

The following lines can be added to write Icelandic text:

This changes text like *Part* into *Hluti*. It makes additional commands available:

To make special characters such as Þ and Æ become available just add:

The default LATEX font encoding is OT1, but it contains only the 128 characters. The T1 encoding contains letters and punctuation characters for most of the European languages using Latin script.

Italian

Italian is well supported by LaTeX. Just add

at the beginning of your document and the output of all the commands will be translated properly.

Japanese

There is a variant of TeX intended for Japanese named pTeX, which supports vertical typesetting.

Another possible way to write in japanese is to use Lualatex and the luatex-ja package. Adapted example from the Luatexja documentation :

You can also use capabilities provided by the Fontspec package and those provided by Luatexja-fontspec to declare the font you want to use in your paper. Let us take an example :

Use UTF-8 as your encoding. In case you don't know how to do this, take a look at Texmaker, a LaTeX editor which use UTF-8 by default.

Another (but old) possible Japanese support is made available thanks to the CJK package collection. If you are using a package manager or a portage tree, the CJK collection is usually in a separate package because of its size (mainly due to fonts).

Make sure your document is saved using the UTF-8 character encoding. See Special Characters for more details. Put the parts where you want to write japanese characters in a CJK environment.

The last argument specifies the font. It must fit the desired language, since fonts are different for Chinese, Japanese and Korean. min is an example for Japanese.

Korean

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.

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

- the CJK package by Werner 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.

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:

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.

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).

In the FAQ section of KTUG it is recommended to use the kotex package

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:

See XePersian page on CTAN

Moreover, Arabic script can be used to type Persian as illustrated in the corresponding section.

Polish

If you plan to use Polish in your UTF-8 encoded document, use the following code

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", "u", "z", 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:

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:

For articles:

Alternatively you can use dedicated document classes:

- the mwart class instead of article,

- mwbk instead of book

- and mwrep instead of report.

Those classes have much more European typography settings but *do not* require the use of Polish babel settings or character encoding.

Simple usage:

Full documentation for those classes is available at http://web.archive.org/web/20040609034031/http://www.ci.pwr.wroc.pl/~{}pmazur/LaTeX/mwclsdoc.pdf (Polish).

Indentation   It may be customary (depending on publisher) to indent the first paragraph in sections and chapters:

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:

To adjust penalties for leaving widows and orphans (clubs in TeX nomenclature) use those commands:

Commas in math   According to some typography rules, fractional parts of numbers should be delimited by a comma, not a dot. To make LaTeX not insert additional space in math mode after a comma (unless there is a space after the comma), use the icomma package.

Unfortunately, it is partially incompatible with the dcolumn package. One needs to either use dots in columns with numerical data in the source file and make dcolumn switch them to commas for display *or* define the column as follows:

The alternative is to use the numprint package, but it is much less convenient.

Another alternative is using package siunitx that lets you typeset numbers and their according units consistently. Number alignment in tables and different output modes re supported.

Further information   Refer the Słownik Ortograficzny (in Polish) for additional information on Polish grammar and typography rules.

Good extract is available at Zasady Typograficzne Składania Tekstu (in Polish).

Portuguese

Add the following code to your preamble:

You can substitute the language for brazilian portuguese by choosing brazilian or brazil.

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:

Spanish

Include the appropriate Babel option:

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:

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, latex2rtf) interacting with LaTeX.

Spanish also defines shorthands for the dot and << >> so that they are used as logical markup: the former is used as decimal marker in math mode, and the output is typically either a comma or a dot; the latter is used for quoted text, and the output is typically either «» or "". This allows different typographical conventions with the same input, as preferences may be quite different from, say, Spain and Mexico.

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

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.

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 preempt 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.

Tibetan

One option to use Tibetan script in LaTeX is to add

to your preamble and use a slightly modified Wylie transliteration for input. Refer to the excellent package documentation for details. More information can be found on

2.8.5   References

[1] The Not So Short Introduction to LaTeX, 2.5.6 Support for Cyrillic, Maksym Polyakov

[2] The Not So Short Introduction to LaTeX, Bulgarian translation

[3] babel-french documentation: "the French language should now be loaded as french, not as frenchb or francais and preferably as a global option of \documentclass. Some tolerance still exists in v3.0, but do not rely on it."

2.9   Rotations

2.9.1   The *rotating* package

The package rotating gives you the possibility to rotate any object of an arbitrary angle. Once you have loaded it with the standard command in the preamble:

you can use three new environments:

it will rotate the whole argument by 90 degrees counterclockwise. Moreover:

it will turn the argument of 30 degrees. You can give any angle as an argument, whether it is positive or negative. It will leave the necessary space to avoid any overlapping of text.

like turn, but it will not add any extra space.

If you want to make a float sideways so that the caption is also rotated, you can use

or

Note, though, they will be placed on a separate page.

If you would like to rotate a TikZ picture you could use sideways together with minipage.

You can also use the \rotatebox command. Let's rotate a tabular inside a table for example:

Options

Default is sidewaysfigures/sidewaystables are oriented depending on page number in two sided documents (takes two passes).

The rotating package takes the following options.

counterclockwise/anticlockwise In single sided documents turn sidewaysfigures/sidewaystables counterclockwise.

clockwise In single sided documents turn sidewaysfigures/sidewaystables clockwise (default).

figuresright In two sided documents all sidewaysfigures/sidewaystables are same orientation (left of figure, table now bottom of page). This is the style preferred by the Chicago Manual of Style (broadside).

figuresleft In two sided documents all sidewaysfigures/sidewaystables are same orientation (left of figure, table now at top of page).

2.9.2   The *rotfloat* package

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

just like for normal tables. The H option can not be used without this package.

2.10   Tables

Tables are a common feature in academic writing, often used to summarize research results. Mastering the art of table construction in LaTeX is therefore necessary to produce quality papers and with sufficient practice one can print beautiful tables of any kind.

Keeping in mind that LaTeX is not a spreadsheet, it makes sense to use a dedicated tool to build tables and then to export these tables into the document. Basic tables are not too taxing, but anything more advanced can take a fair bit of construction; in these cases, more advanced packages can be very useful. However, first it is important to know the basics. Once you are comfortable with basic LaTeX tables, you might have a look at more advanced packages or the export options of your favorite spreadsheet. Thanks to the modular nature of LaTeX, the whole process can be automated in a fairly comfortable way.

For a long time, LaTeX tables were quite a chaotic topic, with dozens of packages doing similar things, while not always being compatible with one another. Sometimes you had to make trade-offs. The situation changed recently (2010) with the release of the tabu package which

combines the power of longtable, tabularx and much more. The tabu environment is far less fragile and restricted than the older alternatives. Nonetheless, before attempting to use this package for the first time it will be beneficial to understand how the classic environment works, since tabu works the same way. Note however that the author of tabu will not fix bugs to the current version, and that the next version introduces new syntax that will likely break existing documents.[1]

2.10.1 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:

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):

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 wraparound 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 list in chapter Lengths.

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:

To specify a font format (such as bold, italic, etc.) for an entire column, you can add >{\format} before you declare the alignment. For example \begin{tabular}{ >{\bfseries}l c >{\itshape}r } will indicate a three column table with the first one aligned to the left and in bold font, the second one aligned in the center and with normal font, and the third aligned to the right and in italic. The "array" package needs to be activated in the preamble for this to work.

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:

Note, any white space inserted between these commands is purely down to one's 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.

Expanding upon that by including some vertical lines:

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

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):

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 will contain text whose length exceeds the column's width, 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). For a more convenient method, have a look at The tabularx package, or The tabulary package.

Instead of p, use the m attribute to have the lines aligned toward the middle of the box or the b attribute to align along the bottom of the box.

Here is a simple 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:

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 by

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 has been cropped, since the output is wider than the page width.

Manually broken paragraphs in table cells

Sometimes it is necessary to not rely on the breaking algorithm when using the p specifier, but rather specify the line breaks by hand. In this case it is easiest to use a \parbox:

Here the t argument controls the placement of the text inside the box. Other allowed values are c for center and b for bottom.

Space between columns

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.

Space between rows

Re-define the \arraystretch command to set the space between rows:

Default value is 1.0.

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

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

Other environments inside tables

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

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).

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:

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

The column specification can be altered using the array package. 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:

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} :

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

@-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:

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:

Aligning columns at decimal points using dcolumn

Instead of using @-expressions to build columns of decimals aligned to the decimal point (or equivalent symbol), it is possible to center a column on the decimal separator using the dcolumn package, which provides a new column specifier for floating point data. See the dcolumn package documentation for more information, but a simple way to

use dcolumn is as follows.

A negative argument provided for the number of decimal places in the new column type allows unlimited decimal places, but may result in rather wide columns. Rounding is not applied, so the data to be tabulated should be adjusted to the number of decimal places specified. Note that a decimal aligned column is typeset in math mode, hence the use of \mathrm for the column heading in the example above. Also, text in a decimal aligned column (for example the header) will be right-aligned before the decimal separator (assuming there's no decimal separator in the text). While this may be fine for very short text, or numeric column headings, it looks cumbersome in the example above. A solution to this is to use the \multicolumn command described below, specifying a single column and its alignment. For example to center the header *Decimal* over its column in the above example, the first line of the table itself would be Left&Right&Center&\multicolumn{1}{c}{Decimal}\\

Bold text and dcolumn   To draw attention to particular entries in a table, it may be nice to use bold text. Ordinarily this is easy, but as dcolumn needs to *see* the decimal point it is rather harder to do. In addition, the usual bold characters are wider than their normal counterparts, meaning that although the decimals may align nicely, the figures (for more than 2-−3 digits on one side of the decimal point) will be visibly misaligned. It is however possible to use normal width bold characters and define a new bold column type, as shown below.[2]

2.10.2   Row specification

It might be convenient to apply the same command over every cell of a row, just as for column. Unfortunately the tabular environment cannot do that by default. We will need tabu instead, which provides the \rowfont option.

2.10.3   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 specify a width p{5.0cm} . And contents is simply the actual data you want to be contained within that cell. A simple example:

Columns spanning multiple rows

The first thing you need to do is add \usepackage{multirow} to the preamble[3]. 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).

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 (unless \multicolumn is also used).

Spanning in both directions simultaneously

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

The command \multicolumn{1}{ 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}{ 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:

2.10.4   Controlling table size

Resize tables

The graphicx packages features the command \resizebox{width}{height}{object} which 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.

Resizing table including the caption

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

Changing font size

A table can be globally switched to a different font size by simply adding the desired size command (here: \foot-

notesize) in the table scope, which may be after the \begin{table} statement if you use floats, otherwise you need to add a group delimiter.

Alternatively, you can change the default font for all the tables in your document by placing the following code in the preamble:

See Fonts for named font sizes. The table caption font size is not affected. To control the caption font size, see Caption Styles.

2.10.5   Colors

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.

The command \hiderowcolors is available to deactivate highlighting from a specified row until the end of the table. Highlighting can be reactivated within the table via the \showrowcolors command. If while using these commands you experience "misplaced \noalign errors" then use the commands at the very beginning or end of a row in your tabular.

or

Colors of individual cells

As above this uses the xcolor package.

2.10.6   Width and stretching

We keep providing documentation for tabular* and tabularx although they are completely eclipsed by the much more powerful and flexible tabu environment. Actually tabu is greatly inspired by those environments, so it may be worth it to have an idea how they work, particularly for tabularx.

The *tabular* * environment

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.

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.

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.

The *tabularx* package

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.

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 to use \newcolumntype to format selected columns 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):

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:

In a way analogous to how new commands with arguments can be created with \newcommand, new column types with arguments can be created with \newcolumntype as follows:

where since there are 4 columns, the sum of the \hsize's (1 + 0.5 + 0.5 + 2) must be equal to 4. The default value used by tabularx for \hsize is 1.

The *tabulary* package

tabulary is a modified tabular* allowing width of columns set for equal heights. tabulary allows easy and convenient writing of well balanced tables.

The problem with tabularx is that it leaves much blank if your cells are almost empty. Besides, it is not easy to have different column sizes.

tabulary tries to balance the column widths so that each column has at least its natural width, without exceeding the maximum length.

The first parameter is the maximum width. tabulary will try not to exceed it, but it will not stretch to it if there is not enough content, contrary to tabularx.

The second parameter is the column disposition. Possible values are those from the tabular environment, plus

These are all capitals.

The *tabu* environment

It works pretty much like tabularx.

to \linewidth specifies the target width. The X parameter can have an optional span factor.

2.10.7 Table across several pages

Long tables are natively supported by LaTeX thanks to the longtable environment. Unfortunately this environment does not support stretching (X columns).

The tabu packages provides the longtabu environment. It has most of the features of tabu, with the additional capability to span multiple pages.

LaTeX can do well with long tables: you can specify a header that will repeat on every page, a header for the first page only, and the same for the footer.

It uses syntax similar to longtable, so you should have a look at its documentation if you want to know more.

Alternatively you can try one of the following packages supertabular or xtab, an extended and somewhat improved version of supertabular.

2.10.8 Partial vertical lines

Adding a partial vertical line to an individual cell:

Removing part of a vertical line in a particular cell:

2.10.9 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.

Or use a parbox to center the image.

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

2.10.10 Footnotes in tables

The tabular environment does not handle footnotes properly. The longtabular fixes that.

Instead of using longtabular we recommend tabu which handles footnotes properly, both in normal and long tables.

2.10.11 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 package is useful for easily providing this professionalism in LaTeX tables, and the documentation 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

Using array

Using booktabs

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 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 uses the booktabs package.

2.10.12 Sideways tables

Tables can also be put on their side within a document using the rotating or the rotfloat package. See the Rotations chapter.

2.10.13 Table with legend

To add a legend to a table the caption package can be used. With the caption package a \caption*{...} statement can be added besides the normal \caption{...}. Example:

The normal caption is needed for labels and references.

2.10.14   The *eqparbox* package

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 \eqmakeboxes 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.

2.10.15   Floating with *table*

In WYSIWYG document processors, it is common to put tables in the middle of the text. This is what we have been doing until now. Professional documents, however, often make it a point to print tables on a dedicated page so that they do not disrupt the flow. From the point of view of the source code, one has no idea on which page the current text is going to lie, so it is hardly possible to guess which page may be appropriate for our table. LaTeX can automate this task by abstracting objects such as tables, pictures, etc., and deciding for us where they might fit best. This abstraction is called a *float*. Generally, an object that is floated will appear in the vicinity of its introduction in the source file, but one can choose to control its position also.

To tell LaTeX we want to use our table as a float, we need to put a table environment around the tabular environment, which is able to float and add a label and caption.

The table environment initiates a type of float just as the environment figure. In fact, the two bear a lot of similarities (positioning, captions, etc.). More information about floating environments, captions etc. can be found in Floats, Figures and Captions.

The environment names may now seem quite confusing. Let's sum it up:

- tabular is for the content itself (columns, lines, etc.).

- table is for the location of the table on the document, plus caption and label support.

In the table, we used a label, so now we can refer to it just like any other reference:

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

The captions now show up in the list of tables, if displayed.

You can set the optional parameter position specifier to define the position of the table, where it should be placed. The following characters are all possible placements. Using sequences of it define your "wishlist" to LaTeX.

Default is *tbp*, which means that it is by default placed on the top of the page. If that's not possible, it's placed at the bottom if possible, or finally with other floating environments on an extra page.

You can force LaTeX to use one given position. E.g. *[!h]* forces LaTeX to place it exactly where you place it (Except when it's really impossible, e.g you place a table *here* and this place would be the last line on a page). Again, understand it correctly: it urges LaTeX to put the table at a specific place, but it will not be placed there if LaTeX thinks it will not look great. If you really want to place your table manually, do not use the table environment.

Centering the table horizontally works like everything else, using the \centering command just after opening the table environment, or by enclosing it with a center environment.

2.10.16   Using spreadsheets and data analysis tools

For complex or dynamic tables, you may want to use a spreadsheet. You might save lots of time by building tables using specialized software and exporting them in LaTeX format. The following plugins and libraries are available for some popular software:

- calc2latex: for OpenOffice.org Calc spreadsheets,

- excel2latex: for Microsoft Office Excel,

- matrix2latex: for MATLAB,

- matrix2latex: for Python and MATLAB,

- pandas: pandas DataFrame's have a method to convert data they contain to latex,

- latex-tools: a Ruby library,

- xtable: a library for R,

- org-mode: for Emacs users, org-mode tables can be used inline in LaTeX documents, see for a tutorial.

- Emacs align commands: the align commands can clean up a messy LaTeX table.

- Online Table generator for LATeX: An online tool for creating simple tables within the browser. LaTeX format is directly generated as you type.

- Create LaTeX tables online : Online tool.

However, copying the generated source code to your document is not convenient at all. For maximum flexibility, generate the source code to a separate file which you can input from your main document file with the \input command. If your speadsheet supports command-line, you can generate your complete document (table included) in one command, using a Makefile for example.

See Modular Documents for more details.

2.10.17   Need more complicated features?

Have a look at one of the following packages:

- hhline: do whatever you want with horizontal lines

- array: gives you more freedom on how to define columns

- colortbl: make your table more colorful

- threeparttable makes it possible to put footnotes both within the table and its caption

- arydshln: creates dashed horizontal and vertical lines

- ctable: allows for footnotes under table and properly spaced caption above (incorporates booktabs package)

- slashbox: create 2D tables with the first cell containing a description for both axes. Not available in Tex Live 2011 or later.

- diagbox: compatible to slashbox, come with Tex Live 2011 or later

- dcolumn: decimal point alignment of numeric cells

- rccol: advanced decimal point alignment of numeric cells with rounding

- numprint: print numbers, in the current mode (text or math) in order to use the correct font, with separators, exponent and/or rounded to a given number of digits. tabular(*), array, tabularx, and longtable environments are supported using all features of numprint

- spreadtab: spread sheets allowing the use of formulae

- siunitx: alignment of tabular entries

- pgfplotstable: Loads, rounds, formats and postprocesses numerical tables.

2.10.18   References

[1] http://tex.stackexchange.com/questions/121841/is-the-tabu-package-obsolete

[2] D Carlisle. "Decimals in table don't align with dcolumn when bolded". Stackexchange. http://tex.stackexchange.com/questions/118458/decimals-in-table-dont-align-with-dcolumn-when-bolded.

[3] Package multirow on CTAN

2.11   Title Creation

For 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. We will detail the process here.

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 classes. 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.

2.11.1   Standard Titles

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

Commonly the date is excluded from the title page by using \date{}. It defaults to \today if omitted in the source file.

To form a title, use

This should go after the preceding commands after beginning the document. For most document classes, this will form a separate page, while the article document class will place the title on the top of the first page. If you want to have a separate title page for articles as well, use the documentclass option titlepage command. For example, one may add

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

It is dependent on the document class which commands are used in the title generated by \maketitle. Referring to the documentation will lead to trusted information.

2.11.2   The title for journal submission

Journals follow a specific layout. To ensure this they often provide a template which defines the layout. What is available for the title (for example emails, affiliation names, keywords) heavily depends on the template and highly differs between different journals. Follow the

template if the journal provides one. If they don't you should use the most basic concepts of LaTeX titles described above.

2.11.3   Create a custom title for a report or book

The title page of a book or a report is the first thing a reader will see. Keep that in mind when preparing your title page.

You need to know very basic LaTeX layout commands in order to get your own title page perfect. Usually a custom titlepage does not contain any semantic markup, everything is hand crafted. Here are some of the most often needed things:

Alignment

if you want to center some text just use \centering. If you want to align it differently you can use the environment \raggedleft for *right*-alignment and \raggedright for *left*-alignment.

Images

the command for including images (a logo for example) is the following : \includegraphics[width=0.15\textwidth]{./logo}. There is no \begin{figure} as you would usually use since you don't want it to be floating, you just want it exactly where want it to be. 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 braces, *{like this}*, and you can use the following commands (in order of size): \Huge, \huge, \LARGE, \Large, \large, \normalsize, \small, \footnotesize, \tiny. So for example:

Remember, if you have a block of text in a different size, even if it is a bit of text on a single line, end it with \par.

Filling the page

the command \vfill as the last item of your content will add empty space until the page is full. If you put it within the page, you will ensure 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. Here is a practical and compilable example. The picture in use comes with package mwe and should be available with every complete LaTeX installation. You can start testing right away.

As you can see, the code looks "dirtier" than standard La-TeX source because you have to take care of the output as well. If you start changing fonts it gets even more complicated, 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.

The result is shown below
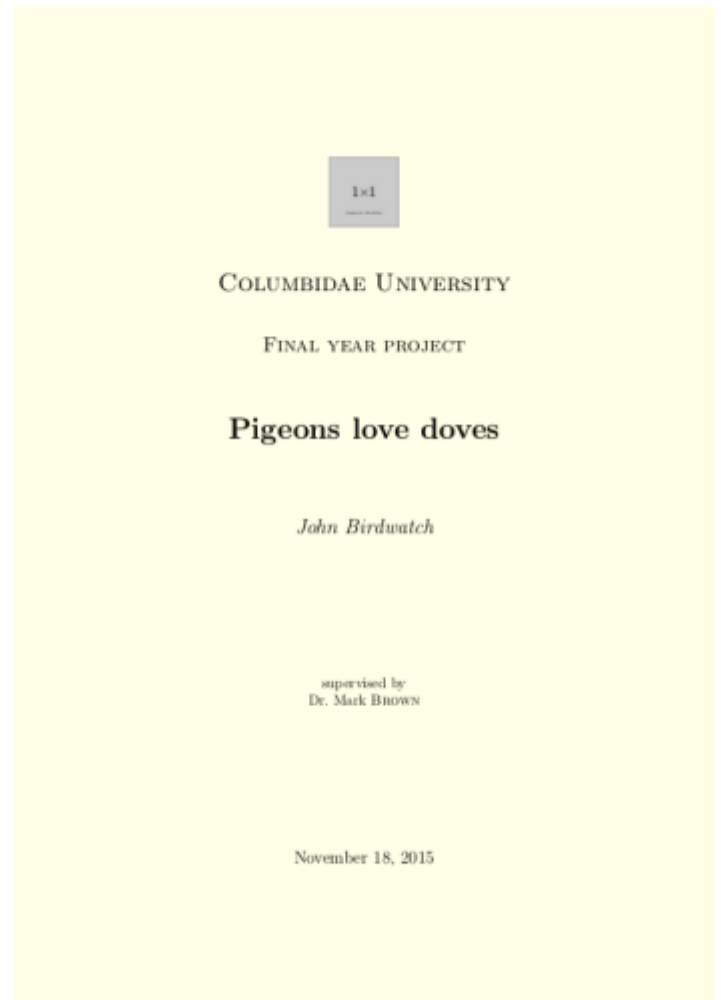
Integrating the title page   A title page for a book or a report to get a university degree {Bachelor, Master, Ph.D., etc.) is quite static, it doesn't really change over time. You can prepare the titlepage in its own little document and prepare a one page pdf that you later include into your real document. This is really useful, if the title page is required to have completely different margins compared to the rest of the document. It also saves compile time, though it is not much.

Assuming you have done the title page of your report in an extra document, let's pretend it is called reportTitlepage2016.pdf, you can include it quite simply. Here is a short document setup.

2.11.4   A title to be re-used multiple times

Some universities, departments and companies have strict rules how a title page of a report should look like. To ensure the very same output for all reports, a redefiniton of the \maketitle command is recommended.

This is best done by an experienced LaTeX user. A simple example follows, as usual there is no real limit with respect to complexity.

As a starting point, a LaTeX package called columbidaeTitle.sty is generated that defines the complete title matter. It will later be hidden from the end user. Ideally, the person creating the package should maintain it for a long time, create an accompanying documentation and ensure user support.

This package can be loaded within a usual document. The user can set the variables for title and the like. Which commands are actually available, and which might be omissible should be written in a documentation that is bundled with the package.

Look around what happens if you leave one or the other command out.

2.11.5   Packages for custom titles

The titling package[1] provides control over the typesetting of the \maketitle and \thanks commands. It is useful for small changes to the standard output.

Italian users may also want to use the frontespizio package[2]. It defines a frontispiece as used in Italia.

Package authblk [3] provides new means to typeset the authors. This is especially helpful for journal submissions without an available template.

2.11.6   More titlepage examples

The titlepages package presents many different styles for title pages.

TeX.SE has a collection of titlepages.

Another small collection can be found on Github.

2.11.7   Notes and References

[1] Titling package webpage on CTAN

[2] Frontespizio package webpage in CTAN

[3] package webpage on CTAN

2.12   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.

We will often have to deal with TeX lengths in this chapter. You should have a look at Lengths for comprehensive details on the topic.

2.12.1   Two-sided documents

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. The most notable effect can be seen in page margins. If you want to make the *article class two-sided*, use \documentclass[twoside]{article}.

Many commands and variables in LaTeX take this concept into account. They are referred to as *even* and *odd*. For one-sided document, only the *odd* commands and variables will be in effect.

2.12.2   Page dimensions

A page in LaTeX is defined by many 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. All sizes are given in TeX points (pt), there are 72.27pt in an inch or 1pt ≈ 0.3515mm.

1. one inch + \hoffset

2. one inch + \voffset

3. \oddsidemargin = 31pt

4. \topmargin = 20pt

5. \headheight = 12pt

6. \headsep = 25pt

7. \textheight = 592pt

8. \textwidth = 390pt

9. \marginparsep = 10pt

10. \marginparwidth = 35pt

11. \footskip = 30pt

- \marginparpush = 7pt (not shown)

- \hoffset = 0pt
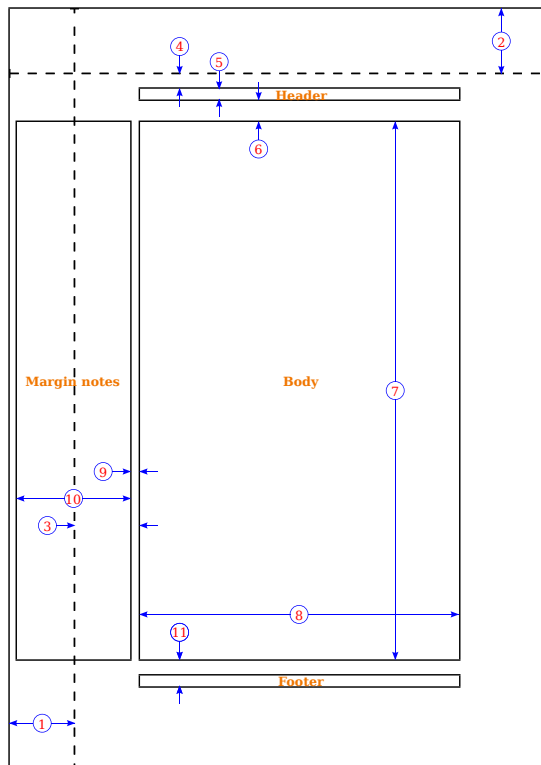
- \voffset = 0pt

- \paperwidth = 597pt

- \paperheight = 845pt

The current details plus the layout shape can be printed from a LaTeX document itself. Use the *layout* package and the command of the same name: \usepackage{layout} ... \layout{}

To render a frame marking the margins of a document you are currently working on, add

\usepackage{showframe}

to the document.

2.12.3 Page size

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). While this is not a serious issue (most printers will print the document without any problems), it is possible to specify alternative sizes as class option. For A4 format:

More size options with *geometry*

One of the most versatile packages for page layout is the geometry package. The immediate advantage of this package is that it lets you customize the page size even with classes that do not support the options. For instance, to set the page size, add the following to your preamble:

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:

Page size issues

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

- TeX → PDF

pdflatex myfile # TeX → PDF

- TeX → DVI → PDF

latex myfile # TeX → DVI dvipdf myfile # DVI → PDF

- TeX → DVI → PS → PDF

latex myfile # TeX → DVI dvips myfile -o myfile.ps # DVI → PS ps2pdf myfile.ps myfile.pdf # PS → PDF

Sadly the PDF output page size may not be completely respectful of your settings. Some of these tools do not have the same interpretation of the DVI, PS and PDF specifications, and you may end up with a PDF which has not exactly the right size. Thankfully there is a solution to that: the \special command lets the user pass PostScript or PDF parameters, which can be used here to set the page size appropriately.

- For pdflatex to work fine, using the package geometry usually works.

- For the DVI and PS ways, the safest way to always get the right paper size in the end is to add

to the tex file, and to append the appropriate parameters to the processors used during output generation:

dvips -t a4 ... ps2pdf -sPAPERSIZE=a4 ... # On Windows: ps2pdf -sPAPERSIZE#a4 ... [1]

If you want US Letter instead, replace 210mm,297mm by 8.5in,11in and a4paper by letter. Also replace a4 by letter in command-line parameters.

Page size for tablets

Those who want to read on tablets or other handheld digital devices need to create documents without the extra whitespace. In order to create PDF documents with optimal handheld viewing, not only must the text field and margins be adjusted, so must the page size. If you are looking for a sensible dimension, consider following the paper size used by the Supreme Court of the United States, 441pt by 666pt (or 6.125 inches by 9.25 inches), which looks great on tablets. You could also use the Supreme Court's text field size of 297 pt by 513 pt, but this is too wide for fonts other than Century Schoolbook, the font required by the Supreme Court.

2.12.4 Margins

Readers used to perusing typical physical literature are probably wondering why there is so much white space surrounding the text. For example, on A4 paper a document will typically have 44 mm margin widths on the left and right of the page, leaving about 60% of the page width for text. The reason is improved readability. Studies have shown[2][3] 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 optimal readability, and excessive margin white space is tolerated as a consequence. Sometimes, this white space is left in the inner margin with the assumption that the document will be bound.

If you wish to avoid excessive white space, rather than changing the margins, consider instead using a two-column (or more) layout. This approach is the one usually taken by print magazines because it provides both readable line lengths and good use of the page. Another option for reducing the amount of whitespace on the page without changing the margins is to increase the font size using the 12pt option to the document class.

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

- One older approach is to use the fullpage package for somewhat standardized smaller margins (around an inch), but it creates lines of more than 100 characters per line at with the 10pt default font size (and about 90 if the 12pt documentclass option is used):

For even narrower margins, the fullpage package has a cm option (around 1.5cm), which results in about 120 characters per line at the 10pt default font size, about double what is considered readable:

- A more modern and flexible approach is to 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). Note that by default (*i.e.* without any options) this package already reduces the margins, so for a 'standard layout' you may not need to specify anything. These values are relative to the edge of paper (0in) and go inward. For example, this command provides more conventional margins, better using the vertical space of the page, without creating the dramatically long lines of the fullpage package (if the 11pt documentclass option is used, the line lengths are about 88 characters for letter-sized paper and slightly less when using a4paper).

It can also recreate the behavior of the fullpage package using

You can combine the margin options with the page size options seen in this paragraph.

- You should not use the a4wide package for a page with A4 document size with smaller margins. It is obsolete and buggy. Use geometry package instead like this:

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

Odd and even margins

Using the geometry package, the options left and right are used for the inside and outside margins respectively. They also have aliases inner and outer. Thus, the easiest way to handle different margins for odd and even pages is to give the twoside option in the document class command and specify the margins as usually.

This will result in a value of 4cm on all inner margins (left margin for odd number pages and right margin for even pages) and 2cm margin on outer margins.

Setting the same value for the inner and outer for geometry will remove the difference between the margins. Another quick way to eliminate the difference in position between even and odd numbered pages would be setting the values to **evensidemargin** and **oddsidemargin** to the half of odd's default:

By default, the value of **evensidemargin** is larger than **oddsidemargin** in the two-sided layout, as one could wish to write notes on the side of the page. The side for the large margin is chosen opposite to the side where pages are joined together.

See the Lengths.

Top margin above Chapter

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

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

2.12.5   Page orientation

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

Change orientation of the whole document

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 we prefer is the geometry package. All you need to do is call the package, with *landscape* as an option:

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:

Using standard LaTeX classes, you can use the same class options:

Change orientation of specific part

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 centered on a landscaped page, one must place the tabular environment inside a table environment, which is itself inside the landscape environment. For instance it should look like this:

For books (and in general documents using the twoside option), the landscape-environment unfortunately does not pay attention to the different layout of even and odd pages. The macro can be fixed using a few lines of extra code in the preamble[4].

Change orientation of floating environment

If you use the above code, you will see that the table is inserted where it is in the code. It will not be floated! To fix this you need the package rotating. See the Rotations chapter.

2.12.6   Margins, page size and rotation of a specific page

If you need to rotate the page so that the figure fits, the chances are good that you need to scale the margins and the font size too. Again, the geometry package comes in handy for specifying new margins for a single page only.

Note that order matters!

2.12.7   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:

The commands \markright and \markboth can be used to set the content of the headings by hand. The following commands placed at the beginning of an article document will set the header of all pages (one-sided) to contain "John Smith" top left, "On page styles" centered and the page number top right:

There are special commands containing details on the running page of the document.

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:

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:

Watch out: if you provide long text in two different "parts" only in the footer or only in the header, you might see overlapping text.

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

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

Plain pages issue   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. Two solutions:

- 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.

- Use fancyhdr as described below.

The tricky problem when customizing headers and footers is to get things like running section and chapter names in there. Standard 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, except if they are starred -- in such a case, \markboth{Chapter/Section name}{} must be used inside the sectioning command if header and footer lines are to be updated.

Again, several packages provide a solution:

- an alternative one-stage mechanism is provided by the package titleps);

- fancyhdr will handle the process its own way.

Customizing 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.

To begin, add the following lines to your preamble:

You can now observe a new style in your document.

The \headheight needs to be 13.6pt or more, otherwise you will get a warning and possibly formatting issues. Both the header and footer comprise three elements each according to its horizontal position (left, centre or right).

The styles supported by fancyhdr:

- the four LaTeX styles;

- fancy defines a new header for all pages but *plainstyle* pages such as chapters and titlepage;

- fancyplain is the same, but for absolutely all pages.

Style customization   The styles can be customized with fancyhdr specific commands. Those two styles may be configured directly, whereas for LaTeX styles you need to make a call to the \fancypagestyle command.

To set header and footer style, fancyhdr provides three interfaces. They all provide the same features, you just use them differently. Choose the one you like most.

- You can use the following six commands.

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 **l**eft, **c**entre and **r**ight respectively.

- You can also use 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:

You can use multiple selectors optionally separated by a comma. The selectors are the following:

so CE,RO will refer to the center of the even pages and to the right side of the odd pages.

- \fancyhf is a merge of \fancyhead and \fancyfoot, hence the name. There are two additional selectors H and F to specify the header or the footer, respectively. If you omit the H and the F, it will set the fields for both.

These commands will only work for fancy and fancyplain. To customize LaTeX default style you need the \fancyplainstyle command. See below for examples.

For a clean customization, we recommend you start from scratch. To do so you should *erase* the current pagestyle. Providing empty values will make the field blank. So

will just delete the current heading/footer configuration, so you can make your own.

Plain pages   There are two ways to change the style of plain pages like chapters and titlepage.

First you can use the fancyplain style. If you do so, you can use the command \fancyplain{...}{...} inside fancyhdr commands like \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. For instance:

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.

Alternatively 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:

In that case you can use any style but fancyplain because it would override your redefinition.

Examples   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:

This is effectively saying author name is top outer, today's date is top inner, and current page number is bottom outer. Using \fancyhf can make it shorter:

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

Using \fancypagestyle one can additionally define multiple styles for one's document that are easy to switch between. Here's a somewhat complicated example for a two-sided book style:

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:

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.

Alternative packages

Other packages for page styles are scrpage2, very similar to fancyhdr, and titleps, which takes a one-stage approach, without having to use \leftmark or \rightmark.

2.12.8   Page background

The eso-pic package will let you print content in the background of every page or individual pages.

The starred-version of the \AddToShipoutPicture command applies to the current page only.

2.12.9   Multi-column pages

Using the *twocolumn* optional class argument

Using a standard Latex document class, like article, you can simply pass the optional argument *twocolumn* to the document class: \documentclass[twocolumn]{article} which will give the desired effect.

While this approach is useful, it has limitations. The multicol package provides the following 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.

Using multicol package

The multicol package overcomes some of the shortcomings of *twocolumn* and provides the multicol environment. To create a typical two-column layout:

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. See this section for a more detailed discussion.

The multicol package has two important parameters which can be set using \setlength:

- \columnseprule, sets the width of the vertical rule between columns and defaults to 0pt

- \columnsep, sets the horizontal space between columns and the defaults to 10pt, which is quite narrow

To force a break in a column, the command \columnbreak is used.

2.12.10 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:

2.12.11 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:

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:

Alternatively, you can use the needspace package to *reserve* some lines and thus to prevent page breaking for those lines.

2.12.12 Troubleshooting

A very useful troubleshooting and designing technique is to turn on the showframe option in the geometry package (which has the same effect as the showframe package described above). It draws bounding boxes around the major page elements, which can be helpful because the boundaries of various regions are usually invisible, and complicated by padding whitespace.

2.12.13 Notes and References

This page uses material from Andy Roberts' Getting to grips with LaTeX with permission from the author.

[1] How to use Ghostscript

[2] http://webtypography.net/2.1.2

[3] http://baymard.com/blog/line-length-readability

[4] https://stackoverflow.com/questions/4982219/ how-to-make-landscape-mode-rotate-properly-in-a-twoside-book/ 5320962#5320962

2.13 Importing Graphics

There are two possibilities to include graphics in your document. Either create them with some special code, a topic which will be discussed in the *Creating Graphics* part, (see Introducing Procedural Graphics) or import productions from third party tools, which is what we will be discussing here.

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.

2.13.1 Raster graphics vs. vector graphics

Raster graphics will highly contrast with the quality of the document if they are not in a high resolution, which is the case with most graphics. The result may be even worse once printed.

Most drawing tools (e.g. for diagrams) can export in vector format. So you should always prefer PDF or EPS to PNG or JPG.

2.13.2 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:

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.

2.13.3   Document Options

The graphics and graphicx packages recognize the draft and final options given in the \documentclass[...]{...} command at the start of the file. (See Document Classes.) 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 final.

2.13.4   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.

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).

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.

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.

- **EPS** can be used with the help of the epstopdf package. Depending on your installation,

    - you may just need to have it installed, there is no need to load it in your document;

    - if it does not work, you need to load it just after the graphicx package. Additionally, since epstopdf will need to convert the EPS file into a PDF file and store it, you need to give "writing permissions" to your compiler. This is done by adding an option to the compiling command, *e.g.* pdflatex -shell-escape file.tex (if you use a LaTeX editor, they usually allow to modify the command in the configuration options). Check the epstopdf documentation for other compilers.

2.13.5 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:

As usual, 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 La-TeX 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). 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:

which specifies the files to include in the document (in order of preference), if files with the same basename exist, but with different extensions.

The variety of possible attributes that can be set is fairly large, so only the most common are covered below:

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 floats 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 XeLa-Tex.

Be careful using any options, if you are working with the chemnum-package. The labels defined by \cmp-dref{<label name>} might not behave as expected. Scaling the image for instance may be done by \scalebox instead.

The *star* version of the command will work for .eps files only. For a more portable solution, the standard way should take precedence. The star command will take the crop dimension as extra parameter:

Examples

OK, it's time to see graphicx in action. Here are some examples. Say you had a file 'chick.jpg' you would include it like:

This simply imports the image, without any other pro-

cessing. However, it is very large (so we won't give an example of how it would look here!) So, let's scale it down:

This has now scaled 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:

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):

To rotate (I also scaled the image down):

And finally, an example of how to crop an image should you wish to focus on one particular area of interest:

Note: the presence of clip, as the trim operation will not work without it.

Trick: You can also use negative trim values to add blank space to your graphics, in cases where you need some manual alignment.

Spaces in names

If the image file were called "chick picture.png", then you need to include the full filename when importing the image:

One option is to not use spaces in file names (if possible), or to simply replace spaces with underscores ("chick picture.png" to "chick_picture.png")

Borders

It is possible to have LaTeX create a border around your image by using \fbox:

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.

See Boxes for more details on \framebox and \fbox.

2.13.6 Graphics storage

The command \graphicspath tells LaTeX where to look for images, which can be useful if you store images centrally for use in many different documents. The \graphicspath command takes one argument, which specifies the additional paths you want to be searched when the \includegraphics command is used. Here are some examples (trailing / is required):

Notice that, even if there is only one path given, there are two curly brackets around the path name.

Please see http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.pdf. In the third example shown there should be a directory named "images" in the same directory as your main tex file, i.e. this is RELATIVE addressing.

Using absolute paths, \graphicspath makes your file less portable, while using relative paths (like the third example), there should not be any problem with portability. The fourth example uses the "safe" (MS-DOS) form of the Windows *MyPictures* folder because it's a bad idea to use directory names containing spaces. Again, ensure file names do not contain spaces or alternatively if you are using PDFLaTeX, you can use the package grffile which will 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.)

2.13.7   Images as figures

The figure environment is not exclusively used for images. We will only give a short preview of figures here. More information on the figure environment and how to use it can be found in Floats, Figures and Captions.

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.

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.

2.13.8   Text wrapping around pictures

See Floats, Figures and Captions.

2.13.9   Seamless text integration

The drawback of importing graphics that were generated with a third-party tool is that font and size will not match with the rest of the document. There are still some workarounds though.

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:

Note that the border around the picture in the above example was added by using \fbox, so the contents of the border is the picture as generated by the above code.

Tools like Inkscape or Xfig have a dedicated LaTeX export feature that will let you use correct font and size for text in vector graphics. See #Third-party graphics tools.

For a perfect integration of graphics, you might consider procedural graphics capabilities of some LaTeX packages like TikZ or PSTricks. It lets you *draw* from within a document source. While the learning curve is steeper, it is worth it most of the time.

2.13.10   Converting graphics

Note

You should also take a look at Export To Other Formats for other possibilities.

epstopdf

You can convert EPS to PDF with the epstopdf utility, included in package of the same name. This tool is actually called by pdflatex to convert EPS files to PDF in the background when the graphicx package is loaded. This process is completely invisible to the user.

You can batch convert files using the command-line. In Bourne Shell (Unix) this can be done by:

```
$ for i in *.eps; do epstopdf "$i"; done
```

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 produces whole page with your small graphics somewhere on it, use

```
$ epstopdf --gsopt=-dEPSCrop foo.eps
```

or try using ps2pdf utility which should be installed with Ghostscript (required for any TeX distribution).

```
$ ps2pdf -dEPSCrop foo.eps
```

to crop final PDF.

eps2eps

When all of the above fails, one can simplify the EPS file before attempting other conversions, by using the eps2eps 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.

imgtops

imgtops is a lightweight graphics utility for conversions between raster graphics (JPG, PNG, ...) and EPS/PS files.

Inkscape

Inkscape can also convert files from and to several formats, either from the GUI or from the command-line. For instance, to obtain a PDF from a SVG image you can do:

$ inkscape -z -D --file=input.svg --export-pdf=output.pdf

It is possible to run this from within a LaTeX file, the Template:LaTeX/package package (when running (pdf)latex with the --shell-escape option) can do this using Inkscape's pdf+tex export option, or a simple macro can be used. See How to include SVG diagrams in La-TeX? -- Stackexchange See Export To Other Formats for more details.

pstoedit

To properly edit an EPS file, you can convert it to an *editable* format using pstoedit. For instance, to get an Xfig-editable file, do:

$ pstoedit -f fig input.eps output.fig

And to get an SVG file (editable with any vector graphics tool like 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).

PDFCreator

Under Windows, PDFCreator 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).

Raster graphics converters

- Sam2p (convert) or
- ImageMagick (convert) or
- GraphicsMagick (gm convert).

These three programs operate much the same way, and can convert between most graphics formats. Sam2p how-ever is the most recent of the three and seems to offer both the best quality and to result in the smallest files.

PNG alpha channel

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

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 of the Ghostscript package and programs:

$ eps2eps input.eps input-e2.eps $ pscol −0gray input-e2.eps input-gray.eps

2.13.11 Third-party graphics tools

We will not tackle the topic of procedural graphics created from within LaTeX code here (TikZ, PSTricks, MetaPost and friends). See Introducing Procedural Graphics for that.

You should prefer vector graphics over raster graphics for their quality. Raster graphics should only be used in case of photos. Diagrams of any sort should be vectors.

As we have seen before, LaTeX handles

- EPS and PDF for vector graphics;
- PNG and JPG for raster graphics.

If some tools cannot save in those formats, you may want to convert them before importing them.

Vector graphics

Dia

Dia is a cross platform diagramming utility which can export eps images, or generate tex drawn using the tikz package.

Inkscape

Another program for creating vector graphics is Inkscape. It can run natively under Windows, Linux or Mac OS X (with X11). It works with Scalable Vector Graphics (SVG) 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, similar to that offered by Xfig. There are instructions 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.[1]. Today there is the svg package[2] which provides an \includesvg command to convert and include svg-graphics directly in your LaTeX document using Inkscape. You may have a look at this extended example too.

An extremely useful plug-in is textext, which can import LaTeX objects. This can be used for inserting mathematical notation or LaTeX fonts into graphics (which may then be imported into LaTeX documents).

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.

lpic

Yet another solution is provided by the lpic packages , which allows TeX annotations to imported graphics. See Labels in the figures.

OpenOffice.org

It is also possible to export vector graphics to EPS format using OpenOffice.org Draw, which is an open source office suite available for Windows, Linux and Mac.

TpX

Vector editor TpX separates geometric objects from text objects. Geometric objects are saved into .PDF file, the rest is saved in .TpX file to be processed by LaTeX. User just create the graphics in TpX editor and calls the .TpX file from latex file by command \input{...TpX}.

Xfig

Xfig is a basic program that can produce vector graphics, which can be exported to LaTeX. It can be installed on Unix platforms.

On Microsoft Windows systems, Xfig can only be installed using Cygwin-X; however, this will require a fast internet connection and about 2 gigabytes of space on your computer. With Cygwin, to run Xfig, you need to first start the "Start X - Server", then launch "xterm" to bring up a terminal. In this terminal type "xfig" (without the quotation marks) and press return.

Alternatively, WinFIG is an attempt to achieve the functionality of xfig on Windows computers.

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:

   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:

And you're done!

For more details on using xfig with LaTeX, this chapter of the xfig User Manual may prove helpful.

Other tools

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 X platforms.

Raster graphics

Adobe Photoshop

It can save to EPS.

GIMP

GIMP, has a graphical user interface, and it is multi-platform. It can save to EPS and PDF.

Plots and Charts

Generic Mapping Tools (GMT)

Generic Mapping Tools (GMT), maps and a wide range of highly customisable plots.

Gnumeric

Gnumeric, spreadsheets has SVG, EPS, PDF export

Gnuplot

Gnuplot, producing scientific graphics since 1986. If you want to make mathematical plots, then Gnuplot can save in any format. You can get best results when used along PGF/TikZ.

matplotlib

matplotlib, plotting library written in python, with PDF and EPS export. On the other hand there is a PGF export also. 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.

R

R, statistical and scientific figures.

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, changing the bounding box, or moving items slightly, but anything further becomes cumbersome. Vector graphics editors, like Inkscape, may also be able to import EPS files for subsequent editing. This approach also for easier editing. However, the importing process may occasionally modify the original EPS image.

2.13.12 Notes and References

[1] Johan B. C. Engelen. "How to include an SVG image in LATEX". mirrorcatalogs.com. http://ctan.mirrorcatalogs.com/info/svg-inkscape/InkscapePDFLaTeX.pdf.

[2] Philip Ilten. "The svg package on CTAN". ctan.org. http://www.ctan.org/tex-archive/graphics/svg.

2.14 Floats, Figures and Captions

The previous chapter introduced importing graphics. However, just having a picture stuck in between paragraphs does not look professional. To start with, 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.

2.14.1 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.

As with various other entities, there exist limitations on the number of floats. LaTeX by default can cope with maximum 18 floats and a symptomatic error is:

! LaTeX Error: Too many unprocessed floats.

The morefloats package lifts such limit.

Figures

To create a figure that floats, use the figure environment.

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.

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. Frank Mittelbach describes the algorithm[2]:

- If a float is encountered, LaTeX attempts to place it immediately according to its rules (detailed later)

  - if this succeeds, the float is placed and that decision is never changed;

  - if this does not succeed, then LaTeX places the float into a holding queue to be reconsidered when the next page is started (but not earlier).

- Once a page has finished, LaTeX examines this holding queue and tries to empty it as best as possible. For this it will first try to generate as many float pages as possible (in the hope of getting floats off the queue). Once this possibility is exhausted, it will next try to place the remaining floats into top and bottom areas. It looks at all the remaining floats and either places them or defers them to a later page (i.e., re-adding them to the holding queue once more).

- After that, it starts processing document material for this page. In the process, it may encounter further floats.

- If the end of the document has been reached or if a \clearpage is encountered, LaTeX starts a new page, relaxes all restrictive float conditions, and outputs all floats in the holding queue by placing them on float page(s).

In some special cases LaTeX won't follow these positioning parameters and additional commands will be necessary, for example, if one needs to specify an alignment other than centered for a float that sits alone in one page[3].

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:

The border will not include the caption.

Tables

Floating tables are covered in a separate chapter. Let's give a quick reminder here. 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:

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.

2.14.2   Keeping floats in their place

The placeins package provides the command \FloatBarrier, which can be used to prevent floats from being moved over it. This can, e.g., be useful at the beginning of each section. The package even provides an option to change the definition of \section to automatically include a \FloatBarrier. This can be set by loading the package with the option [section] (\usepackage[section]{placeins}). \FloatBarrier may also be useful to prevent floats intruding on lists created using itemize or enumerate.

The flafter package can be used to force floats to appear after they are defined, and the endfloat package can be used to place all floats at the end of a document.

The float package provides the H option to floating environments, which completely stops them from floating.

Package caption provides the command \captionof{<type>}{<caption text>} that lets you typeset a caption without a floating environment. You have the full and absolute control about the placement of your figures and captions.

2.14.3   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. LaTeX 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:

Note that the command \reflectbox{...} flips its content horizontally.

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. The floatrow package is newer and has more capabilities.

Unnumbered captions

In some types of document (such as presentations), it may not be desirable for figure captions to start *Figure:*. This is easy to suppress by just placing the caption text in the figure environment, without enclosing it in a caption. This however means that there is no caption available for inclusion in a list of figures.

Renaming table caption prefix

In case you want to rename your table caption from "Table" to something else, you can use the \captionsetup command. For example,

2.14.4 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:

2.14.5 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.

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. If you get an error when the label is inside the caption, use \protect in front of the \label command.

2.14.6 Wrapping text around figures

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 works 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 that wrapfig may not come with the default installation of LaTeX; you might need to install additional packages. Noted also, wrapfig is incompatible with the enumerate and itemize environments

To use wrapfig, you must first add this to the preamble:

This then gives you access to:

The lineheight is expressed as the number of lines of text the figure spans. LaTeX will automatically calculate the value if this option is left blank but this can result in figures that look ugly (with too much spacing). The LaTeX calculation is manually overridden by entering the number of lines you would like the figure to span. This option can't be entered in pt, mm etc...

There are overall eight possible positioning targets:

The uppercase-character allows the figure to float, while the lowercase version means "exactly here". [4]

The overhang of the figure can be manually set using the overhang option in pt, cm etc...

The *width* is, of course, the width of the figure. An example:

You can also allow LaTeX to assign a width to the wrap by setting the width to 0pt. \begin{wrapfigure}{l}{0pt}

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

You can use intextsep parameter to control additional space above and below the figure: \setlength\intextsep{0pt}

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. Also remember that the environment center adds some extra white space at its top and bottom; consider using the command \centering instead.

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:

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 wrapfig 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.

To remove the white space from a figure once for all, one should refer to the program pdfcrop, included in most TeX installations.

2.14.7 Subfloats

A useful extension is the subcaption package, which uses subfloats within a single float. The subfigure and subfig packages are deprecated; however they are useful alternatives when used in-conjunction with LaTeX templates (i.e templates for journals from Springer and IOP, IEEE-Tran and ACM SIG) that are not compatible with subcaption. These packages give 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 the subcaption package:

(a) A gull   (b) A tiger   (c) A mouse
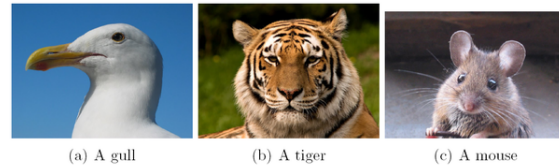
Figure 1: Pictures of animals

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:

If you intend to cross-reference any of the subfloats, see where the label is inserted; \caption outside the subfigure-environment will provide the global caption.

subcaption 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 are controlled by one of several commands, which are placed in between \begin{subfigure} and \end{subfigure}:

- A non-breaking space (specified by ~ as in the example above) can be used to insert a space in between the subfigs.

- Math spaces: \qquad, \quad, \;, and \,

- Generic space: \hspace{"length"}

- Automatically expanding/contracting space: \hfill

2.14.8 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 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{dblfloatfix} 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}).

2.14.9 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, but the caption is always below the content.

   - plaintop - the normal style for LaTeX floats, but the caption is always above the content.

   - 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:

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:

Caption styles

To change the appearance of captions, use the caption package. For example, to make all caption labels small and bold:

The KOMA script packages 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.

Alternatively, you can redefine the \thefigure command:

See this page for more information on counters. Finally, note that the caption2 package has long been deprecated.

2.14.10 Labels in the figures

There is a LaTeX package lpic 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 . The link given also points to the packages psfrag and overpic.

A much more complicated package which can be used

in the same way is TikZ. TikZ is a front-end to a draw-ing library called pgf (which is used to make beamer for instance). It can be used to label figures by adding text nodes on top of an image node.

2.14.11   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.

2.14.12   Notes and references

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

[2] Float environment positioning, by Frank Mittelbach

[3] http://tex.stackexchange.com/questions/28556/how-to-place-a-float-at-the-top-of-a-floats-only-page

[4] http://ftp.univie.ac.at/packages/tex/macros/latex/contrib/wrapfig/wrapfig-doc.pdf

[5] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=2colfltorder

This page uses material from Andy Roberts' Getting to grips with LaTeX with permission from the author.

2.15   Hyperlinks

LaTeX enables typesetting of hyperlinks, useful when the resulting format is PDF, and the hyperlinks can be fol-lowed. It does so using the package hyperref.

2.15.1   Hyperref

The package hyperref[1] 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 interac-tive external links and all your internal references will be turned to hyperlinks. The compiler *pdflatex* makes it pos-sible to create PDF files directly from the LaTeX source, and PDF supports more features than DVI. In particular PDF supports hyperlinks. Moreover, PDF can contain other information about a document such as the title, the author, etc., which can be edited using this same package.

2.15.2   Usage

The basic usage with the standard settings is straightfor-ward. Just load the package in the preamble:

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 (discussed in the chapter on Labels and Cross-referencing); with hyperref those "connections" will be-come links and you will be able to click on them to be redirected to the right page. Moreover the table of con-tents, list of figures/tables and index will be made of hy-perlinks, too. The hyperlinks will not show-up if you are working in draft mode.

Commands

The package provides some useful commands for insert-ing links pointing outside the document.

\hyperref   Usage:

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. If the lemma labelled as mainlemma was number 4.1.1 the following example would result in

with the hyperlink as expected. Note the "*" after \ref for avoiding nested hyperlinks.

\url   Usage:

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:

It will show the string description using standard docu-ment font but, if you click on it, your browser will be opened pointing at my_url. Here is an example:

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 doc-ument.

Other possibilities

Apart from linking to websites discussed above, hyperref can be used to provide *mailto* links, links to local files, and links to anywhere within the PDF output file.

E-mail address   A possible way to insert email links is by

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[2]

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:

Following http://tex.stackexchange.com/questions/ 46488/link-to-local-pdf-file the version with url does not always work, but href does.

It is possible to 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. To create an anchor, use:

and to link to it, use:

where the target caption and link caption are the text that is displayed at the target location and link location respectively.

2.15.3 Customization

The standard settings should be fine for most users, but if you want to change something, that is also possible. There are several variables and two methods to pass those to the package. Options can be passed as an argument of the package when it is loaded (the standard way packages work), or the \hypersetup command can be used as follows:

you can pass as many options as you want; separate them with a comma. Options have to be in the form:

exactly the same format has to be used if you pass those options to the package while loading it, like this:

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:

Checkout 3.8 Big list at hyperref-manual at tug.org

Please note, that explicit 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 doc-

ument and make the changes you want. Next to the variables, there is a short explanations of their meaning:

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:

or make links 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:

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:

2.15.4 Troubleshooting

Problems with Links and Equations 1

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

The error disappears, if you use instead this form:

Beware that the shown line number is often completely different from the erroneous line.

Possible solution: Place the amsmath package before the hyperref package.

Problems with Links and Equations 2

Messages like the following

! Runaway argument? {\@firstoffive }\fi), Some text from your document here (\ref {re\ETC. Latex Error: Paragraph ended before \Hy@setref@link was complete.

appear when you use \label inside an align environment.

Possible solution: Add the following to your preamble:

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:

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:

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 important in a multicolumn environment where the line width 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 PDF-DocEncoded string:

You can now work around this problem by providing a text string for the bookmarks, which replaces the offending text:

Math expressions are a prime candidate for this kind of problem:

which turns \section{$E=mc^2$} to E=mc2 in the bookmark area. Color changes also do not travel well into bookmarks:

produces the string "redRed!". The command \textcolor gets ignored but its argument (red) gets printed. If you use:

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[4]. A workaround exists by using the package hypcap with:

Be sure to call this package *after* loading hyperref.

If you use the wrapfig package[5] 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.*:

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:

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.

Problems with footnotes and special characters

See the relevant section.

Problems with Beamer

Using the command

is broken when pointed at a label. Instead of sending the user to the desired label, upon clicking the user will be sent to the first frame. A simple work around exists; instead of using

to label your frames, use

and reference it with

Problems with draft mode

WARNING! Please note that if you have activated the "draft"-option in your \documentclass declaration the hyperlinks will not show up in the table of contents, or anywhere else for that matter!!!

The hyperlinks can be re-enabled by using the "final=true" option in the following initialization of the hyperref package, just after the package was included:

A good source of further options for the hyperref package can be found here [6].

2.15.5 Notes and References

[1] Hyperref package webpage in CTAN

[2] "Email link with hyperref, url packages". *comp.text.tex User Group.* http://groups.google.com/group/comp. text.tex/browse_thread/thread/ae160fd2fc5680a5/ 71a5a7c7bfceb3cb?lnk=gst&q=email+url+hyperref# 71a5a7c7bfceb3cb. Retrieved 2008.

[3] Other possible values are defined in the hyperref manual

[4] http://www.ctan.org/tex-archive/macros/latex/contrib/ hyperref/README

[5] Wrapfig package webpage in CTAN

[6] Hyperref - Hyperlinks With LaTeX Page

2.16 Labels and Cross-referencing

2.16.1 Introduction

In LaTeX 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. Some packages, such as fancyref, rely on this meta information. Here is an example:

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 an 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.

2.16.2   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{fig:gull} \end{figure} Figure \ref{fig:gull} 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{fig:gull}}

See the Floats, Figures and Captions section for more about the figure and related environments.

Figure 1: Close-up of a gull

Figure 1 shows a photograph of a gull.

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 give the option hypcap to the caption package:

\usepackage{caption} % hypcap is true by default so [hypcap=true] is optionnal in \usepackage[hypcap=true]{caption}

Formulae

Here is an example showing how to reference formulae:

\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}

$$x^2 - 5x + 6 = 0$$

$$x_1 = \frac{5 + \sqrt{25 - 4 \times 6}}{2} = 3$$

$$x_2 = \frac{5 - \sqrt{25 - 4 \times 6}}{2} = 2$$

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 formulae, 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 formulae; it is \eqref{}. It works exactly like \ref{}, but it adds parentheses so that, instead of printing a plain number as *5*, it will print *(5)*. This can be useful to help the reader distinguish between formulae and other things, without the need to repeat the word "formula" before any reference. Its output can be changed as desired; for more information see the amsmath documentation.

tag The \tag{eqnno} command is used to manually set equation numbers where *eqnno* is the arbitrary text string you want to appear in the document. It is normally better to use labels, but sometimes hard-coded equation numbers might offer a useful work-around. This may for instance be useful if you want to repeat an equation that is used before, e.g. \tag{\ref{eqn:before}}.

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.

cases The cases package adds the \numcases and the \subnumcases commands, which produce multi-case equations with a separate equation number and a separate equation number plus a letter, respectively, for each case.

2.16.3 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.

2.16.4 The hyperref package

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). 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 of 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{]}\protect\char"007B\ relaxAppendix~{}\char"005C\relax{}ref*\ protect\char"007B\relaxsec: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.

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...

Anchor manual positioning

When you define a \label outside a figure, a table, or other floating objects, the label points to the current section. In some cases, 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}

2.16.5 The cleveref package

The cleveref package introduces the new command \cref{} which includes the type of referenced object like \autoref{} does. The alternate \labelcref{} command works more like standard \ref{}. References to pages are handled by the \cpageref{} command.

The \crefrange{}{} and \cpagerefrange{} commands expect a start and end label in either order and provide a natural language (babel enabled) range. If labels are enumerated as a comma-separated list with the usual \cref{} command, it will sort them and group into ranges automatically.

The format can be specified in the preamble.

2.16.6 Interpackage interactions for varioref , hyperref , and cleveref

Because varioref,hyperref, and cleveref redefine the same commands, they can produce unexpected results when their \usepackage commands appear in the preamble in the wrong order. For example, using hyperref,varioref, then cleveref can cause \vref{} to fail as though the marker were undefined.[2] The following order generally seems to work:

1. varioref

2. hyperref

3. cleveref[2]

2.16.7 See also

- LaTeX/Glossary

2.16.8 Notes and References

[1] http://www.ctan.org/tex-archive/macros/latex/contrib/hyperref/README

[2] Tests done under report class http://tex.stackexchange.com/questions/139459/vref-and-input-command

Chapter 3

Mechanics

3.1 Errors and Warnings

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

Don't panic if you see error messages: it is very common to mistype or misspell 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.

3.1.1 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:

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.

3.1.2 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 unsuc-

cessfully) by interfering with the letter spacing, 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.

3.1.3 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. Some editors allow common commands and environments to be inserted using drop-down menus or icons, which may be used to avoid these errors.

Not in Mathematics Mode

! Missing $ inserted

A character that can only be used in the mathematics was inserted in normal text. If you intended to use mathematics mode, then use $...$ or \begin{math}...\end{math} or use the 'quick math mode': \ensuremath{...}. If you did not intend to use mathematics mode, then perhaps you are trying to use a special character that needs to be entered in a different way; for example _ will be interpreted as a subscript operator in mathematics mode, and you need _ to get an underscore character.

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.

This warning may also appear when inserting images. It can be avoided by using the \textwidth or possibly \linewidth options, e.g.   \includegraphics[width=\textwidth]{image_name}

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.

The warning can also be issued when the \end{document} tag was not included or was deleted.

Easily spotting overfull hboxes in the document   To easily find the location of overfull hbox in your document, you can make latex add a black bar where a line is too wide:

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 La-TeX 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.

Package babel Error: You haven't loaded the option X yet.

If you previously set the X language, and then decided to switch to Y, you will get this error. This may seem awkward, as there is obviously no error in your code if you did not change anything. The answer lies in the .aux file, where babel defined your language. If you try the compilation a second time, it should work. If not, delete the .aux file, then everything will work as usual.

No error message, but won't compile

One common cause of (pdf)LaTeX getting stuck is forgetting to include \end{document}

3.1.4 Software that can check your .tex Code

There are several programs capable of checking LaTeX source, with the aim of finding errors or highlighting bad practice, and providing more help to (particularly novice) users than the built-in error messages.

- nag (www.ctan.org/tex-archive/macros/latex/contrib/nag) is a LaTeX package designed to indicate the use of obsolete commands.

- lacheck (www.ctan.org/tex-archive/support/lacheck) is a consistency checker intended to spot mistakes in code. It is available as source code or compiled for Windows and OS/2

- chktex (baruch.ev-en.org/proj/chktex/) is a LaTeX semantic checker available as source code for Unix-like systems.

3.2 Lengths

In TeX, a length is

- a floating point number followed by a unit, optionally followed by a stretching value;

- a floating point factor followed by a macro that expands to a length.

3.2.1 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]

The point is the default unit and 1pt is the default length. All other units are converted to the point by a fixed ratio.

Here are some less common units.[2]

3.2.2 Box lengths

A box in TeX is characterized by three lengths:

- *depth*

- *height*

- *width*

See Boxes.

3.2.3 Length manipulation

You can change the values of the variables defining the page layout with two commands. With this one you can set a new value for an existing length variable:

with this other one, you can add a value to the existing one:

You can create your own length with the command, and you must create a new length before you attempt to set it:

You may also set a length from the size of a text with one of these commands:

The calc package provides also the function \settototal-height{\mylength}{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 \savebox to avoid this duplication. You may wish to look at the example below to see how you can use these. See Boxes for more details.

You can also define stretched values. A stretching value is a length preceded by plus or minus to specify to what extent tex is authorized to change the length. Example:

It means that tex will try to use a length of 10pt; if it is underfull, it will raise the length up to a maximum of 15pt; if it is overfull, it will lower the length up to a minimum of 7pt.

Note that it is not mandatory to specify both the plus and the minus values, but if you do, plus must be placed before minus.

To print a length, you can use the \the command:

Plain TeX

To create a new length:

To set a length:

To view, it is the same as with LaTeX, using the command \the.

3.2.4 LaTeX default lengths

Common length macros 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.

3.2.5 Fixed-length spaces

To insert a fixed-length space, use:

\hspace stands for horizontal space, \vspace for vertical space.

If such a space should be kept even if it falls at the end or the start of a line, use \hspace* instead.

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. If you want to add space at the beginning of the document, without anything else written before, then you may use

It's important you use the \vspace* command instead of \vspace, otherwise LaTeX can silently ignore the extra space.

TeX features some macros for fixed-length spacing.

\smallskip   Inserts a small space in vertical mode (between two paragraphs).

\medskip   Inserts a medium space in vertical mode (between two paragraphs).

\bigskip   Inserts a big space in vertical mode (between two paragraphs).

The vertical mode is during the process of assembling boxes "vertically", like paragraphs to build a page. The horizontal mode is during the process of assembling boxes "horizontally", like letters to build a word or words to build a paragraph.

The fact they are vertical mode commands mean they will be ignored (or fail) in horizontal mode such as in the middle of a paragraph. The first token next the a double line-break is still in vertical mode if it does not expand to characters.

3.2.6 Rubber/Stretching lengths

The command:

generates a special rubber space where factor is a number, possibly a float. It stretches until all the remaining space on a line is filled up. If two \hspace{\stretch{factor}} commands are issued on the same line, they grow accordingly to the stretch factor.

The same way, you can stretch vertically:

You can also use \fill instead of \stretch{1}.

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.

There are 'shortcut commands' for stretching with factor 1 (*i.e.* with \stretch{1} or \fill): \hfill and \vfill.

Example:

Fill the rest of the line

Several macros allow filling the rest of the line -- or stretching parts of the line -- in different manners.

- \hfill will produce empty space.

- \dotfill will produce dots.

- \hrulefill will produce a rule.

3.2.7 Examples

Resize an image to take exactly half the text width :

Make distance between items larger (inside an itemize environment) :

Use of \savebox to resize an image to the height of the text:

3.2.8 References

[1] http://www.giss.nasa.gov/tools/latex/ltx-86.html

[2] http://anonscm.debian.org/cgit/debian-tex/texlive-bin. git/tree/texk/web2c/pdftexdir/pdftex.web?h=debian/ 2015.20150524.37493-5#n10460

3.2.9 See also

- University of Cambridge > Engineering Department > computing help > LaTeX > Squeezing Space in LaTeX

3.3 Counters

Counters are an essential part of LaTeX: they allow you to control the numbering mechanism of everything (sections, lists, captions, etc.). To that end each counter stores an integer value in the range of long integer, i.e., from -2^{31} to $2^{31} - 1$.

3.3.1 Counter manipulation

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

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:

To increase the counter, either use

or

or

here the number can also be negative. For automatic resetting you need to use \stepcounter.

To set the counter value explicitly, use

3.3.2 Counter access

There are several ways to get access to a counter.

- \theNameOfTheNewCounter will print the formatted string related to the counter (note the "the" before the actual name of the counter).

- \value{NameOfTheNewCounter} will return the counter value which can be used by other counters or for calculations. It is not a formatted string, so it cannot be used in text.

- \arabic{NameOfTheNewCounter} will print the formatted counter using arabic numbers.

Note that \arabic{NameOfTheNewCounter} may be used as a value too, but not the others.

Strangely enough, LaTeX counters are *not* introduced by a backslash in any case, even with the \the command. plainTeX equivalents \count and \newcounter\mycounter do abide by the backslash rule.

3.3.3 Counter style

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:

3.3.4   LaTeX default counters

- part

- chapter

- section

- subsection

- subsubsection

- paragraph

- subparagraph

- page

- figure

- table

- footnote

- mpfootnote

For the enumerate environment:

- enumi

- enumii

- enumiii

- enumiv

For the eqnarray environment:

- equation

3.3.5   Book with parts, sections, but no chapters

Here follows an example where we want to use parts and sections, but no chapters in the book class :

3.3.6   Custom *enumerate*

See the List Structures chapter.

3.3.7   Custom sectioning

Here is an example for recreating something similar to a section and subsection counter that already exist in La-TeX:

3.4   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. A general overview about different box commands can be found here: http://www.personal.ceu.hu/tex/spacebox.htm .

3.4.1   TeX character boxes

TeX characters are stored in boxes like every printed element. Boxes have three dimensional properties:

- The *height* is the length between the baseline and the top of the box.

- The *depth* is the length between the baseline and the bottom of the box.

- The *width* is the width of the box.
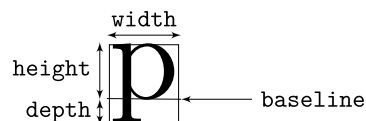
3.4.2   makebox and mbox

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. (See Hyphenation.) As you can put boxes inside boxes, these horizontal box packers give you ultimate flexibility.

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.

3.4.3 framebox

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:

You can tweak the following frame lengths.

- \fboxsep: the distance between the frame and the content.

- \fboxrule: the thickness of the rule.

This prints a thick and more distant frame:

This shows the box frame of a letter.

3.4.4 framed

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:

You can do it manually with a parbox.

3.4.5 raisebox

Now that we control the horizontal, the obvious next step is to go for the vertical. No problem for LaTeX. The

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. The two optional parameters set for the height and depth of the raisebox. For instance you can observe the difference when embedded in a framebox.

3.4.6 minipage and parbox

Most standard LaTeX boxes are not *long* commands, *i.e.* they do not support breaks nor paragraphs. However you can pack a paragraph of your choice into a box with either the \parbox[pos][height][contentpos]{width}{text} command or

the \begin{minipage}[pos][height][contentpos]{width} text \end{minipage} environment.

The pos parameter can take one of the letters **c**enter, **t**op or **b**ottom to control the vertical alignment of the box, relative to the baseline of the surrounding text. The height parameter is the height of the parbox or minipage. The contentpos parameter is the position of the content and can be one of **c**enter, **t**op, **b**ottom or **s**pread. 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.

This should print 3 boxes on the same line. Do not put another linebreak between the \fbox, otherwise you will put the following \fbox in another paragraph on another line.

Paragraphs in all boxes

You can make use of the *long* capabilities of minipage and parbox to embed paragraphs in non-long boxes. For instance:

This prevents the overfull badness.

You can also use

from the pbox package which will create a box of minimal size around the text. Note that the \pbox command takes an optional argument that specifies the vertical position of the text:

The valid values are b (bottom), t (top), and c (center). If you specify a length in the first (required) argument, the text will be wrapped:

3.4.7 savebox

A \savebox is a reference to a box filled with contents. You can use it as a way to print or manipulate something repeatedly.

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.

3.4.8 rotatebox

See Rotations.

3.4.9 colorbox and fcolorbox

See Colors. \fcolorbox can also be tweaked with \fboxsep and \fboxrule.

3.4.10   resizebox and scalebox

The graphicx package features additional boxes.

3.4.11   fancybox

the fancybox package provides additional boxes.

- \doublebox

- \ovalbox

- \shadowbox

3.5   Rules and Struts

3.5.1   Rules

The \rule command in normal use produces a simple
black box:

The parameter thickness determines the height, whereas
width determines the width of the produced rule. With
the optional parameter raise, you can optionally raise or
lower the produced rule above or below the baseline.

Here is an example (the thin lines are located at the base-
line):

This is useful for drawing vertical and horizontal lines.

3.5.2   Struts

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 min-
imal height. You could use it in a tabular environment
or in boxes to make sure a row has a certain minimum
height.

In LaTeX a strut is defined as

3.5.3   Stretched rules

LaTeX provides the \hrulefill command, which work like
a stretched horizontal space. See the Lengths chapter.

Chapter 4

Technical Texts

4.1 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 formulae, while 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. The mathtools package loads the amsmath package and hence there is no need to \usepackage{amsmath} in the preamble if mathtools is used.

4.1.1 Mathematics environments

LaTeX needs to know beforehand that the subsequent text does indeed contain mathematical elements. This is because LaTeX typesets maths notation differently from 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 inline, that is, within the body of text where it is declared, for example, I can say that $a + a = 2a$ within this sentence.

- *displayed* — displayed formulas are separate from the main text.

As math requires 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:

Suggestion: 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.

The equation* and displaymath environments are functionally equivalent.

If you are typing text normally, you are said to be in *text mode*, but 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 $\sum$, like in equations (This

only works with AMSMATH package). It is also possible to force this behaviour for all math environments by declaring \everymath{\displaystyle} at the very beginning (i.e. before \begin{document}), which is useful in longer documents.

4.1.2   Symbols

Mathematics has many symbols! One of the most difficult aspects of learning LaTeX is remembering how to produce symbols. There is 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. There are many examples such as Greek letters, set and relations symbols, arrows, binary operators, etc.

For example:

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.

4.1.3   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, kappa, 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:

Scroll down to #List of Mathematical Symbols for a complete list of Greek symbols.

4.1.4   Operators

An operator is a function that is written as a word: e.g. trigonometric functions (sin, cos, tan), logarithms and exponentials (log, exp), limits (lim), as well as trace and determinant (tr, det). LaTeX has many of these defined as commands:

For certain operators such as limits, the subscript is placed underneath the operator:

For the modular operator there are two commands: \bmod and \pmod:

To use operators that are not pre-defined, such as argmax, see custom operators

4.1.5   Powers and indices

Powers and indices are equivalent to superscripts and subscripts in normal text mode. The caret (^; also known as the circumflex accent) 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 }).

For powers with more than one digit, surround the power with {}.

An underscore (_) can be used with a vertical bar (|) to denote evaluation using subscript notation in mathematics:

4.1.6   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]:

You can embed fractions within fractions:

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.

For relatively simple fractions, especially within the text, it may be more aesthetically pleasing to use powers and indices:

If this looks a little "loose" (overspaced), a tightened version can be defined by inserting some negative space

If you use them throughout the document, usage of xfrac package is recommended. This package provides \sfrac command to create slanted fractions. Usage:

If fractions are used as an exponent curly braces have to be used around the \sfrac command:

$x^\frac{1}{2}$ % no error $x^\sfrac{1}{2}$ % error $x^{\sfrac{1}{2}}$ % no error

In some cases, using the package alone will result in errors about certain font shapes not being available. In that case, the lmodern and fix-cm packages need to be added as well.

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]:

Multiplication of two numbers

To make multiplication visually similar to a fraction, a nested array can be used, for example multiplication of numbers written one below the other.

4.1.7 Roots

The \sqrt command creates a square root surrounding an expression. It accepts an optional argument specified in square brackets ([and]) to change magnitude:

Some people prefer writing the square root "closing" it over its content. This method arguably makes it more clear 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:

This TeX code first renames the \sqrt command as \old-sqrt, then redefines \sqrt in terms of the old one, adding something more. The new square root can be seen in the picture on the left, compared to the old one on the right. 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.

An alternative piece of TeX code that does allow multiple roots is

However this requires the \usepackage{letltxmacro} package

4.1.8 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:

or

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.

There are many other "big" commands which operate in a similar manner:

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:

If you want the limits of an integral to be specified above and below the symbol (like the sum), use the \limits command:

However if you want this to apply to ALL integrals, it is preferable to specify the intlimits option when loading the amsmath package:

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].

4.1.9 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:

where \lbrack and \rbrack may be used in place of [and].

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, \right, and \middle commands. Any of the previous delimiters may be used in combination with these:

Curly braces are defined differently by using \left\{ and \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 (.).

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:

These commands are primarily useful when dealing with nested delimiters. For example, when typesetting

we notice that the \left and \right commands produce the same size delimiters as those nested within it. This can be difficult to read. To fix this, we write

Manual sizing can also be useful when an equation is too large, trails off the end of the page, and must be separated into two lines using an align command. Although the commands \left. and \right. can be used to balance the delimiters on each line, this may lead to wrong delimiter sizes. Furthermore, manual sizing can be used to avoid overly large delimiters if an \underbrace or a similar command appears between the delimiters.

Typesetting intervals

To denote open and half-open intervals, the notations]a,b[, (a,b),]a,b], (a,b], [a,b[and [a,b) are used. If the square bracket notation is used, then the interval must be put between curly braces ({ and }) in order to have correct spacing. Similarly, if a (half-)open interval starts with a negative number, then the number including its minus-symbol must also be put between curly brackets, so that LaTeX understands that the minus-symbol is the unary operation. Compare:

4.1.10   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 (\\)

To specify alignment of columns in the table, use starred version[5]:

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:

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:

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:

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:

If you need "border" or "indexes" on your matrix, plain TeX provides the macro \bordermatrix

Matrices in running text

To insert a small matrix, and not increase leading in the line containing it, use smallmatrix environment:

4.1.11   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:

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:

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 we may simply insert space characters into the \text commands.

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.

4.1.12 Formatting mathematics symbols

See also: w:Mathematical Alphanumeric Symbols, w:Help:Displaying a formula#Alphabets and typefaces and w:Wikipedia:LaTeX symbols#Fonts

We can now format text; what about formatting mathematical expressions? There are a set of formatting commands very similar to the font formatting ones just used, except that they are specifically aimed at text in math mode (requires amsfonts)

These 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 other math commands are unaffected.

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.

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:

4.1.13 Color

The package xcolor, described in Colors, allows us to add color to our equations. For example,

The only problem is that this disrupts the default LaTeX formatting around the - operator. To fix this, we enclose it in a \mathbin environment, since - is a binary operator. This process is described here.

4.1.14 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 on 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 is written as:

Similarly, there exists also a minus-plus sign:

4.1.15 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 *finetune* 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.

This code produces errors with Miktex 2.9 and does not yield the results seen on the right. Use \mathrm 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, \quads 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:

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 in this situation—what is needed are some small spaces to be utilized in this type of instance, and that's what LaTeX

provides:

NB you can use more than one command in a sequence to achieve a greater space if necessary.

So, to rectify the current problem:

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:

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.

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:

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.

4.1.16   Manually Specifying Formula Style

To manually display a fragment of a formula using text style, surround the fragment with curly braces and prefix the fragment with \textstyle. The braces are required because the \textstyle macro changes the state of the renderer, rendering all subsequent mathematics in text style. The braces limit this change of state to just the fragment enclosed within. For example, to use text style for just the summation symbol in a sum, one would enter

The same thing as a command would look like this:

Note the extra braces. Just one set around the expression won't be enough. That would cause all math after \tsum k to be displayed using text style.

To display part of a formula using display style, do the same thing, but use \displaystyle instead.

4.1.17   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:

Introducing 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 (ellipses) in your formulae. 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:

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.

Write an equation with the align environment

How to write an equation with the align environment with the amsmath package is described in Advanced Mathematics.

4.1.18   List of Mathematical Symbols

All the pre-defined mathematical symbols from the \TeX\ package are listed below. More symbols are available from extra packages.

Note: To use the Greek Letters in LaTeX that have the same appearance as their Roman equivalent, just use the Roman form: e.g., A instead of Alpha, B instead of Beta, etc.

If LaTeX does not include a command for the mathematical operator you want to use, for example \cis (cosine plus i times sine), add to your preamble:

\DeclareMathOperator\cis{cis}

You can then use \cis in the document just like \cos or any other mathematical operator.

4.1.19   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.

4.1.20   Notes

[1] http://www.ams.org/publications/authors/tex/amslatex

[2] http://www.ctan.org/tex-archive/macros/latex/contrib/mathtools/mathtools.pdf

[3] requires the amsmath package

[4] http://hdl.handle.net/2268/6219

[5] requires the mathtools package

4.1.21   Further reading

- meta:Help:Displaying a formula: Wikimedia uses a subset of LaTeX commands.

4.1.22   External links

- LaTeX maths symbols

- detexify: applet for looking up LaTeX symbols by drawing them

- amsmath documentation

- LaTeX - The Student Room

- The Comprehensive LaTeX Symbol List

- MathLex - LaTeX math translator and equation builder

4.2   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.

4.2.1   Equation numbering

The equation environment automatically numbers your equation:

You can also use the \label and \ref (or \eqref from the amsmath package) commands to label and reference equations, respectively. For equation number 1, \ref results in 1 and \eqref results in (1) :

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

to the preamble to get enumeration at the section level or

to have the enumeration go to the subsection level.

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 the preamble will result in the equation number in the above example being rendered as follows: (1.1..1).

To remove the duplicate dot, add the following command immediately after \numberwithin{equation}{section}:

For a numbering scheme using \numberwithin{equation}{subsection}, use:

in the preamble of the document.

Note: Although 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:

Referencing subordinate equations can be done using either of two methods: adding a label after the \begin{subequations} command, which will reference the main equation (1.1 above), or adding a label at the end of each line, before the \\ command, which will reference the sub-equation (1.1a or 1.1b above). It is possible to add both labels in case both types of references are needed.

4.2.2   Vertically aligning displayed mathematics

A problem often encountered 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:

or to show usage of L'Hôpital's rule:

It is convenient to define a new operator that will set the equals sign with H and the provided fraction:

which reduces the above example to:

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 (and can be aligned with the \vphantom command):

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]:

Alternatively, to use brackets instead of braces use \underbracket and \overbracket commands[2]:

The optional arguments set the rule thickness and bracket height respectively:

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 [and]) specifies the subscript, and the mandatory argument (using { and }) specifies the superscript (which can be left empty by inserting a blank space).

For more extensible arrows, you must use the mathtools package:

and for harpoons:

align and align*

The align and align* environments, available through the amsmath package, 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:

Note that the align environment must not be nested inside an equation (or similar) environment. Instead, align is a replacement for such environments; the contents inside an align are automatically placed in math mode.

align* suppresses numbering. To force numbering on a specific line, use the \tag{...} command before the line break.

align is similar, but automatically numbers each line like the equation environment. Individual lines may be referred to by placing a \label{...} before the line break. The \nonumber or \notag command can be used to suppress the number for a given line:

Notice that we've added some indenting on the second line. Also, we need to insert the double braces ({ }) before the + sign, 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.

More complicated alignments are possible, with additional &'s on a single line specifying multiple "equation columns", each of which is aligned. The following example illustrates the alignment rule of align*:

Braces spanning multiple lines

If you want a brace to continue across a new line, do the following:

In this construction, the sizes of the left and right braces are not automatically equal, in spite of the use of \left\{ and \right\}. This is because each line is typeset as a completely separate equation —notice the use of \right. and \left. so there are no unpaired \left and \right commands within a line (these aren't needed if the formula is on one line). You can control the size of the braces manually with the \big, \Big, \bigg, and \Bigg commands.

Alternatively, the height of the taller equation can be replicated in the other using the \vphantom command:

Using aligned braces for piecewise functions   You can also use \left\{ and \right. to typeset piecewise functions:

The cases environment

The cases environment[1] allows the writing of piecewise functions:

LaTeX will then take care of defining and or aligning the columns.

Within cases, text style math is used with results such as:

$$a = \begin{cases} \int x \, \mathrm{d}x \\ b^2 \end{cases}$$

Display style may be used instead, by using the dcases environment[2] from mathtools:

Often the second column consists mostly of normal text. To set it in the normal Roman font of the document, the dcases* environment may be used:[2]

Other environments

Although align and align* are the most useful, there are several other environments that may also be of interest:

There are also a few environments that don't form a math environment by themselves and can be used as building blocks for more elaborate structures:

For example:

4.2.3 Indented Equations

To indent an equation, you can set fleqn in the document class and then specify a certain value for the \mathindent variable:

4.2.4 Page breaks in math environments

To suggest that LaTeX insert a page break inside an ams-math environment, you may use the \displaybreak command before the line break. Just as with \pagebreak, \displaybreak can take an optional argument between 0 and 4 denoting the level of desirability of a page break. Whereas 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 too can have an optional argument denoting the priority 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.

Specific usage may look like this:

Page breaks before display maths (of all various forms) are controlled by \predisplaypenalty. Its default 10000 means never break immediately before a display. Knuth (*TeXbook* chapter 19) explains this as a printers' tradition not to have a displayed equation at the start of a page. It can be relaxed with

Sometimes an equation might look best kept together preceding text by a higher penalty, for example a single-line paragraph about a single-line equation, especially at the end of a section.

4.2.5 Boxed Equations

For a single equation or alignment building block, with the tag outside the box, use \boxed{}:

If you want the entire line or several equations to be boxed, use a minipage inside an \fbox{}:

There is also the mathtools \Aboxed{} which is able to box across alignment marks:

4.2.6 Custom operators

Although many common operators are available in La-TeX, sometimes you will need to write your own, e.g. to typeset the argmax operator. The \operatorname and \operatorname* commands[1] display custom operators; the * version sets the underscored option underneath like the \lim operator:

However, if the operator is frequently used, it is preferable to define a new operator that can be used throughout the entire document. The \DeclareMathOperator and \DeclareMathOperator* commands[1] are specified in the header of the document:

This defines a new command which may be referred to in the body:

4.2.7 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:

To override this behavior, use the \nolimits operator:

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 the \limits command.

Similarly one can put subscripts under a symbol that usually has them on the side:

Limits below and under:

To change the default placement of summation-type symbols to the side for every case, add the nosumlimits option to the amsmath package. To change the placement for integral symbols, add intlimits to the options. nonamelimits can be used to change the default for named operators like det, min, lim, etc.

To produce one-sided limits, use \underset as follows:

Subscripts and superscripts

You can place symbols in subscript or superscript (in summation style symbols) with \nolimits:

It's impossible to mix them with typical usage of such symbols:

To add both a prime and a limit to a symbol, one might use the \sideset command:

It is very flexible: for example, to put letters in each corner of the symbol use this command:

If you wish to place them on the corners of an arbitrary symbol, you should use \fourIdx from the fourIdx package.

But a simple grouping can also solve the problem:

since a math operator can be used with limits or no limits. If you want to change its state, simply group it. You can

make it another math operator if you want, and then you can have limits and then limits again.

Multiline subscripts

To produce multiline subscript, use the \substack command:

4.2.8   Text in aligned math display

To add small interjections in math environments, use the \intertext command:

Note that any usage of this command does not change the alignment.

Also, in the above example, the command \shortintertext{} from the mathtools package could have been used instead of intertext to reduce the amount of vertical white space added between the lines.

4.2.9   Changing font size

There may be a time when you would prefer to have some control over the font size. For example, using text-mode maths, by default a simple fraction will look like this: $\frac{a}{b}$, whereas you may prefer to have it displayed larger, like when in display mode, but still keeping it in-line, like this: $\frac{a}{b}$.

A simple approach is to utilize the predefined sizes for maths elements:

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 instead of the method provided below). The following code provides an example.

As you can see, as the fractions continue, they get smaller (although they will not get any smaller than in this example, where they have reached the \scriptstyle limit). If you want to keep the size consistent, you could declare each fraction to use the display style instead; e.g.

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.

But 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.

Note that 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.,

4.2.10   Forcing \displaystyle for all math in a document

Put

before

to force all math to

.

4.2.11   Adjusting vertical white space around displayed math

There are four parameters that control the vertical white space around displayed math:

Short skips are used if the preceding line ends, horizontally, before the formula. These parameters must be set after

.

4.2.12   Notes

[1] Requires amsmath package

[2] requires the mathtools package

4.3   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.

4.3.1   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.

4.3.2 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* is the name of 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.

The \newtheorem command may have at most one optional argument.

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.

4.3.3 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*. Use align* instead.

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}{}

4.3.4 Theorem styles

It adds the possibility to change the output of the environments defined by \newtheorem using the \theoremstyle command[1] 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:

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).

4.3.5   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}

4.3.6   Notes

[1] Requires the amsthm package

4.3.7   External links

- amsthm documentation

4.4   Chemical Graphics

chemfig is a package used to draw 2D chemical structures. It is an alternative to ochem. Whereas ochem requires Perl to draw chemical structures, chemfig uses the tikz package to produce its graphics. chemfig is used by adding the following to the preamble:

\usepackage{chemfig}

4.4.1   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. Absolute 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.

<bond type> describes the bond attaching <atom1> and <atom2>. There are 9 different bond types:

\chemfig{C(-[:0]H)(-[:90]H)(-[:180]H)(-[:270]H)}

<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:

Linear molecules (such as methane) are a weak example of this, but molecules are formed in chemfig by nesting.

4.4.2   Skeletal Diagrams

Skeleton diagrams can be produced as follows:

4.4.3   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).

4.4.4   Lewis Structures

Lewis   structures   use   the   syntax \lewis{<n1><n2>...<ni>,<atom>}, where <ni> is a number between 0 and 7 representing the position of the electrons. By default, the electrons are represented by a dash (-). Appending a period (.) or colon (:) after a number will display single and paired electrons respectively.

Lewis structures can also be included within \chemfig{}.

4.4.5   Ions

For example, consider an acetate ion:

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:

Alternatively, charges can be placed above ions using \chemabove{}{}:

4.4.6 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 $\left\{\chemfig{O-N(=[:60]O)-[:300]O}\right\} \Longleftrightarrow \left\{\chemfig{O=N(-[:60]O)-[:300]O}\right\} \Longleftrightarrow \left\{\chemfig{O-N(-[:60]O)=[:300]O}\right\}$

4.4.7 Chemical Reactions

Commands \chemrel and \chemsign were removed from chemfig package in latest versions, so in order to draw chemical reactions, one must instead use respectively \arrow and \+ commands in a block surrounded with \schemestart and \schemestop.

There are a few types of arrows that can be drawn with the \arrow command:

For more details on the \arrow command and chemical reactions in chemfig in general, consult the Part IV "Reaction schemes" of the chemfig documentation file.

Older versions

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 \chemrel{}:

A\chemrel{->}B\par   A\chemrel{<-}B\par A\chemrel{<->}B\par A\chemrel{<>}B

4.4.8 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.

\schemestart \chemname{\chemfig{R-C(-[:−30]OH)=[:30]O}}{Carboxylic acid} \+ \chemname{\chemfig{R'OH}}{Alcohol} \arrow{->} \chemname{\chemfig{R-C(-[:−30]OR')=[:30]O}}{Ester} \+ \chemname{\chemfig{H_2O}}{Water} \schemestop

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:

\schemestart \chemname{\chemfig{R'OH}}{Alcohol} \+ \chemname{\chemfig{R-C(-[:−30]OH)=[:30]O}}{Carboxylic acid} \arrow{->} \chemname{\chemfig{R-C(-[:−30]OR')=[:30]O}}{Ester} \+ \chemname{\chemfig{H_2O}}{Water} \schemestop

This is fixed by adding \chemnameinit{<deepest molecule>} before the first instance of \chemname{} in a reaction and by adding \chemnameinit{} after the reaction:

\schemestart \chemnameinit{\chemfig{R-C(-[:−30]OH)=[:30]O}} \chemname{\chemfig{R'OH}}{Alcohol} \+ \chemname{\chemfig{R-C(-[:−30]OH)=[:30]O}}{Carboxylic acid} \arrow{->} \chemname{\chemfig{R-C(-[:−30]OR')=[:30]O}}{Ester} \+ \chemname{\chemfig{H_2O}}{Water} \chemnameinit{} \schemestop

Lastly, adding \\ in <name> will produce a line-break, allowing the name to span multiple lines.

4.4.9 Advanced Graphics

For advanced commands and examples, refer to the chemfig manual, where a more thorough and complete introduction to the package can be found.

4.4.10   Chemical Formulae

If you need to typeset chemical formulae, you have the choice between two very god packages: mhchem and chemmacros.

Package mhchem

mhchem is a package used to typeset chemical formulae and equations. As well as typeset basic 2D chemical structures. To use this package, add the following to your preamble:

\usepackage[version=4]{mhchem}

Chemical species are included using the \ce command. For example

\ce{3H2O}   \\   \ce{1/2H2O}   \\   \ce{AgCl2-}   \\
\ce{H2_{(aq)}} \\

renders:

For more examples, see meta:Help:Displaying a formula#Chemistry 2.

A few things here are automatically typeset; The 2 in \ce{H2O} is automatically subscripted without requiring additional commands. The amount of the species precedes the formula. 1/2 and other fractional amounts are automatically typeset as in \ce{1/2H2O}. The charge in \ce{AgCl2-} is automatically superscripted. If the charge is neither 1 or −1, a ^ will superscript it, as in \ce{AgCl2-}. The phase is not automatically subscripted and needs to be enclosed in parenthesis preceded with a _ as in \ce{H2_{(aq)}}.

Since February 2016, the mhchem package is also available in TeX in MediaWiki sites like Wikipedia, using the tag <ce>...</ce>.

Package chemformula

chemformula is a package from a much bigger chemistry bundel used to typeset chemical formulae and equations. As well as typeset basic 2D chemical structures. The package uses a slightly different syntax compared to mhchem. Package chemmacros will be of great interest to people doing chemistry related work.

To use this package, add the following to your preamble:

\usepackage{chemformula}

Chemical species are included using the \ch command. For example

\ch{3 H2O} \\ \ch{1/2 H2O} \\ \ch{AgCl2-} \\
\ch{H2_{(aq)}} \\

renders:

As you can see, the syntax is almost the same.

4.4.11   XyMTeX package

The following code produces the image for corticosterone below.

\documentclass{letter}   \usepackage{epic,carom}   \pagestyle{empty}   \begin{document}   \begin{picture}(1000,500)
\put(0,0){\steroid[d]{3D==O;{{10}}==\lmoiety{H$_{3}$C};{{13}}==\l
\put(684,606){\sixunitv{}{2D==O;1==OH}{cdef}}
\end{picture} \end{document}

Corticosterone as rendered by XyMTeX

4.5   Algorithms

LaTeX has several packages for typesetting algorithms in form of "pseudocode". They provide stylistic enhancements over a uniform style (i.e., all in typewriter font) so that constructs such as loops or conditionals are visually separated from other text. The pseudocode is usually put in an *algorithm* environment. For typesetting *real* code, written in a *real* programming language, consider the *listings* package described in Source Code Listings.

4.5.1   Typesetting

There are four notable packages *algorithmic*, *algorithm2e*, *algorithmicx*, and *program*,

Typesetting using the algorithmic package

The algorithmic package uses a different set of commands than the algorithmicx package. This is not compatible

with revtex4-1. Basic commands are:

\STATE <text> \IF{<condition>} \STATE {<text>}
\ELSE \STATE{<text>} \ENDIF \IF{<condition>}
\STATE   {<text>}   \ELSIF{<condition>}
\STATE{<text>}   \ENDIF   \FOR{<condition>}
\STATE   {<text>}   \ENDFOR   \FOR{<condition>}
\TO <condition> } \STATE {<text>} \ENDFOR
\FORALL{<condition>} \STATE{<text>} \ENDFOR
\WHILE{<condition>} \STATE{<text>} \ENDWHILE
\REPEAT   \STATE{<text>}   \UNTIL{<condition>}
\LOOP \STATE{<text>} \ENDLOOP \REQUIRE
<text> \ENSURE <text> \RETURN <text> \PRINT
<text> \COMMENT{<text>} \AND, \OR, \XOR,
\NOT, \TO, \TRUE, \FALSE

Complete documentation is listed at . Most commands
are similar to the algorithmicx equivalents, but with dif-
ferent capitalization. The package algorithms bundle at
the ctan repository, dated 2009-08-24, describes both the
algorithmic environment (for typesetting algorithms) and
the algorithm floating wrapper (see below) which is de-
signed to wrap around the algorithmic environment.

The algorithmic package is suggested for IEEE journals
as it is a part of their default style sheet.[1]

How to rename require/ensure to input/output:

\floatname{algorithm}{Procedure}   \renewcom-
mand{\algorithmicrequire}{\textbf{Input:}}   \re-
newcommand{\algorithmicensure}{\textbf{Output:}}

Typesetting using the algorithm2e package

The algorithm2e package (first released 1995, latest up-
dated January 2013 according to the v5.0 manual) al-
lows typesetting algorithms with a lot of customization.
Like algorithmic, this package is also not compatible with
Revtex-4.1.[2]

Unlike algorithmic, algorithm2e provides a relatively
huge number of customization options to the algorithm
suiting to the needs of various users. The CTAN-manual
provides a comprehensible list of examples and full set of
controls.

Typically, the usage between \begin{algorithm} and
\end{algorithm} would be
1. Declaring a set of keywords(to typeset as func-
tions/operators), layout controls, caption, title, header
text (which appears before the algorithm's main steps e.g.:
Input,Output)
2. Writing the main steps of the algorithm, with each step
ending with a \;
This may be taken in analogy with writing a latex-
preamble before we start the actual document.

The package is loaded like

\usepackage[]{algorithm2e}

and a simple example, taken from the v4.01 manual, is

\begin{algorithm}[H]   \KwData{this   text}   \KwRe-
sult{how to write algorithm with \LaTeX2e } initial-
ization\; \While{not at end of this document}{ read
current\; \eIf{understand}{ go to next section\; current
section becomes this one\; }{ go back to the beginning of
current section\; } } \caption{How to write algorithms}
\end{algorithm}

which produces

Algorithm 1: How to write algorithms

More details are in the manual hosted on the ctan website.

Typesetting using the algorithmicx package

The algorithmicx package provides a number of
popular constructs for algorithm designs. Put \usep-
ackage{algpseudocode} in the preamble to use the
algorithmic environment to write algorithm pseu-
docode   (\begin{algorithmic}...\end{algorithmic}).
You might want to use the algorithm environ-
ment (\usepackage{algorithm}) to wrap your al-
gorithmic code in an algorithm environment (\be-
gin{algorithm}...\end{algorithm}) to produce a floating
environment with numbered algorithms.

The command \begin{algorithmic} can be given the op-
tional argument of a positive integer, which if given will
cause line numbering to occur at multiples of that inte-
ger. E.g. \begin{algorithmic}[5] will enter the algorith-
mic environment and number every fifth line.

Below is an example of typesetting a basic algorithm
using the algorithmicx package (remember to add the
\usepackage{algpseudocode} statement to your docu-
ment 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.

$$\textbf{if } i \geq maxval \textbf{ then}$$
$$\quad i \leftarrow 0$$
$$\textbf{else}$$
$$\quad \textbf{if } i + k \leq maxval \textbf{ then}$$
$$\quad\quad i \leftarrow i + k$$
$$\quad \textbf{end if}$$
$$\textbf{end if}$$

Basic commands have the following syntax:

Statement (\State causes a new line, can also be used in front of other commands)

\State $x\gets <value>$

Three forms of if-statements:

\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. Note that it is \ElsIf and not \ElseIf.

Loops:

\For{<condition>} <text> \EndFor
\ForAll{<condition>} <text> \EndFor
\While{<condition>} <text> \EndWhile
\Repeat <text> \Until{<condition>}
\Loop <text> \EndLoop

Pre- and postcondition:

\Require <text>
\Ensure <text>

Functions

\Function{<name>}{<params>} <body> \EndFunction
\Return <text>
\Call{<name>}{<params>}

This command will usually be used in conjunction with a \State command as follows:

\Function{Increment}{a} \State $a \gets a+1$ \State \Return a \EndFunction

Comments:

\Comment{<text>}

Note to users who switched from the old algorithmic package: comments may be placed everywhere in the source; there are no limitations as in the old algorithmic package.

The algorithmicx package allows you to define your own environments.

To define blocks beginning with a starting command and ending with an ending command, use

\algblock[<block>]{<start>}{<end>}

This defines two commands \<start> and \<end> which have no parameters. The text displayed by them is \textbf{<start>} and \textbf{<end>}.

With \algblockdefx you can give the text to be output by the starting and ending command and the number of parameters for these commands. In the text the n-th parameter is referenced by #n.

\algblockdefx[<block>]{<start>}{<end>} [<startparamcount>][<default value>]{<start text>} [<endparamcount>][<default value>]{<end text>}

Example:

\algblock[Name]{Start}{End} \algblockdefx[NAME]{START}{END}% [2][Unknown]{Start #1(#2)}% {Ending} \algblockdefx[NAME]{}{OTHEREND}% [1]{Until (#1)} \begin{algorithmic} \Start \Start \START[One]{x} \END \START{0} \OTHEREND{\texttt{True}} \End \Start \End \End \end{algorithmic}

More advanced customization and other constructions are described in the algorithmicx manual: http://mirror.ctan. org/macros/latex/contrib/algorithmicx/algorithmicx.pdf

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 |exptl(2,i); \\ |newline|() \OD % \rcomment{This text will be set flush to the right margin} \WHERE \PROC |exptl(x,n) \BODY z:=1; \DO \IF n=0 \THEN \EXIT \FI; \DO \IF |oddl(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
    where
    proc expt(x, n) ≡
        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
```

This text will be set flush to the right margin

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 \NumberPrograms\true and numbering turned off with \NumberPrograms\false

Package page

Package documentation

4.5.2 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.14)

\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}

The official manual is located at http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf

4.5.3 References

[1]

[2] http://tex.stackexchange.com/questions/70181/revtex4-1-and-algorithm2e-indentation-clash

- The official manual for the algorithms package, Rogério Brito (2009), http://mirrors.ctan.org/macros/latex/contrib/algorithms/algorithms.pdf

4.6 Source Code Listings

4.6.1 Using the *listings* package

Using the package listings you can add non-formatted text as you would do with \begin{verbatim} but its main aim is to include the source code of any programming language within your document. If you wish to include pseudocode or algorithms, you may find Algorithms and Pseudocode useful also.

To use the package, you need:

The listings package supports highlighting of all the most common languages and it is highly customizable. If you just want to write code within your document the package provides the lstlisting environment:

Another possibility, that is very useful if you created a program on several files and you are still editing it, is to import the code from the source itself. This way, if you modify the source, you just have to recompile the LaTeX code and your document will be updated. The command is:

in the example there is a Python source, but it doesn't matter: you can include any file but you have to write the full file name. It will be considered plain text and it will be highlighted according to your settings, that means it doesn't recognize the programming language by itself. You can specify the language while including the file with the following command:

You can also specify a scope for the file.

This comes in handy if you are sure that the file will not change (at least before the specified lines). You may also omit the firstline or lastline parameter: it means *everything up to or starting from this point*.

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}

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write(  Case  insensitive      );
Write(  Pascal  keywords.      );
```

Supported languages

It supports the following programming languages:

ABAP[2,4], ACSL, Ada[4], Algol[4], Ant, Assembler[2,4], Awk[4], bash, Basic[2,4], C#[5], C++[4], C[4], Caml[4], Clean, Cobol[4], Comal, csh, Delphi, Eiffel, Elan, erlang, Euphoria, Fortran[4], GCL, Gnuplot, Haskell, HTML, IDL[4], inform, Java[4], JVMIS, ksh, Lisp[4], Logo, Lua[2], make[4], Mathematica[1,4], Matlab, Mercury, MetaPost, Miranda, Mizar, ML, Modelica[3], Modula-2, MuPAD, NASTRAN, Oberon-2, Objective C[5] , OCL[4], Octave, Oz, Pascal[4], Perl, PHP, PL/I, Plasm, POV, Prolog, Promela, Python, R, Reduce, Rexx, RSL, Ruby, S[4], SAS, Scilab,

sh, SHELXL, Simula[4], SQL, tcl[4], TeX[4], VBScript, Verilog, VHDL[4], VRML[4], XML, XSLT.

For some of them, several dialects are supported. For more information, refer to the documentation that comes with the package, it should be within your distribution under the name listings-*.dvi.

Notes

1. It supports Mathematica code only if you are typing in plain text format. You can't include *.NB files \lstinputlisting{...} as you could with any other programming language, but Mathematica can export in a pretty-formatted LaTeX source.

2. Specification of the dialect is mandatory for these languages (e.g. language={[x86masm]Assembler}).

3. Modelica is supported via the dtsyntax package available here.

4. For these languages, multiple dialects are supported. C, for example, has ANSI, Handel, Objective and Sharp. See p. 12 of the listings manual for an overview.

5. Defined as a dialect of another language

Settings

You can modify several parameters that will affect how the code is shown. You can put the following code anywhere in the document (it doesn't matter whether before or after \begin{document}), change it according to your needs. The meaning is explained next to any line.

\usepackage{listings} \usepackage{color} \definecolor{mygreen}{rgb}{0,0.6,0} \definecolor{mygray}{rgb}{0.5,0.5,0.5} \definecolor{mymauve}{rgb}{0.58,0,0.82} \lstset{ % backgroundcolor=\color{white}, % choose the background color; you must add \usepackage{color} or \usepackage{xcolor}; should come as last argument basicstyle=\footnotesize, % the size of the fonts that are used for the code breakatwhitespace=false, % sets if automatic breaks should only happen at whitespace breaklines=true, % sets automatic line breaking captionpos=b, % sets the caption-position to bottom commentstyle=\color{mygreen}, % comment style deletekeywords={...}, % if you want to delete keywords from the given language escapeinside={\%*}{*)}, % if you want to add LaTeX within your code extendedchars=true, % lets you use non-ASCII characters; for 8-bits encodings only, does not work with UTF-8 frame=single, % adds a frame around the code keepspaces=true, % keeps spaces in text, useful for keeping indentation of code (possibly needs columns=flexible) keywordstyle=\color{blue}, %

keyword style language=Octave, % the language of the code morekeywords={*,...}, % if you want to add more keywords to the set numbers=left, % where to put the line-numbers; possible values are (none, left, right) numbersep=5pt, % how far the line-numbers are from the code numberstyle=\tiny\color{mygray}, % the style that is used for the line-numbers rulecolor=\color{black}, % if not set, the frame-color may be changed on line-breaks within not-black text (e.g. comments (green here)) showspaces=false, % show spaces everywhere adding particular underscores; it overrides 'showstringspaces' showstringspaces=false, % underline spaces within strings only showtabs=false, % show tabs within strings adding particular underscores stepnumber=2, % the step between two line-numbers. If it's 1, each line will be numbered stringstyle=\color{mymauve}, % string literal style tabsize=2, % sets default tabsize to 2 spaces title=\lstname % show the filename of files included with \lstinputlisting; also try caption instead of title }

escapeinside

The escapeinside line needs an explanation. The option escapeinside={A}{B} will define delimiters for escaping into LaTeX code, *i.e.* all the code between the string "A" and "B" will be parsed as LaTeX over the current *listings* style. In the example above, the comments for *Octave* start with %, and they are going to be printed in the document unless they start with %*, in which case they are read as LaTeX (with all LaTeX commands fulfilled) until they're closed with another *). If you add the above paragraph, the following can be used to alter the settings within the code:

There are many more options, check the official documentation.

Style definition

The package lets you define styles, *i.e.* profiles specifying a set of settings.

Example

\lstdefinestyle{customc}{ belowcaptionskip=1\baselineskip, breaklines=true, frame=L, xleftmargin=\parindent, language=C, showstringspaces=false, basicstyle=\footnotesize\ttfamily, keywordstyle=\bfseries\color{green!40!black}, commentstyle=\itshape\color{purple!40!black}, identifierstyle=\color{blue}, stringstyle=\color{orange}, } \lstdefinestyle{customasm}{ belowcaptionskip=1\baselineskip, frame=L, xleftmargin=\parindent, language=[x86masm]Assembler, basicstyle=\footnotesize\ttfamily, commentstyle=\itshape\color{purple!40!black}, } \lstset{escapechar=@,style=customc}

In our example, we only set two options globally: the default style and the escape character. Usage:

\begin{lstlisting} #include <stdio.h> #define N 10 /* Block * comment */ int main() { int i; // Line comment. puts("Hello world!"); for (i = 0; i < N; i++) { puts("LaTeX is also great for programmers!"); } return 0; } \end{lstlisting} \lstinputlisting[caption=Scheduler, style=customc]{hello.c}

The C part will print as

```c
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
```

Automating file inclusion

If you have a bunch of source files you want to include, you may find yourself doing the same thing over and over again. This is where macros show their real power.

\newcommand{\includecode}[2][c]{\lstinputlisting[caption=#2, escapechar=, style=custom#1]{#2}<!---->} % ... \includecode{sched.c} \includecode[asm]{sched.s} % ... \lstlistoflistings

In this example, we create one command to ease source code inclusion. We set the default style to be *customc*. All listings will have their name as caption: we do not have to write the file name twice thanks to the macro. Finally we list all listings with this command from the listings package.

See Macros for more details.

Encoding issue

By default, listings does not support multi-byte encoding for source code. The extendedchar option only works for 8-bits encodings such as latin1.

To handle UTF-8, you should tell listings how to interpret the special characters by defining them like so

\lstset{literate= {á}{{\'a}}1 {é}{{\'e}}1 {í}{{\'i}}1 {ó}{{\'o}}1 {ú}{{\'u}}1 {Á}{{\'A}}1 {É}{{\'E}}1 {Í}{{\'I}}1 {Ó}{{\'O}}1 {Ú}{{\'U}}1 {à}{{\`a}}1 {è}{{\`e}}1 {ì}{{\`i}}1 {ò}{{\`o}}1 {ù}{{\`u}}1 {À}{{\`A}}1 {È}{{\`E}}1 {Ì}{{\`I}}1 {Ò}{{\`O}}1 {Ù}{{\`U}}1 {ä}{{\"a}}1 {ë}{{\"e}}1 {ï}{{\"i}}1 {ö}{{\"o}}1 {ü}{{\"u}}1 {Ä}{{\"A}}1 {Ë}{{\"E}}1 {Ï}{{\"I}}1 {Ö}{{\"O}}1 {Ü}{{\"U}}1 {â}{{\^a}}1 {ê}{{\^e}}1 {î}{{\^i}}1 {ô}{{\^o}}1 {û}{{\^u}}1 {Â}{{\^A}}1 {Ê}{{\^E}}1 {Î}{{\^I}}1 {Ô}{{\^O}}1 {Û}{{\^U}}1 {œ}{{\oe}}1 {Œ}{{\OE}}1 {æ}{{\ae}}1 {Æ}{{\AE}}1 {ß}{{\ss}}1 {ű}{{\H{u}}}1 {Ű}{{\H{U}}}1 {ő}{{\H{o}}}1 {Ő}{{\H{O}}}1 {ç}{{\c c}}1 {Ç}{{\c C}}1 {ø}{{\o}}1 {å}{{\r a}}1 {Å}{{\r A}}1 {€}{{\euro}}1 {£}{{\pounds}}1 }

The above table will cover most characters in latin languages. For a more detailed explanation of the usage of the literate option check section 6.4 in the Listings Documentation.

Another possibility is to replace \usepackage{listings} (in the preamble) with \usepackage{listingsutf8}.

Customizing captions

You can have fancy captions (or titles) for your listings using the caption package. Here is an example for listings.

\usepackage{caption} \usepackage{listings} \DeclareCaptionFont{white}{ \color{white} } \DeclareCaptionFormat{listing}{ \colorbox[cmyk]{0.43, 0.35, 0.35,0.01} }{ \parbox{\textwidth}{\hspace{15pt}#1#2#3} } } \captionsetup[lstlisting]{ format=listing, labelfont=white, textfont=white, singlelinecheck=false, margin=0pt, font={bf,footnotesize} } % ... \lstinputlisting[caption=My caption]{sourcefile.lang}

4.6.2 The *minted* package

minted is an alternative to listings which has become popular. It uses the external Python library Pygments for code highlighting, which as of Nov 2014 boasts over 300 supported languages and text formats.

As the package relies on external Python code, the setup require a few more steps than a usual LaTeX package, so please have a look at their GitHub repo and their manual.

4.6.3 References

A lot more detailed information can be found in a PDF by Carsten Heinz and Brooks Moses.

Details and documentation about the Listings package can be found at its CTAN website.

4.7 Linguistics

There are a number of LaTeX packages available for writing linguistics papers. Various packages have been created for enumerated examples, syntactic trees, OT tableaux, feature matrices, IPA fonts, and many other applications. Some packages such as the tipa package are effectively standard within the field, while others will vary by author preference.

Some recommended packages:[1]

- Glosses: gb4e or Covington;

- 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;

  - The xy package itself has a steep learning curve, but allows a lot of control; for simplest trees use the xymatrix feature and arrows;

  - tikz-qtree has the same syntax as qtree, but uses PGF/TikZ, which allows more options for drawing arrows, etc.

- Dependency trees and bubble parses:

  - The TikZ-dependency package provides a high-level, convenient interface to draw dependency graphs. It is based on PGF/TikZ but does not require prior knowledge of TikZ in order to be used productively.

- Attribute-Value Matrices (AVMs): avm

- John Frampton's expex: expex

4.7.1 Enumerated examples

There are several commonly used packages for creating the kinds of numbered examples that are used in linguistics publications.

gb4e

The gb4e package[2] is called with:

\usepackage{gb4e}

IMPORTANT: If you use gb4e package, let it be **the last \usepackage call** in the document's preamble. Otherwise you may get exceeded parameter stack size error.

Examples for this package are placed within the exe environment, and each example is introduced with the \ex command.

\begin{exe} \ex This is an example. \end{exe}

produces:

(1) This is an example.

Multiple examples can be included within the environment, and each will have its own number.

\begin{exe} \ex This is the first example. \ex This is the second example. \ex This is the third. \end{exe}

produces:

(1) This is the first example.

(2) This is the second example.

(3) This is the third.

To create nested lists of examples, the xlist enviroment is used.

\begin{exe} \ex \begin{xlist} \ex This is a sub-example. \ex This is a second sub-example. \ex \begin{xlist} \ex This is a sub-sub-example. \ex This is a second sub-sub-example. \end{xlist} \end{xlist} \end{exe}

produces:

(1) a. This is a sub-example.
     b. This is a second sub-example.
     c. i. This is a sub-sub-example.
        ii. This is a second sub-sub-example.

For notating acceptability judgments, the \ex command can take an optional argument. When including a judgment marker, the corresponding sentence must be surrounded by braces.

\begin{exe} \ex This sentence is grammatical English. \ex[*] {This sentence English in ungrammatical is.} \end{exe}

produces:

(1) This sentence is grammatical English.

(2)   * This sentence English in ungrammatical i

Referencing examples in text works as it does in normal LaTeX documents. See the labeling and cross-referencing section for more details.

\begin{exe} \ex\label{ex1} Godzilla destroyed the city. \ex\label{ex2} Godzilla roared.   \end{exe} Sentence (\ref{ex1}) contains two arguments, but (\ref{ex2}) contains only one.

Further details can be found in the full documentation available here.

lingmacros

The lingmacros package[3] created by Emma Pease is an alternate method for example numbering. This package uses two main commands, \enumsentence and \eenumsentence. The former is used for singleton examples, while the latter command is used for nested examples.

\enumsentence{This is an example.}

(1)   This is an example.

\enumsentence{This is the first example.} \enumsentence{This is the second example.} \enumsentence{This is the third.}

(1)   This is the first example.

(2)   This is the second example.

(3)   This is the third.

Multiply nested examples make use of the normal LaTeX list environments.

\eenumsentence{\item This is a sub-example.   \item This is a second sub-example. \item \begin{enumerate} \item This is sub-sub-example. \item This is a second sub-sub-example. \end{enumerate} }

produces:

(1)   a. This is a sub-example.
      b. This is a second sub-example.
      c.   i. This is a sub-sub-example.
           ii. This is a second sub-sub-example.

Full documentation can be found here.

4.7.2   Syntactic trees

Often, linguists will have to illustrate the syntactic structure of a sentence. One device for doing this is through syntactic trees. Unfortunately, trees look very different in different grammar formalisms, and different LaTeX packages are suited for different formalisms.

Constituent trees

While there are several packages for drawing syntactic trees available for LaTeX, this article focuses on the qtree and xyling packages.

qtree   Drawing trees with qtree is relatively straightforward. First, the qtree package has to be included in the document's preamble:

\usepackage{qtree}

A new tree is started using the **\Tree** command, each (sub-)tree is indicated by brackets **[ ]**. The root of a (sub-)tree is always preceded by a **.**, leaf nodes are simply expressed by their labels.

For example, the following code

\Tree [.S [.NP LaTeX] [.VP [.V is] [.NP fun]]]

produces this syntactic tree as output:
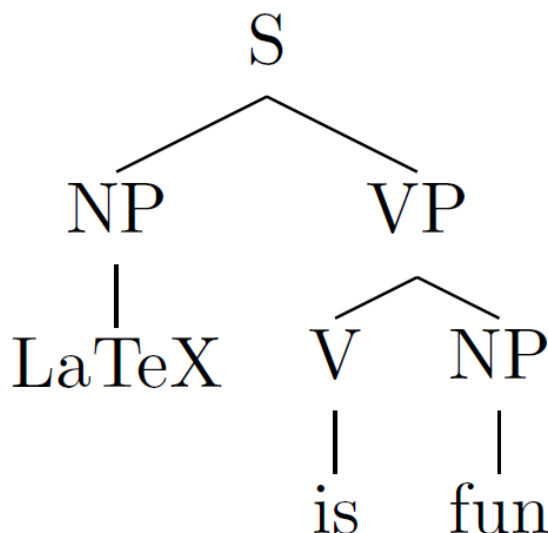
Note that the spaces before the closing brackets are **mandatory**.

By default, qtree centers syntactic trees on the page. This behaviour can be turned off by either specifying the behaviour when loading the package

\usepackage[nocenter]{qtree} % do not center trees

or via the command

\qtreecenterfalse % do not center trees from here on

anywhere in the document. The effect of the latter can be undone by using the command

\qtreecentertrue % center trees from here on

IMPORTANT: If you use gb4e package, let it be the last **\usepackage** call in the document's preamble. Otherwise you may get exceeded parameter stack size error.

tikz-qtree   Using the same syntax as qtree, tikz-qtree is another easy-to-use alternative for drawing syntactic trees.

For simple trees, tikz-qtree is completely interchangable with qtree. However, some of qtree's advanced features are implemented in a different way, or not at all. On the other hand, tikz-qtree provides other features such as controlling the direction of the tree's growth (top to bottom, left to right etc.) or different styles for edges.

To use the tikz-qtree package for drawing trees, put the following into the document's preamble:

\usepackage{tikz} \usepackage{tikz-qtree}

The syntax of tikz-qtree and result when drawing a simple tree is the same as for qtree.

\Tree [.S [.NP LaTeX] [.VP [.V is] [.NP fun]]]
 Note that, other than for qtree, trees are not centered by
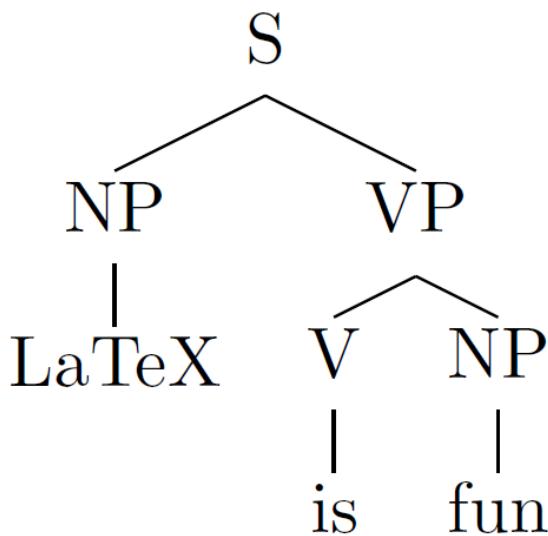
default. To center them, put them into a centered environment:

\begin{center} \Tree [.S [.NP LaTeX] [.VP [.V is] [.NP fun]]] \end{center}

For setting the style of trees, tikz-qtree provides the **\tikzset** command. For example, to make a tree grow from left to right instead of from top to bottom, use the following code:

\tikzset{grow'=right} % make trees grow from left to right \tikzset{every tree node/.style={anchor=base west}} % allign nodes of the tree to the left (west) \Tree [.S [.NP LaTeX] [.VP [.V is] [.NP fun]]]
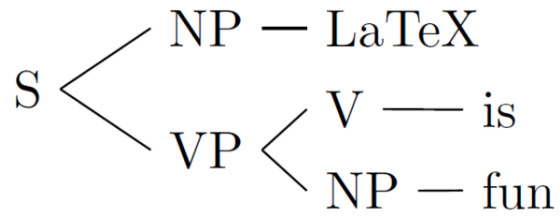
 The above code changes the default orientation for **all** trees that are defined after **\tikzset** commands. To only change the direction of a single tree, it has to be put into a **\tikzpicture** environment:

\begin{tikzpicture} % all changes only affect trees within this environment \tikzset{grow'=right} % make trees grow from left to right \tikzset{every tree node/.style={anchor=base west}} % allign nodes of the tree to the left (west) \Tree [.S [.NP LaTeX] [.VP [.V is] [.NP fun]]] \end{tikzpicture}

Dependency Trees

Dependency trees can take multiple visual forms. Commonly, they quite resemble phrase structure trees. Alternatively, they can be captured by brackets drawn above running text.

Two-dimensional Dependency Trees   These can be either achieved using the fairly universal drawing package TikZ, like so:

% In the preamble: \usepackage{tikz} % In the document: \begin{tikzpicture} \node (is-root) {is} [sibling distance=3cm] child { node {this} } child { node {tree} [sibling distance=1.5cm] child { node {an} } child { node {example} } child { node {.} } child[missing] }; \path (is-root) +(0,−2.5\tikzleveldistance) node {\textit{This is an example tree.}}; \end{tikzpicture}

which gives you the following drawing:

TikZ has the advantage that it allows for generating PDF directly from the LaTeX source, without need for any detour of compiling to DVI using latex, and then converting to PDF probably via PS using tools such as dvips and ps2pdf. Latter is the case of another package based on the package **xy**, namely **xyling**.

The code for a similar tree using **xyling** might look like:

% In the preamble: \usepackage{xyling} % In the document: \Tree{ & \K{is}\B{dl}\B{drr} \\ \K{this} &&& \K{tree}\B{dll}\B{dl}\B{dr} \\ & \K{an} & \K{example} && \K{.} } \medskip \textit{This is an example tree.}

which gives you a drawing like this:
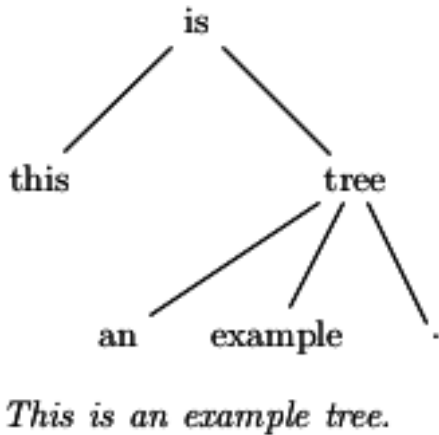
A dependency tree created using TikZ
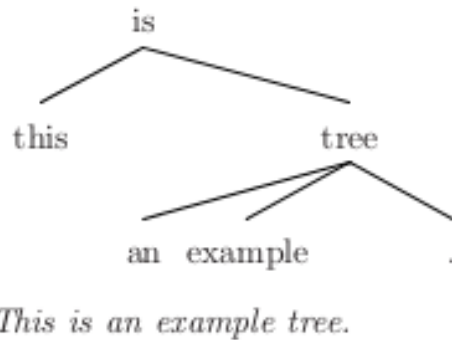
A dependency tree created using xyling

Dependency Trees as Brackets above Text   One way to typeset dependency brackets above running text is using the package **xytree**. It gives you fairly good control of how the brackets are typeset but requires compiling the LaTeX code to DVI (and perhaps converting to PDF using the tools dvips and ps2pdf later).

An example code:

```
% In the preamble: \usepackage{xytree} % In the
document: \xytext{ \xybarnode{Peter} &~~~& \xy-
barnode{and}      \xybarconnect(UL,U){−2}"_{\small
conj}"    \xybarconnect(UR,U){2}"^{\small    conj}"
&~~~&    \xybarnode{Mary}    &~~~&    \xybarn-
ode{bought}     \xybarconnect[8](UL,U){−4}"_{\small
subj}"      \xybarconnect[13]{6}"^{\small     punct}"
\xybarconnect[8](UR,U){4}"^{\small          obj}"
&~~~&   \xybarnode{a}   &~~~&   \xybarnode{car}
\xybarconnect(UL,U){−2}"_{\small    det}"    &~~~&
\xybarnode{.} }
```

results in:
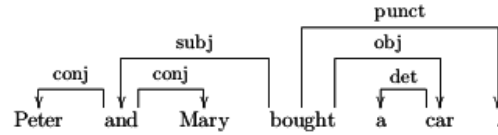
A dependency tree above running text created using xytree

Dependency Trees using TikZ-dependency   The package provides high level commands to design and style dependency graphs. To draw a graph, you only need to create a dependency environment, write the text of the sentence within the deptext environment and use depedge commands to draw the edges. Global and local optional parameters can be used to style and fine tune the looks of the graph, as shown in the following example:

```
% In the preamble:  \usepackage{tikz-dependency}
% In the document:    \begin{dependency}[theme
=    simple]    \begin{deptext}[column    sep=1em]
A \& hearing \& is \& scheduled \& on \&
the \& issue \& today \& .   \\ \end{deptext}
\deproot{3}{ROOT}              \depedge{2}{1}{ATT}
\depedge[edge start x offset=−6pt]{2}{5}{ATT}
\depedge{3}{2}{SBJ}              \depedge{3}{9}{PU}
\depedge{3}{4}{VC}            \depedge{4}{8}{TMP}
\depedge{5}{7}{PC}                \depedge[arc    an-
gle=50]{7}{6}{ATT} \end{dependency}
```

This code snippet would produce the following result:
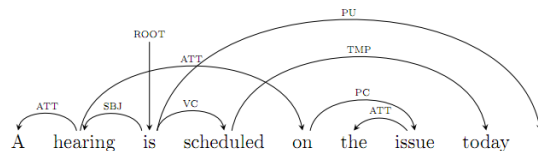
A dependency tree drawn with TikZ-dependency.

4.7.3   Glosses

Below, it is explained how to make glossed examples with different packages.

With gb4e

To create a glossed example, use the normal exe environment. But after the \ex tag, introduce the example and its gloss using \gll and the translation after it with \trans tag.

```
\begin{exe} \ex \gll Кот ест сметану\\ cat.NOM
eat.3.SG.PRS sour-cream.ACC\\ \trans `The cat eats
sour cream' \end{exe}
```

The code will produce the following output:

(1) Кот   ест   сметану

Cat.NOM eat.3.SG.PRS sour-cream.ACC

'The cat eats sour cream'

Vertically aligned glosses are separated by spaces, so if it's necessary to include a space in part the gloss, simply enclose the connected parts inside braces.

\begin{exe} \ex \gll Pekka pel\"astyi karhusta.\\ Pekka {became afraid} bear.ELA\\ \trans `Pekka became afraid because of the/a bear.' \end{exe}

With lingmacros

The lingmacros package uses the \shortex command to introduce glossed examples inside the \enumsentence and \eenumsentence commands. This command takes four arguments and builds off the normal tabular environment. Its first argument specifies the number of columns in the gloss. The second and third arguments give the text and its gloss respectively, and items within each column are divided by the usual & tabular separator. The fourth argument is the translation.

\enumsentence{\shortex{3} {Pekka & pel\"astyi & karhu-sta.} {Pekka & became afraid & bear.ELA} {`Pekka became afraid because of the/a bear.'} }

4.7.4   IPA characters

The tipa package is the standard LaTeX package for International Phonetic Alphabet symbols.

There are two methods for getting IPA symbols into a document. The first way is to use the IPA environment.

This method is useful for long stretches of text that need to be in IPA. Alternatively, there is the \textipa command that will format the text in its argument into IPA. This command is similar to other font typesetting commands.

Basic symbols

The IPA format works by translating ASCII characters into corresponding IPA symbols. Lower case letters are rendered as usual,

however capital letters are rendered differently.

Punctuation marks that are normally used in LaTeX are also rendered faithfully in the IPA environment.

Numerals and @ also have variants in the IPA environment.

In addition, there are a number of special macros for

representing symbols that don't have other associations, some of which are listed here. For a complete list see the official TIPA Manual[4].

The \; macro preceding a capital letter produces a small caps version of the letter.

The \: macro produces retroflex symbols.

The \! macro produces implosive symbols and the bilabial click.

XeLaTeX

Another way of inputting IPA symbols in a document is to use XeTeX as the compiler and insert the symbols directly, using a character map or an IPA keyboard.[5]

4.7.5   Phonological rules

Typesetting phonological rules can be done with the help of the phonrule package.[6]

Following is an example of what you can achieve with the package:

4.7.6   References

[1]   LaTeX for Linguists presentation

[2]   The gb4e package on CTAN

[3]   The lingmacros package on CTAN

[4]   TIPA manual

[5]   For more information on IPA keyboards, see SIL .

[6]   The package on CTAN.

4.7.7   External links

- LaTeX for Linguists
- The qtree package for drawing syntactic trees.
- The gb4e package page on CTAN.

Chapter 5

Special Pages

5.1 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.

5.1.1 Using makeidx

To enable the indexing feature of LaTeX, the makeidx package must be loaded in the preamble with:

and the special indexing commands must be enabled by putting the

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

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

can be re-written as

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

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 right 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:

Subentries   If some entry has subsections, these can be marked off with !. For example,

would create an index entry 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

so that the entry in the index will show as ' $\vec{F}$ ' but be sorted as 'F'.

To combine with the above feature for subentries, you should style the appropriate component(s):

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

and on page 10 of the same book you wish to show the main section on bulldogs with a bold page number, use

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

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 |, which are otherwise escape characters, in the index, one must quote these characters in the \index command by putting a double quotation mark (") in front of them, and one can only place a " in the index by quoting it (i.e., a key for " would be \index{""}).

This rule does not hold for \", so to put the letter ä in the index, one may still use \index{a@\"{a}}.

5.1.2   Abbreviation list

You can make a list of abbreviations with the package nomencl . You may also be interested in using the glos-

saries package described in the Glossary chapter. Another option is the package acronym .

To enable the Nomenclature feature of LaTeX, the nomencl package must be loaded in the preamble with:

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.

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:

5.1.3   Multiple indices

If you need multiple indices you can use the package multind .

This package provides the same commands as makeidx, but now you also have to pass a name as the first argument to every command.

5.1.4   Adding index to table of contents

By default, Index won't show in Table Of Contents, so you have to add it manually.

To add index as a chapter, use these commands:

If you use the book class, you may want to start it on an odd page by using \cleardoublepage.

5.1.5   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 mod-

ifiers, 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 is a much more extensible and robust indexing system than the makeindex system.

For example, one does not need to write:

to get the Lin entry after LAN and before LZA, instead, it's enough to write

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 ISO-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 (for example by adding it to QuickBuild).

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

5.2   Glossary

Many technical documents use terms or acronyms unknown to the general population. It is common practice to add a glossary to make such documents more accessible.

The glossaries package can be used to create glossaries. It supports multiple glossaries, acronyms, and symbols. This package replaces the glossary package and can be used instead of the nomencl package.[1] Users requiring a simpler solution should consider hand-coding their entries by using the description environment, or the longtabu environment provided by the tabu package.

5.2.1   Jump start

Place \usepackage{glossaries} and \makeglossaries in your preamble (after \usepackage{hyperref} if present). Then define any number of \newglossaryentry and \newacronym glossary and acronym entries in your preamble (recommended) or before first use in your document proper. Finally add a \printglossaries call to locate the glossaries list within your document structure. Then pepper your writing with \gls{mylabel} macros (and similar) to simultaneously insert your predefined text and build the associated glossary. File processing must now include a call to makeglossaries followed by at least one further invocation of latex or pdflatex.

5.2.2   Using glossaries

To use the glossaries package, you have to load it explicitly:

if you wish to use xindy (recommended) for the indexing phase, as opposed to makeindex (the default), you need to specify the xindy option:

For the glossary to show up in your Table of Contents, you need to specify the toc option:

See also Custom Name at the bottom of this page.

Finally, place the following command in your document preamble in order to generate the glossary:

Any links in resulting glossary will not be "clickable" unless you load the glossaries package *after* the hyperref package.

In addition, users who wish to make use of makeglossaries will need to have Perl installed — this is not normally present by default on Microsoft Windows platforms. That said, makeglossaries simply provides a convenient interface to makeindex and xindy and is not essential.

5.2.3   Defining glossary entries

To use an entry from a 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.

5.2.4   Defining terms

To define a term in glossary you use the \newglossaryentry macro:

<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:

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:

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 omitting 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:

You can also specify plural forms, if they are not formed by adding "s" (we will learn how to use them in next section):

Or, for acronyms:

This will avoid the wrong long plural: Frame per Seconds.

So far, the glossary entries have been defined as key-value lists. Sometimes, a description is more complex than just a paragraph. For example, you may want to have multiple paragraphs, itemized lists, figures, tables, etc. For such glossary entries use the command longnewglossaryentry in which the description follows the key-value list. The computer entry then looks like this:

Defining symbols

Defined entries can also be symbols:

You can also define both a name and a symbol:

Note that not all glossary styles show defined symbols.

Defining acronyms

To define a new acronym you use the \newacronym macro:

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:

Defined acronyms can be put in separate list if you use acronym package option:

5.2.5   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:

Description of commands used in above example:

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.

This command prints the plural of the defined term, other than that it behaves in the same way as gls.

This command prints the singular form of the term with the first character converted to upper case.

This command prints the plural form with first letter of the term converted to upper case.

This command creates the link as usual, but typesets the *alternate text* instead. It can also take several options which changes its default behavior (see the documentation).

This command prints what ever is defined in \newglossaryentry{<label>}{symbol={Output of glssymbol}, ...}

This command prints what ever is defined in \newglossaryentry{<label>}{description={Output of glsdesc}, ...}

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 the command:

or, if you want to reset the use status of all acronyms:

If you just want to print the long version of an acronym without the abbreviation "<full>", use :

If you just want to print the long version of an acronym with the abbreviation "<full> (<abbrv>)", use :

If you just want to print the abbreviation "<abbrv>", use :

5.3   Displaying the Glossary

To display the sorted list of terms you need to add:

at the place you want the glossary and the list of acronyms to appear.

If all entries are to be printed the command

can be inserted before \printglossaries. You may also want to use \usepackage[nonumberlist]{glossaries} to suppress the location list within the glossary.

Separate Glossary and List of Acronyms

\printglossaries will display all the glossaries in the order in which they were defined.[2] If no custom glossaries are defined, the default glossary and the list of acronyms will be displayed.

The glossary and the list of acronyms can be displayed separately in different places[3]:

Dual entries with reference to a glossary entry from an acronym   It may be useful to have both an acronym and a glossary entry for the same term. To link these two, define the acronym with a reference to the glossary entry like this:

Refer to acronym with \gls{OWD} and the glossary with \gls{gls-OWD}

To make this easier, we can use this command (modified from example in the official docs):

Syntax: \newdualentry[glossary options][acronym options]{label}{abbrv}{long}{description}

then, define new (dual) entries for glossary and acronym list like this:

Custom Name

The name of the glossary section can be replaced with a custom name or translated to a different language. Add the option title to \printglossary to specify the glossary's title. Add the option toctitle to specify a the title used in the table of content (if not used, title is used as default). [4]

Remove the point

To omit the dot at the end of each description, use this code:

Changing Glossary Entry Presentation Using Glossary Styles

A number of pre-built styles are available, and can be changed easily using

Commonly used styles include list

My Term Has some long description 7, 9

altlist (inserts newline after term and indents description)

My Term Has some long description 7, 9

altlistgroup or listgroup (group adds grouping based on the first letters of the terms)

M **My First Term** Has some long description 7, 9 **My Second Term** Has some long description 7, 9

altlisthypergroup or listhypergroup (hyper adds an hyperlinked 'index' at the top of each glossary to jump to a group)

A|B|C|D|F|G|I|M|O|R|S|C|D|G|M|P A **A First term** Has some long description 7, 9 B **Barely missed first** Has some long description 7, 9

5.3.1   Building your document

Building your document and its glossary requires three steps:

1. build your LaTeX document — this will also generate the files needed by makeglossaries

2. invoke makeglossaries — a script which selects the correct character encodings and language settings and which will also run xindy or makeindex if these are specified in your document file

3. build your LaTeX document again — to produce a document with glossary entries

Thus:

latex doc makeglossaries doc latex doc

where latex is your usual build call (perhaps pdflatex) and doc is the name of your LaTeX master file.

If your entries are interlinked (entries themselves link to other entries with \gls calls), you will need to run steps 1 and 2 twice, that is, in the following order: 1, 2, 1, 2, 3.

If you encounter problems, view the doc.log and doc.glg files in a text editor for clues.

5.4 Example for use in windows with Texmaker

5.4.1 Compile glossary with xindy - In Windows with Texmaker

In TeX Live and since June 2015 in MikTeX **xindy** is already included.

There is only one issue with path of the install directory of MikTeX containing spaces. It can be solved via the following edit: http://tex.stackexchange.com/questions/251221/miktex-and-xindy-problems/251801#251801

You need to restart Texmaker after installation of xindy, to update PATH references to xindy and Perl binaries.

Then, in Texmaker, go to **User -> User Commands -> Edit User Commands**.
Choose command 1

1. Menuitem = **makeglossaries**

2. Command = **makeglossaries %**

Now push **Alt+Shift+F1**and then ->**F1**

Note, for use with the "use build directory" option of Texmaker: makeglossaries needs to find the aux file. Thankfully, while Texmaker does not help there, the option **-d <dir>** of makeglossaries provides for the subdirectory case. Hence the Command in this case should be: Command = **makeglossaries -d build %** instead.

5.4.2 Document preamble

In preamble should be included (note, **hyperref** should be loaded **before** the **glossaries**):

\usepackage[nomain,acronym,xindy,toc]{glossaries} % nomain, if you define glossaries in a file, and you use \include{INP-00-glossary} \makeglossaries \usepackage[xindy]{imakeidx} \makeindex

5.4.3 Glossary definitions

Write all your glossaries/acronyms in a file: Ex: **INP-00-glossary.tex**

\newacronym{ddye}{D$_{\text{dye}}$}{donor dye, ex. Alexa 488} \newacronym[description={\glslink{r0}{F\"{o}rster distance}}]{R0}{R_{0}}{F\"{o}rster distance} \newglossaryentry{r0}{name=\glslink{R0}{\ensuremath{R_{0}}},text=F\"{o}rster distance,description={F\"{o}rster distance, where 50\% ...}, sort=R} \newglossaryentry{kdeac}{name=\glslink{R0}{\ensuremath{k_{DEAC}}},text=$k_{DE description={is the rate of deactivation from ... and emission)}, sort=k}

5.4.4 Include glossary definitions and print glossary

Include glossary definitions in the preamble (Before "\begin{document}")

\loadglsentries[main]{INP-00-glossary} % or using \input: %**\input{INP-00-glossary}** \begin{document}

Print glossaries, near end

\appendix \bibliographystyle{plainnat} \bibliography{bibtex} \printindex **\printglossaries** \end{document}

5.4.5 References

[1] http://www.ctan.org/pkg/nomencl

[2] http://mirror.ox.ac.uk/sites/ctan.org/macros/latex/contrib/glossaries/glossaries-user.html#dx1-35001

[3] http://mirror.ox.ac.uk/sites/ctan.org/macros/latex/contrib/glossaries/glossaries-user.html#dx1-43001

[4] User Manual for glossaries.sty v4.02 as of 2014.01.13 http://mirror.ox.ac.uk/sites/ctan.org/macros/latex/contrib/glossaries/glossaries-user.html#sec:printglossary

- The glossaries documentation, http://tug.ctan.org/tex-archive/macros/latex/contrib/glossaries/

- *Using LaTeX to Write a PhD Thesis*, Nicola L.C. Talbot,

- *glossaries FAQ*, Nicola L. C. Talbot, glossaries FAQ

- *Glossaries, Nomenclature, Lists of Symbols and Acronyms*, Nicola L. C. Talbot, link

5.5   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 (which comes bundled as standard with LaTeX). Recently, BibTeX has been succeeded by BibLaTeX, a tool configurable within LaTeX syntax.

BibTeX provides for the storage of all references in an external, flat-file database. (BibLaTeX uses this same syntax.) This database can be referenced in any LaTeX document, and citations made to any record that is contained within the file. This is often more convenient than embedding them at the end of every document written; a centralized bibliography source 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 sources concerning topic A (a.bib) and another concerning topic B (b.bib). When writing about topic AB, both of these files can be linked into the document (perhaps in addition to sources ab.bib specific to topic AB).

5.5.1   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, you might not 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 mandatory argument, which I supplied after the begin statement, is telling LaTeX how wide the item label will be when printed. Note however, that the number itself is not the parameter, but the number of digits is. Therefore, I am effectively telling LaTeX that I will only need reference labels of one character in length, which ultimately means no more than nine references in total. If

you want more than nine, then input any two-digit number, such as '56' which allows up to 99 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.

5.5.2   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 specifically

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[chapter, 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. Here's an example:

\cite{citation01,citation02,citation03}

The result will then be shown as citations inside the same brackets, depending on the citation style.

Bibliography styles

There are several different ways to format lists of bibliographic references and the citations to them in the text. These are called citation styles, and consist of two parts: the format of the abbreviated citation (i.e. the marker that is inserted into the text to identify the entry in the list of references) and the format of the corresponding entry in the list of references, which includes full bibliographic details.

Abbreviated citations can be of two main types: numbered or textual. Numbered citations (also known as the Vancouver referencing system) are numbered consecutively in order of appearance in the text, and consist in Arabic numerals in parentheses **(1)**, square brackets **[1]**, superscript[1], or a combination thereof[1]. Textual citations (also known as the Harvard referencing system) use the author surname and (usually) the year as the abbreviated form of the citation, which is normally fully **(Smith 2006)** or partially enclosed in parenthesis, as in **Smith (2006)**. The latter form allows the citation to be integrated in the sentence it supports.

Below you can see three of the 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

Here are some more often used styles:

However, keep in mind that you will need to use the natbib package to use most of these.

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

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. 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[options]{natbib}

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.

Customization 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.

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 {,~}.

Some of the options controlled by \bibpunct are also accessible by passing options to the natbib package when it is loaded. These options also allow some other aspect of the bibliography to be controlled, and can be seen in the table (right).

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, it's a short document and you can learn even more about how to use it.

5.5.3   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. I will 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*, or the *BibTeX 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. Using \usepackage[utf8]{inputenc} in the preamble to the .tex source file can get round this, as the accented characters can just be stored in the .bib file without any need for special markup. This allows a consistent format to be kept throughout the .bib file, avoiding the need to use braces when there are umlauts to consider.

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 and also this field reference 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. Much of the information contained therein is repeated in the following table for your convenience.

+ Required fields, O Optional fields

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 Mittelbach 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

Be careful if you copy the following templates, the % sign is not valid to comment out lines in bibtex files. If you want to comment out a line, you have to put it outside the entry.

@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, title, chapter and/or pages, publisher, year.
- Optional fields: volume/number, series, type, address, edition, month, note.

@inbook{Xinbook, author = "", editor = "", title = "", chapter = "", pages = "", publisher= "", %volume = "", %number = "", %series = "", %type = "", %address= "", %edition= "", year = "", %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.

@incollection{Xincollection, author = "", title = "", booktitle= "", publisher= "", %editor = "", %volume = "", %number = "", %series = "", %type = "", %chapter= "", %pages = "", %address= "", %edition= "", year = "", %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.

@inproceedings{Xinproceedings, author = "", title = "", booktitle = "", %editor = "", %volume = "", %number = "", %series = "", %pages = "", %address = "", %organization = "", %publisher = "", year = "", %month = "", %note = "", }

@manual Technical manual

- Required fields: title.
- Optional fields: author, organization, address, edition, month, year, note.

@manual{Xmanual, title = "", %author = "", %organization = "", %address = "", %edition = "", year = "", %month = "", %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.

@phdthesis{Xphdthesis, author = "", title = "", school = "", %address = "", year = "", %month = "", %keywords = "", %note = "", }

@proceedings The proceedings of a conference.

- Required fields: title, year.
- Optional fields: editor, volume/number, series, address, month, organization, publisher, note.

@proceedings{Xproceedings, title = "", %editor = "", %volume = "", %number = "", %series = "", %address = "", %organization = "", %publisher = "", year = "", %month = "", %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.

@unpublished{Xunpublished, author = "", title = "", %year = "", %month = "", note = "", }

Not standard templates

@patent BiBTeX entries can be exported from Google Patents.

(see Cite Patents with Bibtex for an alternative)

@collection

@electronic

@Unpublished For citing arXiv.org papers in a REVTEX-style article

(see REVTEX Author's guide)

Preserving case of letters

In the event that BibTeX has been set by the chosen style 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! It is even possible that lower-case letters may need to be preserved - for example if a chemical formula is used in a style that sets a title in all caps or small caps, or if "pH" is to be used in a style that capitalises all first letters.

> title = "The {LaTeX} Companion",

However, **avoid** putting the whole title in curly braces, as it will look odd if a different capitalization format is used:

> title = "{The LaTeX Companion}",

For convenience though, many people simply put double curly braces, which may help when writing scientific articles for different magazines, conferences with different BibTex styles that do sometimes keep and sometimes not keep the capital letters:

> title = {{The LaTeX Companion}},

As an alternative, try other BibTex styles or modify the existing. The approach of putting only relevant text in curly brackets is the most feasible if using a template under the control of a publisher, such as for journal submissions. Using curly braces around single letters is also to be avoided if possible, as it may mess up the kerning, especially with biblatex,[1] so the first step should generally be to enclose single words in braces.

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="Rabbert Klein", 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} or \bibliography{../sample1} (if the .bib file is in the parent directory of the .tex document that calls it).

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 than 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 from 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. Each pass 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 (or more precisely the corresponding AUX file, however 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 %:r (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. 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 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. 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.aux   pdflatex latex_source_code.tex pdflatex latex_source_code.tex

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 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.

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) 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 straightforward. And if you require further guidance, then there is a comprehensive manual 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 automati-cally 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 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}

Showing unused items

Usually LaTeX only displays the entries which are referred to with \cite. It's possible to make uncited entries visible:

\nocite{Name89} % Show Bibliography entry of Name89 \nocite{*} % Show all Bib-entries

Getting bibliographic data

Many online databases provide bibliographic data in BibTeX-Format, making it easy to build your own database. For example, Google Scholar offers the option to return properly formatted output, which can also be turned on in the settings page.

One should be alert to the fact that bibliographic databases are frequently the product of several generations of automatic processing, and so the resulting Bib-Tex code is prone to a variety of minor errors, especially in older entries.

Helpful tools

Literatur-Generator
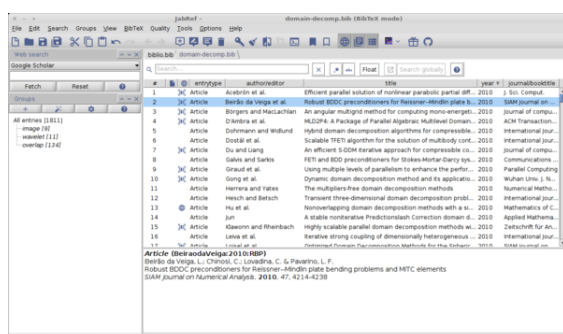
JabRef
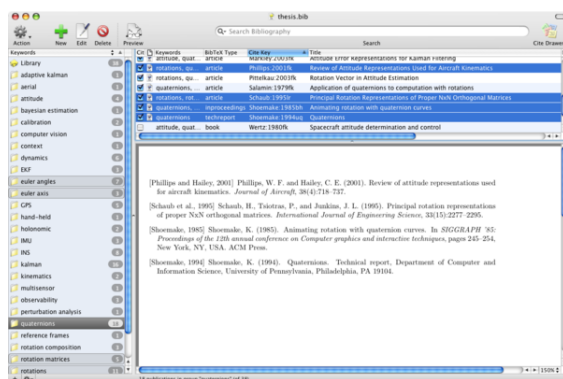
BibDesk

- **BibDesk** 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.

- **BibSonomy** — A free social bookmark and publication management system based on BibTeX.

- **BibTeXSearch** BibTeXSearch is a free searchable BibTeX database spanning millions of academic records.

- **Bibtex Editor** - An online BibTeX entry generator and bibliography management system. Possible to import and export Bibtex files.

- **Bibwiki** Bibwiki is a Specialpage for MediaWiki to manage BibTeX bibliographies. It offers a straightforward way to import and export bibliographic records.

- **cb2Bib** The cb2Bib is a tool for rapidly extracting unformatted, or unstandardized bibliographic references from email alerts, journal Web pages, and PDF files.

- **Citavi** 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.

- **CiteULike** 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.

- **DokuWiki** 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

- **Ebib** — a BibTeX database manager for Emacs, well resolved and never more than a few keystrokes away.

- **JabRef** 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.

- **KBib** 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.

- **KBibTeX** 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.

- **Literatur-Generator** is a German-language online tool for creating a bibliography (Bibtex, Endnote, Din 1505, ...).

- **Mendeley** Mendeley is cost-free academic software for managing PDFs which can manage a bibliography in Open Office and read BibTeX.

- **Qiqqa** Qiqqa is a free research manager that has built-in support for automatically associating BibTeX records with your PDFs and a 'BibTeX Sniffer' for helping you semi-automatically find BibTeX records.

- **Referencer** Referencer is a Gnome application to organise documents or references, and ultimately generate a BibTeX bibliography file.

- **Synapsen** — Hypertextual Card Index / Reference Manager with special support for BiBTeX / biblatex, written in Java.

- **Zotero** Zotero is a free and open reference manager working as a Firefox plugin or standalone application, capable of importing and exporting bib files.

Summary

Although it can take a little time to get to grips with Bib-TeX, 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.

5.5.4 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}

Or, if you are using BibTeX, your references will be saved in a .bib file, and your TeX document will include the bibliography by these commands:

\bibliographystyle{plain} \bibliography{mybibtexfile}

Both of these examples will create a chapter-like (or section-like) output showing all your references. But even though the resulting "References" looks like a chapter or section, it will not be handled quite the same: it will not appear in the Table of Contents.

Using tocbibind

The most comfortable way of adding your bibliography to the table of contents is to use the dedicated package **tocbibind** that works with many standard document classes. Simply include this code in the preamble of your document:

\usepackage[nottoc]{tocbibind}

This will include the Bibliography in the Table of Contents without numbering. If you want to have proper numbering, include the following code in the preamble:

\usepackage[nottoc,numbib]{tocbibind}

The **tocbibind** package can also handle including the List of Figures, List of Tables and the Table of Contents itself in the Table of Contents. It has many options for numbering, document structure etc. to fit almost any scenario. See the **tocbibind** CTAN page for detailed documentation.

Other methods

As unnumbered item If you want your bibliography to be in the table of contents, just add the following two lines just before the *thebibliography* environment:

\clearpage%   or   cleardoublepage   \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.

\clearpage% or 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.

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*{−1em}} followed by \vspace*{−1em} to counteract the extra space the blank \refname inserts.

If you are using BibTeX, the \bibliography command, and the book or report class, you will need to redefine \bibname instead of \refname like so.

\renewcommand{\bibname}{\section{Sources}} % Redefine bibname \bibliographystyle{IEEEtran} % Set any options you want \bibliography{your_bib_file_names} % Build the bibliography

5.5.5 biblatex

As we said before, biblatex is widely considered the `successor' of BibTeX. Intended as a full replacement for BibTeX, it is more configurable in its output and provides a multitude of new styles (for output) and fields (for the database) that can be used in a document. For now, refer to its comprehensive documentation on CTAN.

Entry and field types in .bib files

The following table shows most field types. Some field types are lists, either lists of person names, others are literal lists. A date can either be given in parts or full, some keys are necessary, page references are provided as ranges and certain special fields contain verbatim code. There are many kinds of titles.

Some entry types are hard to distinguish and are treated the same by standard styles:

- @article is the same as hypothetic *@inperiodical and therefore encompasses existing @suppperiodical

- @inbook = @bookinbook = @suppbook

- @collection = @reference

- @mvcollection = @mvreference

- @incollection = @suppcollection = @inreference

- @online = @electronic = @www

- @report = @techreport

- @thesis = @mastersthesis = @phdthesis

Some field types are defined, but the documentation does not say which entry types they can be used with. This is either because they depend on another field being set to be useful or they can always be used in a user-defined manner, but will never be used in standard styles:

- abstract, annotation

- entrysubtype

- file

- label

- library

- nameaddon

- origdate, origlocation, origpublisher

- origtitle, reprinttitle, indextitle

- pagination, bookpagination

- shortauthor, shorteditor, shorthand, shorthandintro, shortjournal, shortseries shorttitle

The only field that is always mandatory, is title. All entry types also require either date or year and they specify which of author and editor they expect or whether they can use both. Some field types can optionally be used with any entry type:

- addendum, note

- language

- pubstate

- urldate

All physical (print) entry types share further optional field types:

- url, doi

- eprint, eprintclass, eprinttype

Multimedia entry types

- @artwork
- @audio
- @image
- @movie
- @music
- @performance
- @video
- @software

and legal entry types

- @commentary
- @jurisdiction
- @legislation
- @legal
- @letter
- @review
- @standard

are defined, but not yet supported (well).

The entry types @bibnote, @set and @xdata are special.

Printing bibliography

Presuming we have defined our references in a file called references.bib, we add this to biblatex by adding the following to the preamble:

\addbibresource{references.bib}

Print the bibiography with this macro (usually at the end of the document body):

\printbibliography

Printing separate bibliographies   We want to separate the bibliography into papers, books and others

If the bib entries are located in multiple files we can add them like this:

We can also filter on other fields, such as entrysubtype. If we define our online resources like this:

we filter with \printbibliography[title={Online resources}, subtype=inet]

Example with prefix keys, subheadings and table of contents   As the numbering of the bibliographies are independent, it can be useful to also separate the bibliographies using prefixnumbers such as a, b and c. In addition we add a main heading for the bibliographies and add that to the table of contents.

To make Hyperref links point to the correct bibliography section, we also add \phantomsection before printing each bibliography

To add each of the bibliographies to the table of contents as sub-sections to the main Bibliography, replace heading=subbibliography with heading=subbibintoc.

5.5.6   Multiple bibliographies

Using multibib

This package is for multiple Bibliographies for different sections in your work. For example, you can generate a bibliography for each chapter. You can find information about the package on CTAN[2]

Using bibtopic

The bibtopic-Package[3] is created to differ the citations on more files, so that you can divide the bibliography into more parts.

\documentclass[11pt]{article}   \usepackage{bibtopic} \begin{document}   \bibliographystyle{alpha} \section{Testing}   Let's   cite   all   the   books: \cite{ColBenh:93}   and   \cite{Munt:93};   and an article: \cite{RouxSmart:95}.   File books.bib is used for this listing: \begin{btSect}{books} \section{References from books} \btPrintCited \end{btSect} Here, the articles.bib is used, and the listing is in plain-format instead of   the   standard   alpha.   \begin{btSect}[plain]{articles} \section{References from articles} \btPrintCited \section{Articles not cited} \btPrintNotCited \end{btSect} Just   print   all   entries   here   with   \btPrintAll   \begin{btSect}[plain]{internet}   \section{References from the internet} \btPrintAll \end{btSect} \end{document}

5.5.7   Notes and references

[1] The biblatex manual

[2] http://ctan.org/pkg/multibib

[3] http://ctan.org/pkg/bibtopic

This page uses material from Andy Roberts' Getting to grips with LaTeX with permission from the author.

5.6 More Bibliographies

This is a gentle introduction to using some of the bibliography functionality available to LaTeX users beyond the BibTeX basics. This introduction won't be discussing how to create new styles or packages but rather how to use some existing ones. It is worth noting that *Harvard*, for example, is a *citation* style. It is associated with an alphabetical reference list secondarily ordered on date, but the only strictly defined element of *Harvard* style is the citation in *author-date* format.

5.6.1 The example data

The database used for my examples contains just the following

@article{Erdos65, title = {Some very hard sums}, journal={Difficult Maths Today}, author={Paul Erd\H{o}s and Arend Heyting and Luitzen Egbertus Brouwer}, year={1930}, pages={30} }

5.6.2 The limits of BibTeX styles

Using cite.sty and BibTeX makes it very easy to produce **some** bibliography styles. But *author-date* styles - for example the often mentioned, never defined "Harvard" - are not so easy. It's true that you can download some *.bst* files from CTAN that will handle some variants but using them is not always straightforward. This guide deals with *Natbib* a supplementary package that can access *.bib* files and has sophisticated functionality for producing custom or default author-year format citations and bibliographies as well as the numerical styles handled by BibTeX.

5.6.3 Natbib

Natbib is a package created by Patrick Daly as a replacement for the *cite.sty* package when *author-date* citation styles are required. Natbib provides three associated bibliography styles:

* plainnat

* abbrvnat

* unsrtnat

which correspond to the three styles available by default in BibTeX where you have a plain numbered style, an abbreviated numbered style and an unsorted numbered style.

Alongside these new styles is an extended set of citation commands to provide flexible citation formats. These are

\citet[]{ }

and

\citep[]{ }

each of which has a number of variants.

The Preamble

All Natbib styles require that you load the package in your document preamble. So, a skeleton LaTeX file with Natbib might look like this:

\documentclass[]{article} \usepackage[round]{natbib} \begin{document} Document body text with citations. \bibliographystyle{plainnat} \bibliography{myrefs} \end{document}

Options   Options available with Natbib can be specified in the brackets on the \usepackage command. Among them are:

Clearly some of these options require explanation but that will be achieved via examples below. For now, we just note that they can be passed through *\usepackage[]{}* in the preamble of your LaTeX file.

5.6.4 Citation

Basic Citation Commands

To cite with Natbib, use the commands *\citet* or *\citep* in your document. The "plain" versions of these commands produced abbreviated lists in the case of multiple authors but both have * variants which result in full author listings. We assume the use of the *round* option in these examples.

\citet and \citet*   The *\citet* command is used for *textual* citations, that is to say that author names appear in the text outside of the parenthetical reference to the date of publication. This command can take options for chapter, page numbers etc. Here are examples

Here are the *\citet** versions

\citep and \citep*   The *\citep* command is used where the author name is to appear inside the parentheses alongside the date.

Here are the *\citep** versions

The Reference List

Having dealt with basic varieties of citation, we turn to the creation of the bibliography or reference list.

Inserting a correct and correctly formatted bibliography when using Natbib is no different than when using plain BibTeX. There are two essential commands -

\bibliography{mybibliographydatabase}

which LaTeX interprets as an instruction to read a bibliographic database file (eg myrefs.bib) and insert the relevant data here, and

\bibliographystyle{plainnat}

which specifies how the data are to be presented.

Above the three basic Natbib styles were mentioned as analogues of the partially homonymous styles in BibTeX. Let us imagine documents bearing citations as in the section about citation above. Here is, approximately, how these citations would appear in plainnat.

Paul Erdos, Arend Heyting, and Luitzen Egbertus Brouwer. *sums*. Kluwer, Dortmund, 1930.

What more is there?

This covers the basic functionality provided by the package Natbib. It may not, of course, provide what you are looking for. If you don't find what you want here then you should probably next investigate *harvard.sty* which provides a slighly different set of author-date citation functions. Providing a gentle guide to *harvard.sty* is my next rainy day project.

Chapter 6

Special Documents

6.1 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.

6.1.1 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). If you want the same body in all the letters, you may want to consider putting the entire body in a new command like \newcommand{\BODY}{actual body} and then using \BODY in all the letters.

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:

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

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.

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.

6.1.2 Envelopes

Using the envlab package

The envlab package provides customization to the \makelabels command, allowing the user to print on any of an assortment of labels or envelope sizes. For example, beginning your LaTeX file the following way produces a document which includes the letter and a business-size (#10) envelope on the following page.

Refer to the envlab user guide for more information about this capable package. Note that the envlab package has issues displaying characters outside the base ASCII character set, see this bug report for more information.

Using the geometry package

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).

```
FROM-NAME
FROM-STREET ADDRESS
FROM-CITY, STATE,  ZIP



                              TO-NAME
                              TO-STREET ADDRESS
                              TO-CITY, STATE,  ZIP
```

A sample envelope to be printed in landscape mode.

Printing

The above will certainly take care of the spacing but the actual printing is between you and your printer. One user reports that printing envelopes created with envlab is relatively painless. If you use the geometry package, 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

It is reported 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.

6.1.3 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 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.

Joe Bloggs
21 Bridge Street
Smallville
Dunwich DU3 4WE
Phone: 0123 45679

Joe Bloggs, 21 Bridge Street , Smallville , Dunwich DU3 4WE

Director
Doe & Co
35 Anthony Road
Newport
Ipswich IP3 5RT

Customer no.   Date
2342   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.

6.1.4   Reference: letter.cls commands

6.1.5   Sources

- KOMA-Script - The Guide

6.2   Presentations

LaTeX can be used for creating presentations. There are several packages for the task, including the beamer package.

6.2.1   The Beamer package

The beamer package is provided with most LaTeX distributions, but is also available from CTAN. If you use Mik-TeX, all you have to do is to include the beamer package and let LaTeX download all wanted packages automatically. The documentation explains the features in great detail. You can also have a look at the PracTex article **Beamer by Example**.[1]

The beamer package also loads many useful packages including hyperref.

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 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}
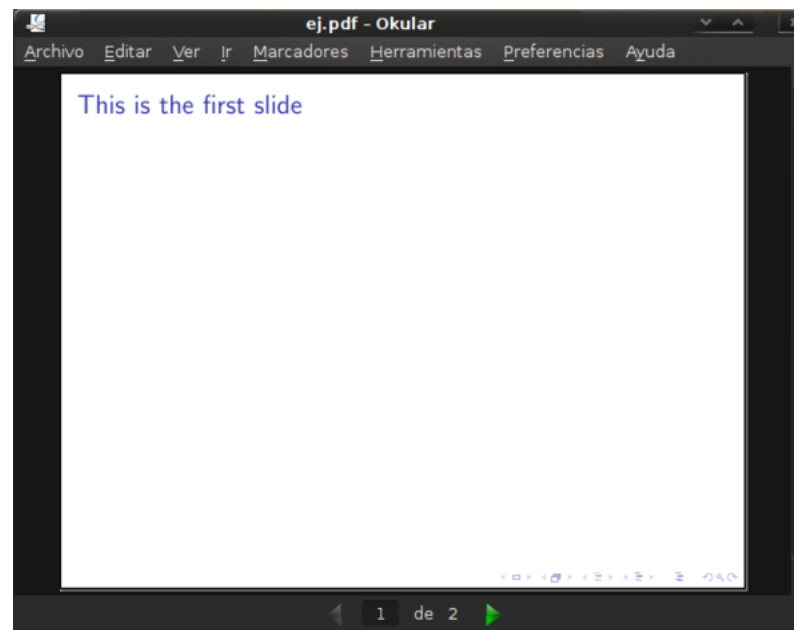
The usual environments (itemize, enumerate, equation, etc.) may be used.

Inside frames, you can use environments like block, theorem, proof, ... Also, \maketitle is possible to create the frontpage, if title and author are 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 a pdf reader with a fullscreen mode, such as Okular, Evince or Adobe Reader. 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 First, 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[Universities Here and There] % (optional) { \inst{1}% Institute of Computer Science\\ University Here \and \inst{2}% Institute of Theoretical Philosophy\\ University There } \date[KPT 2004] % (optional) {Conference on Presentation Techniques, 2004} \subject{Computer Science}

Then, in the document, you add the title page :

\frame{\titlepage}

Table of Contents The table of contents, with the current section highlighted, is displayed by:

\begin{frame} \frametitle{Table of Contents} \tableofcontents[currentsection] \end{frame}

This can be done automatically at the beginning of each section using the following code in the preamble:

\AtBeginSection[] { \begin{frame} \frametitle{Table of Contents} \tableofcontents[currentsection] \end{frame} }

Or for subsections:

\AtBeginSubsection[] { \begin{frame} \frametitle{Table of Contents} \tableofcontents[currentsection,currentsubsection] \end{frame} }

Sections and subsections As in all other LaTeX files, it is possible to structure the document using

\section[Section]{My section}

,

\subsection[Subsection]{My subsection}

and

\subsubsection[Subsubsection]{My subsubsection}

Those commands have to be put before and between frames. They will modify the Table of contents with the optional argument. The argument in brackets will be written on the slide, depending on the theme used.

References (Beamer) Beamer does not officially support BibTeX. Instead bibliography items will need to be partly set "by hand" (see beameruserguide.pdf 3.12). The following example shows a references slide containing two entries:

\begin{frame}[allowframebreaks] \frametitle<presentation>{Further Reading} \begin{thebibliography}{10} \beamertemplatebookbibitems \bibitem{Autor1990} A.~Autor. \newblock {\em Introduction to Giving Presentations}. \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}

As the reference list grows, the reference slide will divide into two slides and so on, through use of the allowframebreaks option. Individual items can be cited after adding an 'optional' label to the relevant bibitem stanza. The citation call is simply \cite. Beamer also supports limited customization of the way references are presented (see the manual). Those who wish to use natbib, for example, with Beamer may need to troubleshoot both their document setup and the relevant BibTeX style file.

The different types of referenced work are indicated with a little symbol (e.g. a book, an article, etc.). The Symbol is set with the commands beamertemplatebookbibitems and beamertemplatearticlebibitems . It is also possible to use setbeamertemplate directly, like so

\begin{frame}[allowframebreaks] \frametitle<presentation>{Further Reading} \begin{thebibliography}{10} \setbeamertemplate{bibliography item}[book] \bibitem{Autor1990} A.~Autor. \newblock {\em Introduction to Giving Presentations}. \newblock Klein-Verlag, 1990. \setbeamertemplate{bibliography item}[article] \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}

Other possible types of bibliography items, besides book and article , include e.g. online , triangle and text . It is also possible to have user defined bibliography items by including a graphic.

If one wants to have full references appear as foot notes, use the \footfullcite. For example, it is possible to use

\documentclass[10pt,handout,english]{beamer} \usepackage[english]{babel} \usepackage[backend=biber,style=numeric-comp,sorting=none]{biblatex} \addbibresource{biblio.bib} \begin{frame} \frametitle{Title} A reference~\footfullcite{ref_bib}, with ref_bib an item of the .bib file. \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. Frankfurt, Madrid, Berlin, etc.).

This Theme Matrix 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 full list of themes is:

Color themes, typically with animal names, can be specified with

\usecolortheme{beaver}

The full list of color themes is:

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}

Colors can be defined as usual:

\definecolor{chocolate}{RGB}{33,33,33}

Block styles can also be defined:

\setbeamertemplate{blocks}[rounded][shadow=true] \setbeamertemplate{background canvas}[vertical shading][bottom=white,top=structure.fg!25] \setbeamertemplate{sidebar canvas left}[horizontal shading][left=white!40!black,right=black]

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 ren-

dered 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 Open-Type 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=1.28, 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 off 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]{ % ... }

The *allowframebreaks* option will auto-create new frames if there is too much content to be displayed on one.

\frame[allowframebreaks]{ % ... }

Before using any verbatim environment (like listings), you should pass the option fragile to the frame environment, as verbatim environments need to be typeset differently. Usually, the form fragile=singleslide is usable (for details see the manual). Note that the fragile option may not be used with \frame commands since it expects to encounter a \end{frame}, which should be alone on a single line.

\begin{frame}[fragile] \frametitle{Source code} \begin{lstlisting}[caption=First C example] int main() { printf("Hello World!"); return 0; } \end{lstlisting}

\end{frame}

Hyperlink navigation

Internal and external hyperlinks can be used in beamer to assist navigation. Clean looking buttons can also be added.

By default the beamer class adds navigation buttons in the bottom right corner. To remove them one can place

\beamertemplatenavigationsymbolsempty

in the preamble.

Animations

The following is merely an introduction to the possibilities in beamer. Chapter 8 of the beamer manual provides much more detail, on many more features.

Making items appear on a slide is possible by simply using 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}

Text or figures after \pause will display after one of the following events (which may vary between PDF viewers): pressing space, return or page down on the keyboard, or using the mouse to scroll down or click the next slide button. Pause can be used within \itemize etc.

Text animations   For text animations, for example in the itemize environment, it is possible to specify appearance and disappearance of text by using <a-b> where **a** and **b** are the numbers of the events the item is to be displayed for (inclusive). For example:

\begin{itemize} \item This one is always shown \item<1-> The first time (i.e. as soon as the slide loads) \item<2-> The second time \item<1-> Also the first time \only<1-1> {This one is shown at the first time, but it will hide soon (on the next event after the slide loads).} \end{itemize}

A simpler approach for revealing one item per click is to use \begin{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}

In all these cases, pressing page up, scrolling up, or clicking the previous slide button in the navigation bar will backtrack through the sequence.

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 any \itemize[<+->] environments visible 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 specify precisely the behavior 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. Blocks and columns can be used inside each other.

Columns   Example

\begin{frame}{Example   of   columns   1}   \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}{Example of columns 2} \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 (a blank heading can be used). This allows to visually distinguish parts of a slide easily. There are three basic types of block. Their formatting 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.[2]

\hypersetup{pdfstartview={Fit}} % fits the presentation to the window when first displayed

Numbering slides

It is possible to number slides using this snippet:

\insertframenumber/\inserttotalframenumber

However, this poses two problems for some presentation authors: the title slide is numbered as the first one, and the appendix or so-called "backup" (aka appendix, reserve) slides are included in the total count despite them not being intended to be public until a "hard" question is asked.[3] This is where two features come in:

- Ability to reset the frames counter at any slide. For instance, this may be inserted at the title slide to avoid counting it:

\addtocounter{framenumber}{−1}

Or alternatively this:

\setcounter{framenumber}{0}   or   \setcounter{framenumber}{1}

- The first of the above applies to section slides to avoid counting them.

- This stuff works around the problem of counting the backup slides:

% (Thanks, David Gleich!) % All your regular slides % After your last numbered slide \appendix \newcounter{finalframe} \setcounter{finalframe}{\value{framenumber}} % Backup frames \setcounter{framenumber}{\value{finalframe}} \end{document}

6.2.2 The powerdot package

The powerdot package is an alternative to beamer. It is available from CTAN. The documentation 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}

6.2.3 References

[1] Andrew Mertz and William Slough, *Beamer by Example*

[2] Other possible values are defined in the hyperref manual

[3] Appendix Slides in Beamer: Controlling frame numbers

6.2.4 Links

- Wikipedia:Beamer (LaTeX)

- Beamer user guide (pdf) from CTAN

- The powerdot class (pdf) from CTAN

- A tutorial for creating presentations using beamer

6.3 Teacher's Corner

6.3.1 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], the **mathexm** document class[3], or the **exsheets** package[4].

6.3.2 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 :

You can replace the 3 first lines with the following :

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.

6.3.3 References

[1] examdoc Using the exam document class

[2] Probsoln Creating problem sheets optionally with solutions

[3] mathexm documentation

[4] exsheets documentation Create exercise sheets and exams

6.4 Curriculum Vitae

A *curriculum vitæ* or résumé has a universal requirement: its formatting must be flawless. This is a great example of cases where the power of LaTeX comes to the front. Thanks to its strong typographical stance, LaTeX is definitely a document processor of choice to write a CV.

Of course you can design your own CV by hand. Otherwise, you may want to use a dedicated class for that task. This way, writing a CV in LaTeX is as simple as filling the forms, and you are done. The important thing to mention

is that a class or package not only defines the syntax of the form to be filled out, but also the layout. While filling out a form is very easy, changing a layout that is supposed to be fixed, is extremely hard. Best results will be achieved when doing a CV from scratch, using list environments and tabulars.

> ⇒ Be aware of res.cls which is still used by some people. It is a very old class, that is nearly impossible to customize. Only very experienced users should use res.cls. ⇐

A full list of CV packages is available at CTAN.

6.4.1 europecv

6.4.2 moderncv

From CTAN:

Moderncv provides a documentclass for typesetting modern curriculums vitae, both in a classic and in a casual style. It is fairly customizable, allowing you to define your own style by changing the colours, the fonts, etc.

The official package provides some well commented templates which may be a good start. You can find those templates in your distribution (if documentation is installed along packages) or ultimately on CTAN.

We will not repeat the templates here, so we will only provide a crash course. You should really have a look at the templates for more details.

First document

Most commands are self-explanatory.

Theme previews

- Themes
- Banking black theme
- Classic green theme

6.4.3 Multilingual support

It is especially convenient for résumés to have only one document for several output languages, since many parts are shared among versions (personal data, structure, etc.).

LaTeX with appropriate macros provide a comfortable way to manage it. See Internationalization.

6.4.4 References

Chapter 7

Creating Graphics

7.1 Introducing Procedural Graphics

In the Importing Graphics 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:

There are several packages supporting the creation of graphics directly in LaTeX, including picture, xy-Pic and PGF/TikZ, described in the following sections.

Compared to WYSIWIG tools like Xfig or Inkscape, this approach is more time consuming, but leads to much better results. Furthermore, the output is flawlessly integrated to your document (no contrast in size nor fonts).

See the Importing Graphics for more details on graphics importation and some attempts to circumvent to integration issue.

7.1.1 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 from TeX source, you have to go from TeX to DVI to PDF, losing hyperlinks, metadata, and microtypographic features of pdflatex in the process.

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 are 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 are the PGF/TikZ and Asymptote systems. While the previous systems (picture, epic, pstricks or metapost) focus on the *how* to draw, TikZ and Asymptote focus more on the *what* to draw. One could say that TikZ and Asymptote are to drawing in LaTeX as LaTeX is to digital typesetting. It's recommended to use one of these if your LaTeX distribution includes it. TikZ is a pure (La)TeX system, not reliant on external software, while Asymptote is an external system which integrates seamlessly with (La)TeX. If using Asymptote, it is very helpful to use latexmk to manage the compilation steps.

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 LaTeX/Importing_Graphics). However most software does not support LaTeX fonts or mathematical notation, which can result in not suitable and inconsistent graphics. There are several solutions to this problem.

7.2 MetaPost

7.3 Picture

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 pict2e, epic, eepic or pstricks enhance the original picture environment, and greatly strengthen the graphical power of LaTeX.

7.3.1 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

or

The first pair, (x, y) , affects 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.

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

The default value of \unitlength is 1pt.

Most drawing commands have one of the two forms

or

Bézier curves are an exception. They are drawn with the command

With the package picture absolute dimension (like 15pt) and expression are allowed, in addition to numbers relative to \unitlength.

7.3.2 Line segments

Line segments are drawn with the command:

The \line command has two arguments:

1. a direction vector,

2. a "length" (sort of: this argument is the vertical length in the case of a vertical line segment and in all other cases the horizontal distance of the line, 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.

7.3.3 Arrows

Arrows are drawn with the command

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.

7.3.4 Circles

The command

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, pstricks, or tikz.

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.

7.3.5   Text and formulae

As this example shows, text and formulae can be written in the environment with the \put command in the usual way:

7.3.6   *\multiput* and *\linethickness*

The command

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!

7.3.7   Ovals

The command

or

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.

7.3.8   Multiple use of predefined picture boxes

A picture box can be *declared* by the command

then *defined* by

and finally arbitrarily often be *drawn* by

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.

7.3.9   Quadratic Bézier curves

The command

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.

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.

7.3.10   Catenary

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

its lower left corner (marked by the black disk) is assigned the coordinates (−2.5,−0.25).

7.3.11   Plotting graphs

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).

7.3.12 The *picture* environment and gnu-plot

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. See [] for a tutorial.

7.4 PGF/TikZ
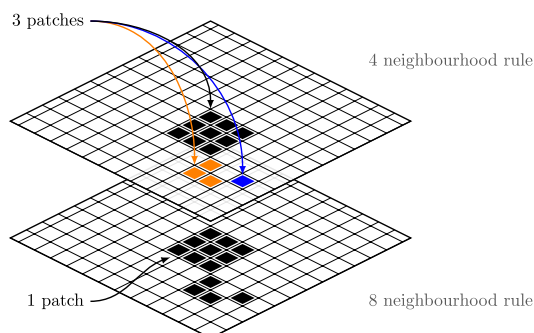
Example of graphics done with Tikz. Note the slightly translucent top layer.

One way to draw graphics directly with TeX commands is PGF/TikZ. TikZ 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 TikZ ("TikZ ist *kein* Zeichenprogramm") is the frontend layer with a special syntax, making the use of PGF easier. TikZ commands are prevalently similar to Metafont, the option mechanism is similar to PsTricks syntax.

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.

Other packages building on top of TikZ (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 TikZ are presented.

7.4.1 Loading Package, Libraries - tikzpicture environment

Using TikZ in a LaTeX document requires loading the tikz package:

\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⟩

or different for height and width, e.g:

xscale=2.5, yscale=0.5

7.4.2 Specifying Coordinates

Coordinates are specified in round brackets in an arbitrary TEX dimension either using Cartesian coordinates (comma separated), e.g. 1cm in the x direction and 2pt in the 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)

7.4.3   Syntax for Paths

A path is a series of straight and curved line segments(in a simplified explanation). The instruction has to end with a semicolon.

\path[<options>]⟨specification⟩;

One instruction can spread over several lines, or several instructions can be put on one line.

Path actions

Options for path actions are e.g: "draw", "fill", "pattern", "shade" (a variation on filling that changes colors 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 actions

Geometric path options: "rotate=<angle in degree>", "xshift=<length>", "yshift=<length>", "scaling=<factor>", "xscale=<factor>", "yscale=<factor>".

Color

Color options for drawing paths: "color=<color name>", "draw=<line color>", "opacity=<factor>". Following colors are predefined: red, green, blue, cyan , magenta, yellow, black, gray, darkgray, lightgray, brown, lime, olive, orange, pink, purple, teal, violet and white.

Line width

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 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

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", "dash-dotted", "densely dotted", "loosely dotted", "double".

Options for filling paths are e.g. "fill=<fill color>", "pattern=<name>", "pattern color=<color>"

7.4.4   Drawing straight lines

Straight lines are given by coordinates separated by a double minus,

The first coordinate represents a move-to operation. This is followed by a series of "path extension operations", like "-- (coordinates)".

The same path with some drawing options:

A connected path can be closed using the "--cycle" operation:

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):

Two points can be connected by straight lines that are only horizontal and vertical. For a connection that is first horizontal and then vertical, use

or first vertical then horizontal, use

7.4.5   Drawing curved paths

Curved paths using a Bezier curve can be created using the "..controls() ..()" command, with one or two control points.

7.4.6 User-defined paths

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.

(The syntax for a bend to the right may seem a little counter-intuitive. Think of it as an instruction to veer to the right at the beginning of the path and then smoothly curve to the end point, not as saying that the path curves to the right throughout its length.)

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.

The fill color "green!20!white" means 20% green and 80% white mixed together.

7.4.7 Circles, ellipses

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.

7.4.8 Arcs

The command "arc" creates a part of a circle or an ellipse:

Or in an alternative syntax:

\draw (0,0) arc[radius = 8mm, start angle= 0, end angle= 270]; \draw (0,0) arc[x radius = 1.75cm, y radius = 1cm, start angle= 0, end angle= 315];

7.4.9 Special curves

There are many more predefined commands for special paths, like "grid", "parabola", "sin", "cos" (sine or cosine curve in the interval [0,π/2]).

The option "help lines" denotes "fine gray".

To add arrow tips there are simple options for the drawing command:

A loop can be realized by "\foreach ⟨variable⟩ in {⟨list of values⟩} ⟨commands⟩".

PGF also has a math engine which enables you to plot functions:

\draw [domain=<xmin>:<xmax>] plot (\x, {function});

Many functions are possible, including factorial(\x), sqrt(\x), pow(\x,y), exp(\x), ln(\x), log10(\x), log2(\x), abs(\x), mod(\x,y), round(\x), floor(\x), ceil(\x), sin(\x), cos(\x), tan(x), min(\x,y,), and max(\x,y). The trigonometric functions assume that x is in degrees; to express x in radians follow it with the notation "r", e.g., sin(\x r). Two useful constants are e, which is equal to 2.718281828, and pi, which is equal to 3.141592654.

An example with two functions:

7.4.10 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 node command is shown as a simple example:

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 placed beside 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".

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.

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 second 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 the name "(y)".

Equivalent to

\coordinate (x) at (0,0); \coordinate (y) at (3,1); \draw (x) -- (y);

Multiline text can be included inside a node. A new line is indicated by double backslash "\\", but additionally you have to specify the alignment using the node option "align=". Here an example:

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.

Once the node x has been defined, you can use anchors as defined above relative to (x) as "(x.<anchor>)", like "(x.north)".

7.4.11   Examples

Example 1

Example 2

Example 3: A Torus

Example 4: Some functions

7.5   PSTricks

PSTricks is a set of extensions. The base package is pstricks, other packages may be loaded when required.

The xcolor extension gets loaded along PSTricks, so there is no need to load it manually.

PSTricks has one technical specification: it uses PostScript internally, hence the name. Thus you cannot use the pdftex or pdflatex compilers, you will need to use dvips to get your proper document. It is still possible to get PDF from PS files thanks to ps2pdf. There is also the possibility to use the PDFTricks extension, which makes it feasible to use pdflatex together with PSTricks commands.

However, if you have installed the package xetex-pstricks, you can use pstricks with xetex or xelatex without modification of source file.

7.5.1   The pspicture environment

PSTricks commands are usually placed in a pspicture environment.

\begin{pspicture}(x1,y1) % ... \end{pspicture}

The first argument between parentheses specifies the coordinates of the upper-right corner of the picture. The bottom-left corner is at (0,0) and is placed at the reference point of the next character in the LaTeX document. It is also possible to specify the coordinates (x0,y0) of the bottom-left corner:

\begin{pspicture}(x0,y0)(x1,y1) % ... \end{pspicture}

Thus the size of the picture is *(x1-x0)x(y1-y0)*. The default unit for coordinates is centimeters (cm); this can be changed with \psset, as in \psset{unit=1bp}. Any TeX dimension is allowed.

7.5.2   Fundamental objects

Lines and polylines

A simple line gets printed with

\psline(x0,y0)(x1,y1)

To get a vector, add an arrow as parameter:

\psline{->}(x0,y0)(x1,y1)

You can add as many points as you want to get a polyline:

\psline(x0,y0)(x1,y1)(x2,y3)…(xn,yn)

To get rounded corners, add the following option:

\psline[linearc=0.2]{->}(0,0)(2,1)(1,1)

or

\psline[linearc=0.2,arrows=->](0,0)(2,1)(1,1)

Rectangles

\psframe(x0,y0)(x1,y1) \psframe*(x0,y0)(x1,y1)

The starred version prints a filled rectangle. Use the following parameter to get rounded corners:

\psframe[framearc=0.2](x0,y0)(x1,y1)

Polygons

Polygons are always closed. The syntax is the same as for \psline:

\pspolygon(x0,y0)(x1,y1)(x2,y2)...(xn,yn)

As for rectangles, the starred version prints a filled polygon. And the linearc=0.2 option will print rounded corners.

Circles, arc and ellipses

Starred version fills the shape.

For circles, you need to provide center coordinates and radius:

\pscircle(x,y){r}

To restrict the drawing to an arc, append the starting and ending angles in trigonometric notation:

\psarc(x,y){r}{angle1}{angle2}

Finally, ellipses:

\psellipse(x,y)(horizontal_axis,vertical_axis)

Curves

\psparabola(x0,y0)(x1,y1)

will print a symmetric parabola with vertical asymptote, vertex *(x1,y1)* and ending at (x0,y0).

Use \psbezier to print a Bézier curve with an arbitrary number of control points. Arcs have at most 4 control points. Use the showpoints=true option to print the control points and the tangents.

Use \pscurve to print the interpolation of the given points. The \psecurve command omits the first and the last arcs.

7.5.3 Text

Use

\rput(x,y){text}

to print text. Provide an angle to rotate the text.

\rput{angle}(x,y){text}

You can provide the anchor of the text which will be at the specified coordinate.

\rput[t]{45}(5,5){text}

Available anchors:

- B, Bl, Br: baseline center, left and right.

- t, tl, tr: top center, left and right.

- b, bl, br: bottom center, left and right.

There is also the \uput command with further options:

\uput{distance}[angle](x,y){text}

The distance parameter is the distance from the coordinate.

PSTricks features several frame style for text.

- \psframebox{text}: rectangle.

- \psdblframebox{text}: double rectangle.

- \psshadowbox{text}: shaded rectangle.

- \pscirclebox{text}: circle.

- \psovalbox{text}: oval.

- \psdiabox{text}: diamond.

- \pstribox{text}: triangle.

Example:

\rput(5,5){\psdiabox*[fillcolor=green]{text}}

Using the pst-text extension, it is possible to draw a text path.

\pstextpath{shape}{text}

To print a text following a path without printing the path, you need to use \psset{linestyle=none}.

Example:

\usepackage{pst-text}   %   ...   \begin{pspicture}(5,5)   \psset{linestyle=none}   \pstextpath{\psline(0,0)(1,1)(2,0)}{triangle   text} \end{pspicture}

7.5.4 Grids

Without any parameter, the \psgrid command will print a grid all over the *pspicture*, with a spacing of 0.2 (*i.e.* 2mm). You can specify parameters:

- \psgrid(xmax,ymax): prints a grid from *(0,0)* to *(xmax,ymax)*.

- \psgrid(xmin,ymin)(xmax,ymax): prints a grid from *(xmin,ymin)* to *(xmax,ymax)*.

- \psgrid(x0,y0)(xmin,ymin)(xmax,ymax): prints a grid from *(xmin,ymin)* to *(xmax,ymax)*, one of the node is at *(x0,y0)*.

- griddots=value: the full line of the main graduations is replaced by a dotted line. The *value* is the number of dots per graduation.

- subgriddots=value: same as griddots but for sub-graduations.

- gridcolor=color,subgridcolor=color: color of graduations and sub-graduations.

- gridwidth=value,subgridwidth=value: width of the lines.

- subgriddiv=value: number of subgraduations between two main graduations.

- gridlabels=value: size of the label numbers.

- ticksize=value: self-explanatory.

- ticksize=valueneg valuepos: same as above, but *valueneg* specifies the size for negative coordinates, *valuepos* for positive coordinates.

- ticklinestyle=value: self-explanatory. *value* may be one of solid, dashed, dotted. This is useful for huge graduations (*i.e.* ticksize is high).

Example

\psgrid[griddots=5,   subgriddiv=0, gridlabels=0pt](−1,−1)(5,5)

Axis

If you want to add axes, use the pstricks-add extension with the following commands:

\psaxes(xmin,ymin)(xmax,ymax)
\psaxes(x0,y0)(xmin,ymin)(xmax,ymax)

(xmin,ymin) and *(xmax,ymax)* being the extreme, *(x0,y0)* being the intersection.

Options

- Dx=value and Dy=value defines the spacing between graduations.

- comma lets you use the comma as decimal separator.

- As for lines, {->} adds arrows on axes.

Example

\usepackage{pstricks-add}   %
...   \begin{pspicture}(−1,−1)(5,5)
\psaxes[comma,Dx=0.5,Dy=0.5]{->}(0,0)(3,3)
\end{pspicture}

7.5.5 Generic parameters

All shapes

These are to be added between square brackets.

- linewidth=value: if *value* is without unit, then the default unit is used.

- linecolor=color: *color* is as defined by the xcolor-package.

- linestyle=value: *value* is one of dashed,dotted.

- doubleline=true.

- showpoints=true: highlights points.

    - dotscale=value specifies the size of the points.
    - dotstyle=value where *value* is among:

        - *: disc
        - o: circle
        - +,x: cross
        - square, squarre*: starred version is filled.
        - diamond, diamond*
        - triangle, triangle*
        - etc.

For example

\pscircle[linewidth=0.2,linestyle=dashed,linecolor=blue](0,0){1}

To apply parameters globally:

\psset{linewidth=0.2,linestyle=dashed,linecolor=blue}
\pscircle(0,0){1}

This command also lets you change the default unit for lengths.

- unit=value

- xunit=value and yunit=value

value is a number with or without unit. This changes the scale of the drawings, but will not change the width of lines.

Open shapes

You can define the extreme of an open shape (line, polyline, arc, etc.) with an optional parameter {symbol1-symbol2}. There is a decent list of available symbols.

- < or >: arrow.
- << or >>: double arrow.
- |: bar.
- |*: centered bar.
- oo: circle.
- o: centered circle.
- **: disk.
- *: centered disk.
- |< or >|: arrow plus bar.
- cc: rounded extreme.
- c: centered rounded extreme.

Example:

\psline{|->>}(x0,y0)(x1,y1)

Closed shapes

For closed shape you may define the fillstyle.

- fillstyle=value: pattern. Possible values: crosshatch, crosshatch*, vlines, vlines*, hlines, hlines*, solid.
- fillcolor=color.
- hatchcolor=color.
- hatchwidth=value.
- hatchsep=value.
- hatchangle=value.

Example:

\pscircle[hatchcolor=blue,fillstyle=vlines](0,0){1}

7.5.6 Object location

The \rput and \uput commands can be used to move any object.

Example

\begin{pspicture}(5,5)   \psline{->}(0,0)(1,1)
\rput(1,1){\psline{->}(0,0)(1,1)} \end{pspicture}

or

\begin{pspicture}(5,5) \psline{->}(0,0)(1,1) \psline{->}(1,1)(2,2) \end{pspicture}

You can repeat the operation with \multirput:

\multirput(x0,y0)(xoffset, yoffset){times}{object}

You can use the same options as for \rput:

\multirput[reference]{angle}(x0,y0)(xoffset,   yoffset){times}{object}

With no text but with graphics only, you can use the \multips command:

\multips(x0,y0)(xoffset, yoffset){times}{object} \multips{angle}(x0,y0)(xoffset,yoffset){times}{object}

7.5.7 The PDFTricks extension

The original PSTricks package does not work with pdflatex, but thankfully PDFTricks allows us to bypass this limitation.

Usage

- Declare the PDFTricks packages in the preamble.
- Place all PSTricks extensions in a psinputs environment; place all PSTricks commands in a pdfpic environment.
- Compile with pdflatex -shell-escape <file>.

The -shell-escape parameter enables shell command calls. It is required for PDFTricks to run.

Example

\documentclass{article}   \usepackage{pdftricks}
\begin{psinputs}   \usepackage{pstricks}   \usepackage{multido} \end{psinputs} % ... \begin{document}
% ...   \begin{pdfpic} \psset{unit=\linewidth} \begin{pspicture}(0,0)(10,10)   [...]   \end{pspicture}
\end{pdfpic} % ... \end{document}

Another way to use PSTricks with pdflatex is the pst-pdf package.

7.6   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:

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.

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).

To make diagonals, just use more than one direction. In fact, you can repeat directions to make bigger arrows.

We can draw even more interesting diagrams by adding labels to the arrows. To do this, we use the common superscript and subscript operators.

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.

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

Notice the difference between the following two diagrams:

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, which contains links to several other tutorials as well as the reference documentation.

7.7   Creating 3D graphics

For creating three-dimensional graphics, there is basic functionality in the PGF/TikZ package, although drawing 3D graphics with PGF/TikZ is very non-flexible, mainly because it lacks functionality for identifying the surfaces that are covered by other surfaces and should be excluded from the rendered image.

A package that can handle this correctly is the pst-solides3d package.

Another way to create 3D graphics is to use Asymptote.

Yet another way to create 3D graphics is to use Sketch.

Chapter 8

Programming

8.1 Macros

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, we 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.

LaTeX is a fairly high-level language compared to Plain TeX and thus is more limited. The next chapter will focus on Plain TeX and will explain advanced techniques for programming.

8.1.1 New commands

To add your own commands, use the

command. Basically, the command requires two arguments: the *name* of the command you want to create, and the *definition* of the command. Note that the command *name* can but need not be enclosed in braces, as you like. 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.

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.

Name your new command \wbalTwo and not \wbal2 as

digits cannot be used to name macros — invalid characters will error out at compile-time.

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 the new command.

With LaTex2e, it is also possible to add a default parameter to a command with the following syntax:

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.

Note When the command is used with an explicit first parameter it is given enclosed with brackets (here "[lots of users]").

Here is a common 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 (*a*); another common version is to write it bold (**a**). Let us assume you want to write your vectors with an arrow over them; then add the following line in your mystyle.sty.

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 and increase the consistency of your document.

DeclareRobustCommand

Some commands are *fragile*, that is they fail in some environments. If a macro works in body text but not in (for example) a figure caption, it's worth trying to replace the \newcommand{\MyCommand}... declaration with \DeclareRobustCommand{\MyCommand}... in the preamble. This is especially true for macros which, when expanded, produce text that is written to a .aux file.

8.1.2 New environments

Just as with the \newcommand command, there is a command to create your own environments. The \newenvironment command uses the following syntax:

Again \newenvironment can have an optional argument. When the \begin{name} command (which starts the environment) is encountered, 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 (which ends the environment) 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 environment, 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:

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.

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:

or

8.1.3 Declare commands within new environment

New commands can be declared within newenvironment. Commands declared within the newenvironment refer to their arguments by doubling the # character. In the following example, a new environment is declared along with a nested command:

If, by mistake, the arguments passed to the \topics macro are defined with a single # character, the following error message will be thrown:

! Illegal parameter number in definition of \topics.

8.1.4 Extending the number of arguments

The xkeyval packages will let you define key/value options for commands.

The package is quite complete and documentation is exhaustive. We recommend that package developers read it.

Let's provide a simple example[1]:

8.1.5 Arithmetic

LaTeX can manipulate numbers.

The calc package provides the common infix notation.

For high-precision computations, you can use the fp[2] package.

8.1.6 Conditionals

LaTeX can use conditionals thanks to the ifthen package.

8.1.7 Loops

The PGF/TikZ extension provides the \foreach command.

If you are only using \foreach and not drawing graphics, you may instead use the pgffor package directly.

Alternatively you can check out the multido package.

8.1.8 Strings

xstring provides a lot of features. From CTAN:

- testing a string's contents
- extracting substrings
- substitution of substrings
- string length

- position of a substring

- number of recurrences of a substring

Examples:

8.1.9 LaTeX Hooks

LaTeX provide two hooks:

- \AtBeginDocument will let you specify a set of commands that will be executed when \begin{document} is met.

- \AtEndDocument does the same for \end{document}.

This gives you some more flexiblity for macros. It can be useful to override settings that get executed after the preamble. These hooks can be called several times. The commands will be executed in the order they were set.

For instance, let's replace the page numbers with old-stylenums:

There are also hooks for classes and packages. See Creating Packages.

8.1.10 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:

Now you can call LaTeX like this:

latex '\providecommand{\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.

8.1.11 Notes and References

[1] tex.stackexchange.com

[2] ctan.mackichan.com

8.2 Plain TeX

While you play with LaTeX macros, you will notice that it is quite limited. You may wonder how all these packages you are using every day have been implemented with so little. In fact, LaTeX is a set of Plain TeX macros and

most packages use Plain TeX code. Plain TeX is much more low-level, it has much more capabilities at the cost of a steep learning curve and complex programming.

Up to a few exceptions, you can use the full Plain TeX language within a valid LaTeX document whereas the opposite is false.

8.2.1 Vocabulary

To avoid confusion it seems necessary to explain some terms.

- A *group* is everything after an opening brace and before the matching closing brace.

- A *token* is a character, a control sequence, or a group.

- A *control sequence* is anything that begins with a \. It is not printed as is, it is expanded by the TeX engine according to its type.

- A *command* (or *function* or *macro*) is a control sequence that may expand to text, to (re)definition of control sequences, etc.

- A *primitive* is a command that is hard coded in the TeX engine, *i.e.* it is not written in Plain TeX.

- A *register* is the TeX way to handle variables. They are limited in numbers (256 for each type of register in classic TeX, 32767 in e-TeX).

- A *length* is a control sequence that contains a length (a number followed by a unit). See Lengths.

- A *font* is a control sequence that refers to a font file. See Fonts.

- A *box* is an object that is made for printing. Anything that ends on the paper is a box: letters, paragraphs, pages... See Boxes.

- A *glue* is a certain amount of space that is put between boxes when they are being concatenated.

- A *counter* is a register containing a number. See Counters.

There may be more terms, but we hope that it will do it for now.

8.2.2 Catcodes

In TeX some characters have a special meaning that is not to print the associated glyph. For example, \ is used to introduce a control sequence, and will not print a backslash by default.

To distinguish between different meanings of the characters, TeX split them into *category codes*, or *catcodes* for short. There are 16 category codes in TeX.

A powerful feature of TeX is its ability to redefine the language itself, since there is a \catcode function that will let you change the category code of any characters.

However, this is not recommended, as it can make code difficult to read. Should you redefine any catcode in a class or in a style file, make sure to revert it back at the end of your file.

If you redefine catcodes in your document, make sure to do it after the preamble to prevent clashes with package loading.

Active characters

Active characters resemble macros: they are single characters that will expand before any other command.

Note that an active character needs to be directly followed by a definition, otherwise the compilation will fail.

Examples

Texinfo

Texinfo uses a syntax similar to TeX with one major difference: all functions are introduced with a @ instead of a \. This is not by chance: it actually uses TeX to print the PDF version of the files. What it basically does is inputting texinfo.tex which redefines the control sequence character. Possible implementation:

With this redefinition, the '@' should now introduce every command, while the '\' will actually print a backslash character.

Itemize

Some may find the LaTeX syntax of list environments a bit cumbersome. Here is a quick way to define a wiki-like itemize:

Dollar and math

If you have many 'dollar' symbols to print, you may be better off to change the math shift character.

makeatletter and *makeatother*

If you have done a bit of LaTeX hacking, you must have encountered those two commands, \makeatletter and \makeatother.

In TeX the '@' characters belongs to catcode 11 *letters* by default. It means you can use it for macro names. LaTeX

makes use of the catcode to specify a rule: all non-public, internal macros that are not supposed to be accessed by the end-user contains at least one '@' character in their name. In the document, LaTeX changes the catcode of '@' to 12, *others*.

That's why when you need to access LaTeX internals, you must enclose all the commands accessing private functions with \makeatletter and \makeatother. All they do is just changing the catcode:

8.2.3   Plain TeX macros

\newcommand and \renewcommand are LaTeX-specific control sequences. They check that no existing command gets shadowed by the new definition.

In Plain TeX, the primitives for macro definition make no check on possible shadowing. It's up to you to make sure you are not breaking anything.

The syntax is

You can use (almost) any sequence of character between arguments. For instance let's write a simple macro that will convert the decimal separator from point to comma. First try:

This will print *(123,4)56*. We added the parentheses just to highlight the issue here. Each parameter is the shortest possible input sequence that matches the macro definition, separators included. Thus #1 matches all characters up to the first point, and #2 matches the first token only, *i.e.* the first character, since there is no separator after it.

Solution: add a second separator. A space may seem convenient:

As a general rule, everytime you expect several parameters with specific separators, think out the last separator. If you do not want to play with separators, then Plain TeX macros are used just as LaTeX macros (without default parameter):

Expanded definitions

TeX has another definition command: \edef, which stands for *expanded def*. The syntax remains the same:

The content gets expanded (but not executed, *i.e.* printed) at the point where \edef is used, instead of where the defined macro is used. Macro expansion is not always obvious...

Example:

Here the redefinition of \intro will have no effect on \example.

Global definitions

Definitions are limited to their scope. However it might be convenient sometimes to define a macro inside a group that remain valid outside the group, and until the end of the document. This is what we call *global definitions*.

You can also use the \global command with \edef.

Both commands have a shortcut:

- \gdef for \global\def

- \xdef for \global\edef

Long definitions

The previous definition commands would not allow you to use them over multiple paragraphs, *i.e.* text containing the \par command -- or double line breaks.

You can prefix the definition with the \long command to allow multi-paragraph arguments.

Example:

Outer definitions

This prefix macro prevent definitions from being used in some context. It is useful to consolidate macros and make them less error-prone because of bad contexts. *Outer macros* are meant to be used outside of any context, hence the name.

For instance the following code will fail:

Outer macros are not allowed to appear in:

- macro parameters

- skipped conditional

- ...

let and *futurelet*

\let<csname><token> is the same as \expandafter\def\expandafter<csname>\expandafter{<content>}. It defines a new control sequence name which is equivalent to the specified token. The token is usually another control sequence.

Note that \let will expand the token one time only, contrary to \edef which will expand recursively until no further expansion is possible.

Example[1]:

\futurelet<csname><token1><token2>... works a bit differently. token2 is assigned to csname; after that TeX processes the <token1><token2>... sequence. So \futurelet allows you to assign a token while using it right after.

Special control sequence name

Some macros may have a name that is not directly writable as is. This is the case of macros whose name is made up of macro names. Example:

The last line will print a sentence depending on the \status.

This command actually does the opposite of \string which prints a control sequence name without expanding it:

Controlling expansion

\expandafter{token1}{token2} will expand token2 before token1. It is sometimes needed when token2 expansion is desired but cannot happen because of token1.

\noexpand is useful to have fine grained control over what gets expanded in an \edef. Example:

\the control sequence will let you see the content of various TeX types:

- catcodes

- chardef

- font parameters

- internal parameters

- lengths

- registers

- ...

Example:

8.2.4   Registers

Registers are kind of typed variables. They are limited in numbers, ranging from 0 to 255. There are 6 different types:

TeX uses some registers internally, so you would be better off not using them. List of reserved registers:

- \box255 is used for the contents of a page

- \count0-\count9 are used for page numbering

Scratch registers (freely available):

- \box0-\box254

- \count255

- \dimen0-\dimen9

- \muskip0-\muskip9

- \skip0-\skip9

Assign register using the '=' control character. For box registers, use the \setbox command instead.

You may use one of the following reservation macro to prevent any clash:

These macros use the following syntax: \new*<csname>. Example:

These commands can not be used inside macros, otherwise every call to the macro would reserve another register.

You can print a register using the \the command. For counters use the \number command instead. For boxes use the \box command.

8.2.5   Arithmetic

The arithmetic capabilities of TeX are very limited, although this base suffice to extend it to some interesting features. The three main functions:

register may be of type count, dimen, muskip or skip. It does not make sense for box nor toks.

8.2.6   Conditionals

The base syntax is

where \if* is one command among the following.

Example:

Self defined conditionals

You can create new conditionals (as a kind of *boolean variables*) with the \newif command. With this self defined conditionals you can control the output of your code in an elegant way. The best way to illustrate the use of conditionals is through an example.

Two versions of a document must be generated. One version for group A the other one for the rest of people (i.e. not belonging to group A):

1. We use \newif to define our conditional (i.e. boolean variable).

2. In the following way we set a value (true or false) for our conditional

that is:

depending on which value we want to set in our conditional.

3. Now we can use our conditional anywhere after in an *if control structure*.

A full example is:

Case statement

The   syntax   is   \ifcase   <number><case0>\or<case1>\or...\else<defaultcase>\fi.
If number is equal to the case number, its content will be printed. Note that it starts at 0.

\else is used to specify the default case (whenever none of the previous cases have matched).

8.2.7   Loops

The base syntax is

As always, content and true action are arbitrary TeX contents. \if* refers to any of the conditionals. Note that there is no false action, you cannot put an \else between \if* and \repeat. In some case this will be the opposite of what you want; you have to change the condition or to define a new conditional using \newif. Example:

The above code will print TeX ten times.

8.2.8   Doing nothing

Sometimes it may be useful to tell TeX that you want to do nothing. There is two commands for that: \relax and \empty.

Classic example:

The \relax prevents undesired behaviour if a plus or a minus is encounter after the command.

The difference between \empty and \relax lies in the expansion: \empty disappears after macro expansion.

8.2.9   TeX characters

char

We can print all characters using the \char {charcode} command. The charcode is actually the byte value. For example

Most characters correspond to the ASCII value (e.g. A-Za-z), some replace the non-printable characters from ASCII.

chardef and *mathchardef*

You can define control sequence to expand to a specific char.   The syntax is \chardef<control sequence>=<charcode>. The following sequences do the same thing.

Example:

Font encoding map

We can use the above primitive to print the font encoding map.

Another version, with different fonts, one entry per line:

8.2.10   Verbatim lines and spaces

It is rather confusing to discover (La)TeX treats all whitespace as the same type of spacing glue. Plain TeX provides some commands to preserve the spacing and newlines as you wrote it:

which means that you will probably need to combine your own verbatim environment, and your command:

and then in your tex file:

8.2.11   Macros defining macros

This is useful in some case, for example to define language commands as explained in Multilingual versions, where the end user can write

and make sure it switches to the appropriate Babel language.

Let's define a macros that will define language commands for instance. These commands are simple: if the argument is the value of the \locale variable, then the corresponding macro prints its content directly. Otherwise, it does nothing.

Basicly, what we want to do is extremely simple: define a bunch of macros like this:

In the previous snippet of code, only the \de command in going to output its content, \en and \fr will print nothing at all. That's what we want. The problem arises when you want to automate the task, or if you have a lot of languages, and you want to change the language selection. You just have to move the #1, but that's not convenient and it makes it impossible to choose the Babel language from command line. Think this out...

What we are going to do is to define the language commands dynamically following the value of the \locale variable (or any variable of your choice). Hence the use of the \equal command from the ifthen package.

Since it is hardly possible to write it in LaTeX, we will use some Plain TeX.

Another problem arises: how to define a command whose name is a variable? In most programming languages that's not possible at all. What we could be tempted to write is

It will fail for two reasons.

1. The two last '#1' are supposed to refer to the arguments of the *new* macro, but they get expanded to

the \localedef macro first argument because they are in the body of that macro.

2. \#1 gets expanded to two tokens: '#' and '1', and the \def command will fail as it requires a valid control sequence name.

The solution to problem 1 is simple: use '##1', which will expand to '#1' when the macro is executed.

For problem 2, it is a little bit tricky. It is possible to tell tex that a specific token is a control sequence. This is what the \csname...\endcsname is used for. However

will fail because it will redefine \csname to '#1', which is not what we want, then tex will encounter \endcsname, which will result in an error.

We need to delay the expansion of \def, *i.e.* to tell tex to expand the \csname stuff first, then to apply \def on it. There is a command for that: \expandafter{token1}{token2}. It will expand {token2} before {token1}.

Finally if we want to set language from command line, we must be able to set the \locale variable so that the one in the source code is the default value that can be overridden by the one in the command line. This can be done with \providecommand:

The final code is

And you can compile with

latex '\providecommand\locale{en}\input{mydocument.tex}'

8.2.12   Notes and References

[1] From tex.stackexchange.com: What is the difference between \let and \edef?

Further reading

- The TeXbook, *Donald Knuth*

- TeX by Topic, *Victor Eijkhout*

- TeX for the Impatient, *Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves*

8.3   Creating Packages

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 or class containing all your command and environment definitions. It can be made dynamic enough to fit to all your future documents.

Classes are .cls files, packages are stored in .sty files. They are very similar, the main difference being that you can load only one class per document.

After deciding to create an own package or class, you should think about which license the package/class has. A license is of great importance, either to protect your file, or to make it available for others.

8.3.1   makeatletter and makeatother

By default, LaTeX will allow the use of the '@' characters for control sequences from within package and class files, but not from within an end-user document. This way it is possible to *protect* commands, *i.e.* to make them accessible from packages only.

However it is possible to override this security with the duo \makeatletter and \makeatother. These commands only make sense in a regular document, they are not needed in package or class files.

8.3.2   Creating your own package

Your package can be made available in your document just like any other package: using the \usepackage command. Writing a package basically consists of copying the contents of your document preamble into a separate file with a name ending in .sty.

Let's write a first custom.sty file as an example package:

- \NeedsTeXFormat{...} specifies which version of TeX or LaTeX is required at least to run your package. The optional date may be used to specify the version more precisely.

- \ProvidesPackage{<name>}[<version>] A package introduces itself using this command. *<name>* should be identical to the *basename* of the file itself. The *<version>* should begin with a date in the format *YYYY/MM/DD*. Version information should be kept updated while developing a package.

- Next you may write some TeX or LaTeX code like loading package, but write only the bare minimum needed for the package options set below.

- \RequirePackage is equivalent to \usepackage.

- \DeclareOptions are end-user parameters. Each option is declared by one such command.

- \ExecuteOptions{...} tells which are the default.

- \ProcessOptions\relax terminates the option processing.

- Write whatever you want in it using all the LaTeX commands you know. Normally you should define new commands or import other packages.

- \endinput: this *must* be the last command.

Once your package is ready, we can use it in any document. Import your new package with the known command \usepackage{mypack}. The file custom.sty and the LaTeX source you are compiling must be in the same directory.

For a more convenient use, it is possible to place the package within $TEXMFHOME (which is ~/texmf by default) according to the TeX Directory Structure (TDS). That would be

$TEXMFHOME/tex/latex/custom/custom.sty

On Windows '~' is often C:\Users\username.

You may have to run texhash (or equivalent) to make your TeX distribution index the new file, thus making it available for use for any document. It will allow you to use your package as detailed above, but without it needing to be in the same directory as your document.

8.3.3   Creating your own class

It is also possible to create your own class 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:

The name of the class file is then myclass.cls. Let's write a simple example:

- \ProvidesClass is the counterpart of \ProvidesPackage.

- \PassOptionsToClass and \PassOptionsToPackage are used to automatically invoke the corresponding options when the class or the package is loaded.

- \DeclareOption*: the starred version lets you handle non-implemented options.

- \ClassWarning will show the corresponding message in the TeX compiler output.

- \LoadClass specifies the unique parent class, if any.

8.3.4   Hooks

There are also hooks for classes and packages.

- \AtEndOfPackage

- \AtEndOfClass

They behave as the document hooks. See LaTeX Hooks.

8.4   Themes

Newcomers to LaTeX often feel disappointed by the lack of visual customization offered by the system. Actually

this is done on purpose: the LaTeX philosophy takes a point at doing the formatting while the writer focuses on the content.

In this chapter, we will show what we can achieve with some efforts.

8.4.1 Introduction

In the following we will write the theme, a package that will only change the appearance of the document, so that our document will work with or without the theme.

Note that if it may look eye-candy, this is absolutely not a model of typography. You should not use such theme for serious publications. This is more a technogical example to exhibit LaTeX capabilities.

- Custom theme (TOC)

- Custom theme

- Custom theme (red)

8.4.2 Package configuration

Nothing much to say here. This is a direct application of the Creating Packages chapter.

We load the required packages.

- needspace is used to prevent page break right after a sectioning command.

- tikz is used to draw the fancy material.

We define a color option, you may use as much as you want. Defining colors with specific names makes it very flexible. We also use an option to toggle the fancy reflection effect which might be a little *too much*!

8.4.3 Header and footer

We use TikZ to draw a filled semicircle.

fancyhdr is used to set header and footer. We take care of using the fancy style and to start from scratch by erasing the previous header and footer with \fancyhf{}.

8.4.4 Table of contents

We redefine commands used by \tableofcontents.

8.4.5 Sectioning

This is definitely the most complex part. It is not that hard since the code is almost the same for \section, \subsection and \subsubsection.

We use \needspace to make sure there is no line break right after a sectioning command. We enclose the command in a group where we set a font size since the space we need is \baselineskip which depends on the font size.

Starred commands will not set the counters (LaTeX detault behaviour). You can choose to handle starred command differently by resetting the counters for instance.

We preceed the section printing by a \noindent. We make sure to end the section printing by a \par command to make sure following text gets printed properly.

For \subsection we make use of the mirrors option to change the appearance accordingly.

To handle the PDF bookmarks properly we need the following lines at the end of the definitions.

Finally, for \section only, we want it to print in the header, so we call the \sectionmark command. Here we changed the behaviour of the starred command over the original LaTeX version, since we define and use the \sectionmarkstar command.

8.4.6 Notes and References

Chapter 9

Miscellaneous

9.1 Modular Documents

During this guide we have seen what 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 is all about organizing your files using the modular capabilities of LaTeX. 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.

9.1.1 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.

9.1.2 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}

Comparing the methods: input vs include

The differences between these two ways to include files is explained below. What they have in common is that they process the contents of filename.tex before continuing with the rest of the base file (the file that contains these statements). When the compiler processes your base file and reaches one of the commands \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 write the contents using \input or \include in 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. This 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, whereas \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!

Another 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 use within, say, a long article with discrete sections, which of course are not normally set on a new page).

In either case the .tex filename extension is optional.

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 rewriting 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.

When using relative paths within a LaTeX file imported by \input or \include, it is important to note that the paths are relative to the directory in which the main .tex file resides, not to the directory in which the included (or input) file is found. This is likely to be an issue if using a folder per chapter, with the figures in each chapter's folder, and using \include to read the chapter source into a main LaTeX file in a parent folder.

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}   %\in-

put{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 rerun 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. If you are using absolute or relative paths to the files, type in the complete reference.

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}   % compile   just chapters 1 and 4, space characters not permitted \begin{document} \include{Chapter_1} % omit the '.tex' extension   \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.

Separate compilation of child documents

A disadvantage of solely using \input and \include is that only the base document can be compiled. However, you may decide that you work better on individual sections of text and wish to edit and compile those separate from the main file. There are a few packages available to address this problem.

Subfiles   The subfiles package provides a way to compile sections of a document using the same preamble as the main document.

In the main 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[main.tex]{subfiles} \begin{document}

and end with:

\end{document}

It is possible to add parts that will only be applied if the child document is compiled by its own, by defining an "identity" command \newcommand{\onlyinsubfile}[1]{#1} in the main document and then overwriting it after \begin{document} using \renewcommand{\onlyinsubfile}[1]{}. Similarly, the same can be done for parts to appear only if compiled by the main document.

In summary, the base document (main.tex) looks like:

\documentclass{book}   \usepackage{subfiles} \newcommand{\onlyinsubfile}[1]{#1}   \newcommand{\notinsubfile}[1]{}   \begin{document}   \renewcommand{\onlyinsubfile}[1]{}   \renewcommand{\notinsubfile}[1]{#1} %% 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 \onlyinsubfile{this only appears if chapter1.tex is compiled (not when main.tex is compiled)} \notinsubfile{this only appears if main.tex is compiled (not when chapter1.tex is compiled)} %% more of my chapter 1 content %% \end{document}

Some linux distributions don't have subfiles package in their latex distributions, since it was not included until TeXLive 2012. You can download subfiles.tds.zip from CTAN. This package will contain two files subfiles.cls and subfiles.sty. Move these files to a directory under the name subfiles in the path /usr/share/texmf/tex/latex. This still won't make the package available; the texhash program must be executed first. Now you are good to go!

Standalone   The standalone package is designed for moving more of the opposite direction than subfiles. It provides a means for importing the preamble of child documents into the main document, allowing for a flexi-

ble way to include text or images in multiple documents (e.g. an article and a presentation).

In the main document, the package must be loaded as:

\usepackage{standalone}

Child documents are loaded using \input or \include.

The child documents contain, for example, the following statements:

\documentclass{standalone} % Load any packages needed for this document \begin{document} % Your document or picture \end{document}

In summary, the base document (main.tex) looks like:

\documentclass{book} \usepackage{standalone} \begin{document} %% my document content \input{chapter1} %% more of my document content \end{document}

and Chapter 1 (chapter1.tex) looks like:

\documentclass{standalone} % Preamble \begin{document} %% my chapter 1 content %% %% more of my chapter 1 content \end{document}

Import The import package allows for relative directories. While subfiles fails to have a way of a subfile itself having references relative to its own directory, the \subimport command provides this functionality.

9.1.3 Including complete pdf documents or single pages

There is a great package for including full pages of PDF files: pdfpages. It is capable of inserting entire pages as is and more pages per one page in any layout (e.g. 2×3).

The package has several options:

Options:

- **final**: Inserts pages. This is the default.

- **draft**: Does not insert pages, but prints a box and the filename instead.

- **enable-survey**: Activates survey functionalities. (Experimental, subject to change.)

The first command is

Options for **key=val** (A comma separated list of options using the key = value syntax)

You can also insert pages of several external PDF documents.

Several PDFs can be placed table-like on one page. See more information in its documentation.

Produce a single .tex from modular documents

If you need to produce a single tex file from a modular document, several scripts are available

- a perl script latexpand

- a C script flatex

- a python version of the script flatex

- another C script flatten

9.1.4 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 main.tex 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 of the mystyle.sty file:

\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.

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.

This is actually the beginning of the process of writing a package. See LaTeX/Macros for more details.

For a list of several packages you can use, see the List of Packages section.

9.1.5 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 look like this (it's the sample code for a *report*, but you might easily change it to *article* or whatever else):

\documentclass[12pt,a4paper]{report}   \usepackage{graphicx} % put all the other packages here: \usepackage{mystyle}   \usepackage{hyperref}   \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 \input{./tex/mybibliography.tex} \end{document}

Here a lot of code expressed in previous sections has been used. 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 is 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 separate file.

Once you have 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 into 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 give your files names 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.

9.1.6   External Links

- Subfiles package documentation

- Standalone package documentation

- pdfpages package documentation

9.2   Collaborative Writing of La-TeX 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/).

9.2.1   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

- You can use one of the online solutions listed in the Installation chapter. Most of them have collaboration features.

- Another option for collaboration is dropbox. It has 2 GB free storage and versioning system. Works like SVN, but more automated and therefore especially useful for beginning LaTeX users. However, Dropbox is not a true versioning control system, and as such it does not allow you to roll the article back to previous versions.

- You can use an online collaborative tool built on top of a versioning control system, such as Authorea or ShareLatex. Authorea performs most of the actions described in this document, but in the background (it is built on Git). It allows authors to enter LaTeX or Markdown via a GUI with mathematical notation, figures, d3.js plots, IPython notebooks, data, and tables. All content is rendered to HTML5. Authorea also features a commenting system and article-based chat to ease collaboration and review.

- 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).

- TitanPad (or other clones of EtherPad). To compile use the command:

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').

- With a dedicated Linux box with LaTeX & Dropbox it's possible to use Google docs and some scripting to get automatically generated PDFs on Dropbox from updates on Google Docs.

- You can use a distributed version control system such as Mercurial or Git. This is the definitive solution for users looking for control and advanced features like branch and merge. The learning curve will be steeper than that for a web-based solution.

9.2.2 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.

9.2.3 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. 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).

9.2.4 The Version Control System *Subversion*

Subversion (SVN) 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 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 client, because it provides the most usual interface for common users. Linux users usually use SVN utilities from the command-line, or eSvn--a GUI frontend--with KDiff3 for showing complex differences.

9.2.5   Hosting LaTeX files in *Subversion*
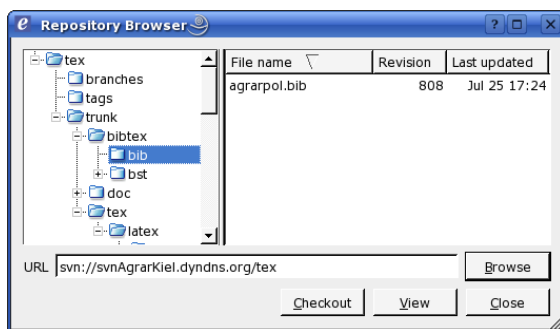
Figure 1: Common texmf tree shown in eSvn's *Repository Browser*

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 TEXMFHOME = $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.

9.2.6 *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).
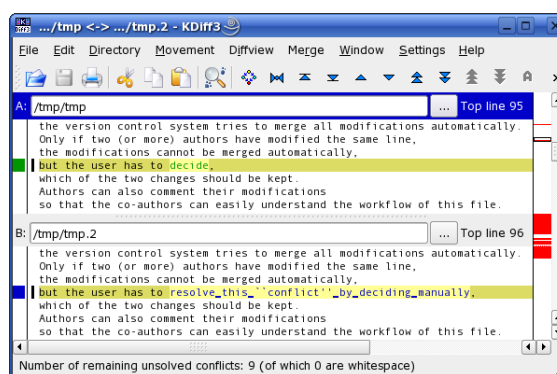
Figure 2: Reviewing modifications in KDiff3

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.

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.

Visualizing *diffs* in LaTeX: latexdiff and changebar

The tools latexdiff and changebar can visualize differences of two LaTeX files inside a generated document. This makes it easier to see impact of certain changes or discuss changes with people not custom to LaTeX. Changebar comes with a script chbar.sh which inserts a bar in the margin indicating parts that have changed. Latexdiff allows different styles of visualization. The default is that discarded text is marked as red and added text is marked as blue. It also supports a mode similar to Changebar which adds a bar in the margin. Latexdiff comes with a script latexrevise which can be used to accept or decline changes. It also has a wrapper script to support version control systems such as the discussed Subversion.

An example on how to use Latexdiff in the Terminal.

latexdiff old.tex new.tex > diff.tex # Files old.tex and new.tex are compared and the file visualizing the changes is written to diff.tex pdflatex diff.tex # Create a PDF showing the changes

The program DiffPDF can be used to compare two existing PDFs visually. There is also a command line tool comparepdf based on DiffPDF.

9.2.7 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://www.jabref.org). 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').
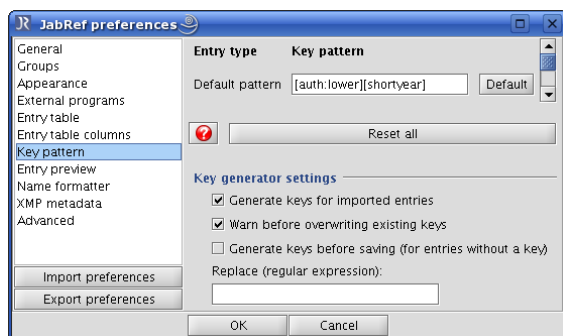
Figure 3: Specify default key pattern in JabRef

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).
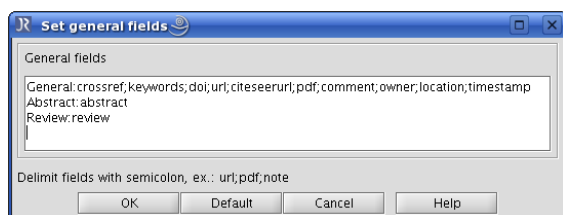
Figure 4: Set up general fields in JabRef

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).

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
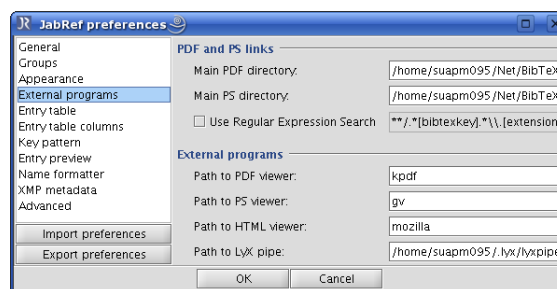
Figure 5: Specify 'Main PDF directory' in JabRef

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.

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).

9.2.8 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.

9.2.9   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.

9.2.10   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

9.3   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.

9.3.1   Tools installation

This chapter features a lot of third-party tools; most of them are installed independently of your TeX distribution.

Some tools are Unix-specific (*BSD, GNU/Linux and Mac OS X), but it may be possible to make them work on Windows. If you have the choice, it is often easier with Unix systems for command line tools.

Some tools may already be installed. For instance, you can check if dvipng is installed and ready to use (Unix only):

type dvipng

Most of these tools are installable using your package manager or portage tree (Unix only).

9.3.2   Preview mode

This section describes how to generate a screenshot of a LaTeX page or of a specific part of the page using the LaTeX package preview. 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 LibreOffice Impress. First, start by making sure you have preview. See Installing Extra Packages.

Say you want to take a screenshot of

$$\pi = \sqrt{12} \sum_{k=0}^{\infty} \frac{(-3)^{-k}}{2k+1}.$$

Write this formula in the preview environment:

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 any output.

This package is also very useful to export specific parts to other format, or to produce graphics (*e.g.* using PGF/TikZ) and then including them in other documents. You can also automate the previewing of specific environments:

This will produce a PDF containing only the listing content, the *page* layout will depend on the shape of the source code.

9.3.3 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:

Using Windows gswin32 -dNOPAUSE -sDEVICE=pdfwrite -sOUTPUTFILE=Merged.pdf -dBATCH 1.pdf 2.pdf 3.pdf

Using Linux gs -dNOPAUSE -sDEVICE=pdfwrite -sOUTPUTFILE=Merged.pdf -dBATCH 1.pdf 2.pdf 3.pdf

Alternatively, PDF-Shuffler 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 (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

Using pdfLaTeX *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.*:

Three simple shell scripts using the pdfpages package are provided in the pdfjam bundle by D. Firth. They include options to merge several pdf files (pdfjoin), put several pages in one physical sheet (pdfnup) and rotate pages (pdf90).

See also Modular Documents

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.

9.3.4 Convert to PostScript

from PDF

pdf2ps my_file.pdf

from DVI

dvi2ps my_file.dvi

9.3.5 Convert to RTF

LaTeX can be converted into an RTF file, which in turn can be opened by a word processor such as LibreOffice Writer or Microsoft Word. This conversion is done through latex2rtf, which may run on any computer platform, however is only actively supported on Windows, Linux and BSD, with the last mac update being from 2001. 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 la-tex2rtf 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 word processors such as Microsoft Word or Libre-Office.

9.3.6   Convert to HTML

There are many converters to HTML. Some of them use an intermediate file which then will be converted to the destination format.

HEVEA

hevea mylatexfile

latex2html

latex2html -html_version 4.0,latin1,unicode -split 1 -nonavigation -noinfo -title "MyDocument" MyDocument.tex

LaTeXML

latexmlc paper.tex --destination=paper.html

pdf2htmlEX

pdf2htmlEX [options] <input.pdf> [<output.html>]

pdf2htmlEX can convert PDF to HTML without losing text or format. It is designed as a general PDF to HTML converter, not only restricted to the PDF generated by La-TeX source. LaTeX users can compile the LaTeX source code to PDF, and then convert the PDF to HTML via pdf2htmlEX. Some introductions of pdf2htmlEX can be found on its own wiki page. More technical details can be found on the paper published on TUGboat: *Online publishing via pdf2htmlEX* HTML / PDF. The Figure 3 of the paper gives different work-flows of publishing HTML online.

TeX4ht

TeX4ht is a very powerful conversion program, but its configuration is not straightforward. Basically a config-uration file has to be prepared, and then the program is called. TeX4ht does not convert the source files directly but processes the .dvi file instead.

bibtex2html

For exporting the BibTeX file only.

bibtex2html mybibtexfile

9.3.7   Convert to image formats

It is sometimes useful to convert LaTeX output to image formats for use in systems that do not support DVI nor PDF files, such as Wikipedia.

There are two families of graphics:

- Vector graphics can be scaled to any size, thus do not suffer from quality loss. SVG is a vector format.

- Raster graphics define every pixel explicitly. PNG is a raster format.

So vector graphics are usually preferred. There is still some cases where raster graphics are used:

- The target system does not handle vector graphics, only raster graphics are supported.

- SVG can not embed fonts. So either the font will be rendered using a local .ttf or .otf font (which will mostly change the output), or all characters must be turned to vector graphics. This last method makes the SVG big and slow. If the input LaTeX file con-tains a lot of text which formatting must be pre-served, SVG is not that great.

So SVG is great for drawings and a small amount of text. JPG is a well known raster formats, however it is usually not as good as PNG for text.

In some cases it may be sufficient to simply copy a region of a PDF (or PS) file using the tools available in a PDF viewer (for example using LaTeX to typeset a formula for pasting into a presentation). This however will not gen-erally have sufficient resolution for whole pages or large areas.

Multiple formats

pdftocairo

There is pdftocairo featured in the poppler toolset.

pdftocairo -svg latexdoc.pdf output.svg

pdftocairo also supports various raster graphic formats.

Vector graphics

pdf2svg

Direct conversion from PDF to SVG can be done using the command line tool pdf2svg.

pdf2svg file.pdf file.svg

ps2svg

Alternatively DVI or PDF can be converted to PS as described before, then the bash script ps2svg.sh can be used (as all the software used by this script is multiplatform, this is also possible in Windows, a step-by-step guide could be written).

dvisvgm

One can also use dvisvgm, an open source utility that converts from DVI to SVG.

dvisvgm -n file.dvi

Inkscape

Inkscape is able to convert to SVG, PDF, EPS, and other vector graphic formats.

inkscape --export-area-drawing --export-ps=OUTPUT INPUT inkscape --export-area-page --export-plain-svg=OUTPUT INPUT

Raster graphics

JPEG

Run ghostscript on the PostScript file created by pdf2ps as follows:

echo "quit" | gs -sDEVICE=jpeg -sOutputFile=document.jpg -r300 document.ps

GIMP

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.

dvipng

A method for DVI files is dvipng. Usage is the same as dvipdfm.

Run latex as usual to generate the dvi file. 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.

ImageMagick

The convert command from the ImageMagick suite can convert both DVI and PDF files to PNG.

convert input.pdf output.png

optipng

You can optimize the resulting image using optipng so that it will take up less space.

9.3.8 Convert to plain text

If you are thinking of converting to plain text for spell-checking or to count words, there may be an easier way — 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, or use HTML conversion first and then copy text from your browser.

If you want to keep the formating, you can use a *DVI-to-plain text* converter, like catdvi. Example:

catdvi yourfile.dvi | fmt -u

The use of *fmt -u* (available on most Unices) will remove the justification.

Chapter 10

Help and Recommendations

10.1 FAQ

10.1.1 Margins are too wide

LaTeX's default margins may seem too large. In most cases, this is a preferred default and improves readability.

If you still disagree, you can easily change them with

See Page Layout.

10.1.2 Avoid excessive double line breaks in source code

Too many paragraphs of one line or two do not look very good.

Remember the TeX rule:

- If two or more consecutive line breaks are found, TeX starts a new paragraph.

- If only one linebreak is found, TeX inserts a space if there is no space directly before or after it.

You might be tempted to put blank lines all the time to improve the readability of your source code, but this will have an impact on formatting. The solution is simple: put a comment at the very beginning of the blank lines. This will prevent TeX from seeing another line break— all characters up to and including the next line break after a comment are ignored.

Example:

10.1.3 Simplified special character input

So long as your computing environment supports UTF-8, you can enter special characters directly rather than entering the TeX commands for diacritics and other extended characters. E.g.,

This requires that:

- your text editor supports and is set to save your file in UTF-8;

- you add the \usepackage[utf8]{inputenc} line in the preamble.

Avoid using latin1. See Special Characters.

10.1.4 Writing the euro symbol directly

Add the following lines in your preamble:

10.1.5 LaTeX paragraph headings have title and content on the same line

Some people do not like the way \paragraph{...} writes the title on the same line as the content. This is actually fairly common in a lot of documents and not as weird as it may seem at first.

There are ways to get around the default behavior, however; see \paragraph line break for more information.

10.1.6 *Fonts are ugly/jagged/bitmaps* **or** *PDF search fails* **or** *Copy/paste from PDF is messy*

You must be using diacritics (*e.g.* accents) with OT1 encoding (the default). Switch to T1 encoding:

If you have ugly jagged fonts after the font encoding change, then you have no Type1 compatible fonts available. Install Computer Modern Super or Latin Modern (package name may be lm). To use Latin Modern you need to include the package:

See Fonts for an explanation.

10.1.7 Manual formatting: use of line breaks and page breaks

You should *really* avoid breaking lines and pages manually. The TeX engine is in charge of that. The big problem with manual formatting is that it is not dynamic. Even

if it looks right the first time, the content is likely to render really badly if you change anything before the point you manually formatted.

The only place where page breaks are recommended is at the upper level of sectioning in your documents, *e.g.* parts or chapters (although when you start a new part or chapter, LaTeX will ordinarily do this for you). When you do manually insert a page break, you should use \clearpage or \cleardoublepage which print currently floating figures before starting a new page.

If you absolutely have to insert line or page breaks manually, you should do it after you are sure you have completed your document otherwise, so that you don't later have to come back and update it.

10.1.8 Always finish commands with *{}*

TeX has an unintuitive rule that if a control sequence (a command) is not followed by a pair of braces (with a parameter in between or not), then the following space character(s) are ignored. LaTeX will not print *any* space, and the command (say, the TeX or LaTeX logos) are run together with the following word.

To fix this, use a pair of braces after the command, even if there are no parameters. Example:

(Technical explanation: a control sequence name can only be composed of characters with catcode 11, that is A-Z and a-z by default. TeX knows where the control sequence name start thanks to the backslash, and it knows where it ends when it encounters the first token which is not of catcode 11. This character is then skipped. Since consecutive spaces have been concatenated into one single space, no space is taken into account.)

It is possible to define macros that will insert a space dynamically by using the xspace package.

- If there is no brace and a space following the command, an extra space will be appended.

- If there are braces, no extra space will be printed.

Example:

10.1.9 Avoid bold and underline

Typographically speaking, it is usually poor practice to use bold or underline formats in the middle of a paragraph. This has become a common habit for users of traditional word processors because these two functions are very easily accessible (along with italics).

However, bold and underline tend to overweight the text and distract the reader. When you start reading a paragraph with a bold word in the middle, you often read the emphasized part first, thus spoiling the content and breaking the order of the ideas. Italics are less obvious and do not have more weight than normal characters, so they are usually a better choice for emphasizing small amounts of text.

The original and more appropriate use of bold and underline is for special parts, such as headers, the index, glossaries, and so on. (Actually, underlining is rarely used in professional environments.)

LaTeX has a macro \emph{...} for emphasizing text using italics. It should be preferred to \textit{...} because \emph{...} will correctly print emphasized text inside other italic text in the regular font.

10.1.10 The proper way to use figures

Users used to WYSIWYG document processors like Microsoft Word or LibreOffice often get frustrated with figures. The answer is simple: a figure is *not* a picture!

If you use \includegraphics without enclosing it in a figure environment, it will behave just as in a word processor, placing the picture right at the spot where it was placed in the source.

Figures are a type of float, which is a virtual object that LaTeX can put in places other than the exact location it was created, which helps to prevent cluttering your text with pictures and tables.

See Importing Graphics and Floats, Figures and Captions for more details.

10.1.11 Text stops justifying

Most likely you have used \raggedleft, \raggedright or \centering at some point and forgotten to switch it off. These commands are switches—they remain active until the end of the scope, or until the end of the document if there is no scope. See Paragraph Alignment for more information.

10.1.12 Rules of punctuation and spacing

LaTeX does some work for you, but not everything. Especially regarding punctuation, you are pretty free to do what you want. Punctuation rules are different for each language. In English there is *no* space before a punctuation mark and one space after it.

There are a lot of rules, but you can have a quick look at Wikipedia.

10.1.13 Compilation fails after a Babel language change

This is a limitation of Babel. Delete the .aux file (or clean the project), then try compiling again.

10.1.14   Learning LaTeX quickly or correctly

Nowadays it is very common to "learn" on the web by using a search engine and copying and pasting things here and there. As with every programming language, this is generally a poor method which will lead to lack of control, unexpected results, and a lot of frustration. Really learning LaTeX is not that difficult and does not take that long. Most chapters in this book are dedicated to a specific usage, so the basics are actually covered very quickly.

If you are getting frustrated with a specific package, make sure you read its official documentation, which is usually the best source of information. Content found on the web, even in this book, is rarely as accurate as the official documentation. Inaccurate information might result in causing mistakes without you understanding why.

The time you spend learning is worth it, and it quickly makes up for the time you would lose if you don't learn things properly and end up stuck all the time.

10.1.15   Non-breaking spaces

This useful feature is unknown to most newcomers, although it is available on most WYSIWYG document processors. A non-breaking space between two tokens (e.g. words, punctuation marks) prevents processors from inserting a line break between them. It is very important for consistent reading. LaTeX uses the '~' symbol as a non-breaking space.

You usually use non-breaking spaces for punctuation marks in some languages, for units and currencies, for initials, etc.

For example, in French typography, you put a non-breaking space before all two-parts punctuation marks. Example:

Note that writing French like this might get really painful. Thankfully, Babel with the frenchb option will take care of the non-breaking spaces for all punctuation marks. In the above example, only the non-breaking space for the euro symbol must remain.

10.1.16   Smart mathematics

All virtual objects designated by letters, variables or others should use a dedicated formatting. For math and a lot of other fields, the LaTeX math formatting is perfect. For instance, if you want to refer to an object A, write

If you want to refer to several objects in a sentence, it is the same.

If you refer to a set of objects, you can still use the math notation.

Note that this is different from usual text parentheses.

10.1.17   Use vector graphics rather than raster images

Raster (bitmap) graphics scale poorly and often create jagged or low-quality results which clash with the document quality, particularly when printed.

Using vector (line-oriented) graphics instead, either through LaTeX's native diagramming tools or by exporting vector formats from your drawing or diagramming tools, will produce much higher quality results. When possible, you should prefer PDF, EPS, or SVG graphics over PNG or JPG.

10.1.18   Stretching tables

Trying to stretch tables with the default tabular environment will often lead to unexpected results. The nice tabu package will do what you want and even much more. Alternatively if you cannot use the tabu package you may try tabularx or tabulary packages See Tables.

10.1.19   Tables are easier than you think

Even though the Tables chapter is quite long, it is worth reading. In the end, you only need to know a few things about the environment of your choice.

Some LaTeX editors feature table assistants. Also, many spreadsheet applications have a LaTeX export feature (or plugin). Again, see Tables for more details.

10.1.20   Relieving cumbersome code (lists and long command names)

LaTeX is sometimes cumbersome to write, especially if you are not using an adequate editor. See Editors for some interesting choices.

You can define aliases to shorten some commands:

Here the xspace package comes in handy to avoid swallowed spaces.

For lists you may want to try the easylist package. Now writing a list is as simple as

10.1.21   Reducing the size of your LaTeX installation

The Installation article explains in detail how to manually install a fully functional TeX environment, including LaTeX and other features, in under 100 MB.

10.2 Tips and Tricks

10.2.1 Always writing LaTeX in roman

If you insert the \LaTeX command in an area with a non-default font, it will be formatted accordingly. If you want to keep LaTeX written in Computer Modern roman shape, you must redefine the function. However, the naive

\renewcommand{\LaTeX}{{\rm \LaTeX}}

will output:

TeX capacity exceeded , sorry [grouping levels =255].

So you need to create a temporary variable.

Sadly,

\newcommand{\LaTeXtemp}{\LaTeX} \renewcommand{\LaTeX}{{\rm \LaTeXtemp}}

does not work as well.

We must use the TeX primitive \let instead.

\let\LaTeXtemp\LaTeX \renewcommand{\LaTeX}{{\rm \LaTeXtemp }}

10.2.2 *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.

10.2.3 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".

10.2.4 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 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.

LaTeXDraw is a free and open source graphical PSTricks generator and editor. It allows you to draw basic geometric objects and save the result in a variety of formats including .jpg, .png, .eps, .bmp as well as .tex. In the last case the saved file contains PSTricks/LaTeX code only. Owing to that you can include any possible LaTeX code in the picture, since the file is rendered by your LaTeX environment directly.

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:
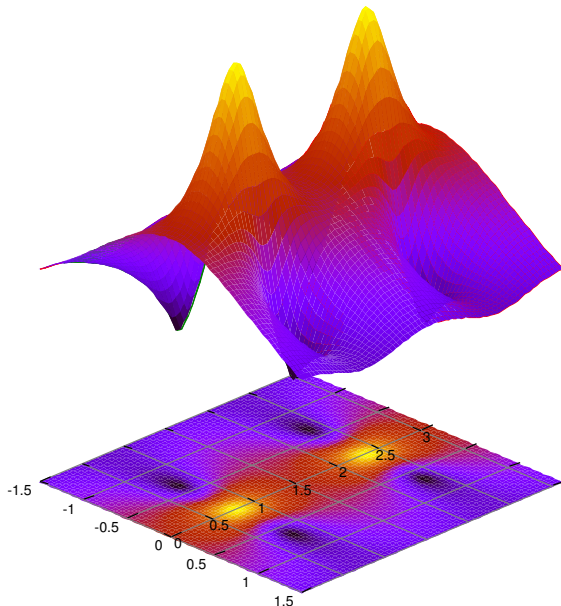
gnuplot can plot various numerical data, functions, error distribution as well as 3D graphs and surfaces

set format "$%g$" set title "Graph 3: Dependence of V_p on R_0" set xlabel "Resistance R_0 [Ω]" 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 \be-

gin{gnuplot}\end{gnuplot}, and compiling with latex -shell-escape, the graphs are created and added into your document.

Failure to access gnuplot from latex for Windows can be solved by making file title only in one word. Don't type **my report.tex** for your title file, but do **myreport.tex** .

When you are using gnuplottex it is also possible to directly pass the terminal settings as an argument to the environment

\begin{gnuplot}[terminal=epslatex, terminaloptions=color, scale=0.9, linewidth=2] ... \end{gnuplot}

Using gnuplottex can cause fraudulent text-highlighting in some editors when using algebraic functions on imported data, such as:

(2*($1)):2

Some editors will think of all following text as part of a formula and highlight it as such (because of the '$' that is interpreted as part of the latex code). This can be avoided by ending with:

#$ \end{gnuplot}

As it uncomments the dollar sign for the gnuplot interpreter, but is not affecting the interpretation of the .tex by the editor.

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.

Generate png screenshots

See Export To Other Formats.

10.2.5   Spell-checking and Word Counting

If you want to spell-check your document, you can use the command-line aspell, hunspell (preferably), or ispell

programs.

ispell yourfile.tex aspell --mode=tex -c yourfile.tex hunspell -l -t -i utf-8 yourfile.tex

All three understand LaTeX and will skip LaTeX commands. You can also use a LaTeX editor with built-in spell checking, such as LyX, Kile, or Emacs. 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

10.2.6   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 % 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.

10.2.7   Sidebar with information

If you want to put a sidebar with information like copyright and author, you might want to use the eso-pic package. Example:

\usepackage{eso-pic} ... \AddToShipoutPicture{% \AtPageLowerLeft{% \rotatebox{90}{%

\begin{minipage}{\paperheight} \centering\textcopyright~\today{} Humble me \end{minipage} % } } % }%

If you want it on one page only, use the starred version of the *AddToShipoutPicture* command at the page you want it. (\AddToShipoutPicture*{...})

10.2.8   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 a Unix-based system you can do something like this:

alias pdflatex='mkdir -p tmp; pdflatex -output-directory tmp'

Or for vim modify your .vimrc:

" use pdflatex let g:Tex_DefaultTargetFormat='pdf' let g:Tex_MultipleCompileFormats='pdf,dvi' let g:Tex_CompileRule_pdf = 'mkdir -p tmp; pdflatex -output-directory tmp -interaction=nonstopmode $*; cp tmp/*.pdf .'

Chapter 11

Appendix

11.1 Authors

11.1.1 Included books

The following books have been included in this wikibook (or we are working on it!), with permission of the author:

- Andy Roberts' Getting to grips with Latex.

- Not So Short Introduction to LaTex2e by Tobias Oetiker, Hubert Partl and Irene Hyna. We have contacted the authors by email asking for permission: they allowed us to use their material, but they never edited directly this wikibook. That book is released under the GPL, that is not compatible with the GFDL used here in Wikibooks. Anyway, we have the permission of the authors to use their work. You can freely copy text from that guide to here. If you find text on both the original book and here on Wikibooks, then that text is double licensed under GPL and GFDL. For more information about Tobias Oetiker and Hubert Partl, their websites are http://it.oetiker.ch/ and http://homepage.boku.ac.at/partl/ respectively.

- LaTeX Primer from the Indian TeX Users Group. Their document is released under the *GNU Free Documentation License*, the same as Wikibooks, so we can include parts of their document as we wish. In any case, we have contacted Indian TeX Users Group and they allowed us to do it.

- David Wilkins' Getting started with LaTeX. The book is not released under any free license, but we have contacted the author asking him for the permission to use parts of his book on Wikibooks. He agreed: his work is still protected but you are allowed to copy the parts you want on this Wikibook. If you see text on both the original work and here, then that part (and only that part) is released under the terms of GFDL, like any other text here on Wikibooks.

In progress

- Peter Flynn's Formatting information, a beginner's guide to typesetting with LaTeX. We have contacted him by email asking for permission to use his work. The original book is released under the *GNU Free Documentation License*, the same as Wikibooks. For more information, his personal website is http://silmaril.ie/cgi-bin/blog.

11.1.2 Wiki users

Major contributors to the book on Wikibooks are:

- Alessio Damato
- Jtwdog
- Pierre Neidhardt

11.2 Links

Here are some other online resources available:

Community

- The TeX Users Group Includes links to free versions of (La)TeX for many kinds of computers.

- UK-TUG The UK TeX Users' Group

- TUGIndia The Indian TeX Users Group

- comp.text.tex Newsgroup for (La)TeX related questions

- CTAN hundreds of add-on packages and programs

Tutorials/FAQs

- Tobias Oetiker's Not So Short Introduction to LaTex2e:
  http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf also at
  http://web.archive.org/web/20010603070337/http://people.ee.ethz.ch/~{}oetiker/lshort/lshort.pdf

- Vel's introduction to LaTeX: What is it, why should you use it, who should use it and how to get started: http://www.vel.co.nz/vel.co.nz/Blog/Entries/2009/11/4_LaTeX_Document_Preparation_System.html

- Peter Flynn's beginner's guide (formatting): http://www.ctan.org/tex-archive/info/beginlatex/beginlatex-3.6.pdf

- The AMS Short Math Guide for LaTeX, a concise summary of math formula typesetting features http://www.ams.org/tex/amslatex.html

- amsmath users guide (PDF) and related files: http://www.ctan.org/tex-archive/macros/latex/required/amslatex/math/

- LaTeX Primer from the Indian TeX Users Group: http://sarovar.org/projects/ltxprimer/

- LaTeX Primer http://www.maths.tcd.ie/~{}dwilkins/LaTeXPrimer/

- PSTricks--fancy graphics exploiting PDF capabilities http://sarovar.org/projects/pstricks/

- PDFScreen--create LaTeX PDF files that have navigation buttons used for presentations: http://sarovar.org/projects/pdfscreen/

- David Bausum's list of TeX primitives (these are the fundamental commands used in TeX): http://www.tug.org/utilities/plain/cseq.html

- Leslie Lamport's manual for the commands that are unique to LaTeX (commands not used in plain TeX): http://www.tex.uniyar.ac.ru/doc/latex2e.pdf

- The UK TeX FAQ List of questions and answers that are frequently posted at comp.text.tex http://www.tex.ac.uk/faq

- TeX on Mac OS X: Guide to using TeX and LaTeX on a Mac http://www.rna.nl/tex.html

- Text Processing using LaTeX http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/

- The (La)TeX encyclopaedia http://tex.loria.fr/index.html

- Hypertext Help with LaTeX http://www.giss.nasa.gov/tools/latex/index.html

- EpsLatex: a very comprehensive guide to images, figures and graphics http://www.ctan.org/tex-archive/info/epslatex.pdf

- The Comprehensive LaTeX Symbol List (in PDF) http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf

- Getting to Grips with LaTeX (HTML) Collection of Latex tutorials taking you from the very basics towards more advanced topics http://www.andy-roberts.net/misc/latex/index.html

- Chapter 8 (about typesetting mathematics) of the *LaTeX companion* http://www.macrotex.net/texbooks/latexcomp-ch8.pdf

Reference

- LaTeX Project Site

- The Comprehensive TeX Archive Network Latest (La)TeX-related packages and software

- TeX Directory Structure, used by many (La)TeX distributions

- Natural Math converts natural language math formulas to LaTeX representation

- Obsolete packages and commands

- Lamport's book *LaTeX: A Document Preparation System*

Templates

- A resource for free high quality LaTeX templates for a variety of applications

- LaTeX template for writing PhD thesis, 2007

- UCL computer department thesis template

- UT thesis template, 2006

- A template that supports an easy conversion to *.odt (*.doc), *.pdf and *.html in one run, 2009

11.3 Package Reference

This is an incomplete 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 Installing Extra Packages.

The list is in alphabetical order.

11.4   Sample LaTeX documents

The easiest way to learn how to use latex is to look at how other people use it. Here is a list of *real world* latex sources that are freely available on the internet. The information here is sorted by application area, so that it is grouped by the scientific communities that use similar notation and LaTeX constructs.

11.4.1   General examples

Tutorial examples, books, and real world uses of LaTeX.

- caption.tex, simple.tex, wrapped.tex

- small2e.tex and sample2e.tex. The "official" sample documents...

- A short example of how to use LaTeX for scientific reports by Stephen J. Eglen.

- The not so Short Introduction to LaTeX by Tobias Oetiker is distributed with full latex sources.

11.4.2   Semantics of Programming Languages

Articles on programming language research, from syntax to semantics, including source code listings, type rules, proof trees, and even some category theory. A good place to start is Mitchell Wand's Latex Resources, including a sample file that also demonstrates Didier Remy's mathpartir package. The following are latex sources of some articles, books, or presentations from this field:

- Pugs: Bootstrapping Perl 6 with Haskell. This paper by Audrey Tang contains nice examples on configuring the listings package to format source code.

11.5   Index

This is an alphabetical index of the book.

11.5.1   A

- Absolute Beginners
- Abstract
- Accents
- Algorithms
- Arrays
- Authors

11.5.2   B

- babel
- Basics
- beamer package
- Bibliography Management
- BibTeX
- Bold
- Bullets
- Bullet points

11.5.3   C

- Captions
- Collaborative Writing of LaTeX Documents
- Color
- color package
- Columns, see Multi-column Pages
- Cross-referencing
- Customizing LaTeX

11.5.4   D

- Dashes
- description environment
- Diactrical marks
- Document Classes
- Document Structure
- Drawings

11.5.5   E

- e.g. (exempli gratia)
- Ellipsis
- em-dash
- en-dash
- enumerate
- Errors and Warnings
- Euro currency symbol
- Export To Other Formats

11.5.6 F

- Figures
- Floats
- Fonts
- Footer, Page
- Footnotes
- Formatting

11.5.7 G

- General Guidelines
- Graphics
  - Creating
  - Embedding
  - Importing
- graphicx package

11.5.8 H

- Header, Page
- HTML output
- Hyperlinks
- hyperref package
- hyphen
- Hyphenation

11.5.9 I

- i.e. (id est)
- Images
- Importing Graphics
- Indexing
- Internationalization
- Introduction
- Italics
- itemize

11.5.10 L

- Labels
- Letters
- Links
- Lists

11.5.11 M

- makeidx package
- \maketitle
- Margin Notes
- Creating Graphics
- Mathematics
- Matrices
- Minipage environment example
- Multi-column Pages

11.5.12 P

- Packages
  - Creating 1
- Page Layout
- PDF output
- picture
- Pictures
- PNG output
- Presentations
- Pseudocode

11.5.13 Q

- LaTeX/Paragraph Formatting#Quoting_text

11.5.14 R

- References
- RTF output

11.5.15   S

- Sentences
- Small Capitals
- Source Code Listings
- Space Between Words
- Spell-checking
- Superscript and subscript: powers and indices
- Superscript and subscript: text mode
- SVG output

11.5.16   T

- Table of contents
- Tables
- Teletype text
- Text Size
- Theorems
- Tips and Tricks
- Title Creation

11.5.17   U

- URLs

11.5.18   V

- Verbatim Text

11.5.19   W

- Word Counting

11.5.20   X

- XeTeX
- XY-pic package
- xy package

11.6   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.)

11.6.1

/   see slash marks

\@   following period ends sentence

\\[*][extra-space]   new line

\,   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

11.6.2   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 Length

\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

11.6.3 B

\backslash prints a backslash

\baselineskip a length command (see Lengths), 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

\boldsymbol bold font for symbols

11.6.4 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. LaTeX/Colors

\copyright makes © sign. See Formatting.

11.6.5 D

\dashbox

\date

\ddots Inserts a diagonal ellipsis (3 diagonal dots) in math mode

\documentclass[options]{style} Used to begin a latex document

\dotfill

11.6.6 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) Treats everything inside the curly braces as if it were in a math environment. Useful when creating commands in the preamble as they will work inside or out of math environments.

\epigraph Adds an epigraph. Requires epigraph package.

\euro Prints euro € symbol. Requires eurosym package.

11.6.7 F

\fbox

\flushbottom

\fnsymbol

\footnote Creates a footnote.

\footnotemark

\footnotesize Sets font size. See Text 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 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.

11.6.8 G

11.6.9 H

\hfill Abbreviation for \hspace{\fill}.

\hline adds a horizontal line in a tabular environment. See also \cline, Tables chapter.

\href Add a link, or an anchor. See Hyperlinks

\hrulefill

\hspace   Produces horizontal space.

\huge   Sets font size. See Text Formatting.

\Huge   Sets font size. See Text Formatting.

\hyphenation{word list}   Overrides default hyphenation algorithm for specified words. See Hyphenation

11.6.10   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 {\itThis is in italics}. \em is generally preferred since this allows nesting.

\item   Creates an item in a list. Used in list structures.

11.6.11   K

\kill   Prevent a line in the tabbing environment from being printed.

11.6.12   L

\label   Used to create label which can be later referenced with \ref. See Labels and Cross-referencing.

\large   Sets font size. See Text Formatting.

\Large   Sets font size. See Text Formatting.

\LARGE   Sets font size. See Text 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 TOC

\listoftables   Inserts a list of the tables in the document. Similar to TOC

\location

11.6.13   M

\makebox   Defines a box that has a specified width, independent from its content. See Boxes.

\maketitle   Causes the title page to be typeset, using information provided by commands such as \title{} and \author{}.

\markboth \markright

\mathcal

\mathop

\mbox   Write a text in roman font inside a math part

\medskip

\multicolumn

\multiput

11.6.14   N

\newcommand   Defines a new command. See New Commands.

\newcolumntype   Defines a new type of column to be used with tables. See Tables.

\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 Text Formatting.

\nopagebreak Suggests LaTeX not to break page in this place. See Page Layout.

\not

11.6.15   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.

11.6.16   P

\pagebreak Suggests LaTeX breaking page in this place. See Page Layout.

\pagenumbering Defines the type of characters used for the page numbers. Options : arabic, roman, Roman, alph, Alph, gobble (invisible).

\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 Boxes.

\parindent Normal paragraph indentation. See Lengths.

\parskip

\part Starts a new part of a book. See Document Structure.

\protect

\providecommand (LaTeX2e) See Macros.

\put

11.6.17   Q

\quad Similar to space, but with the size of a capital M

\qquad double \quad

11.6.18   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/Boxes.

\ref Used to reference to number of previously declared \label. See Labels and Cross-referencing.

\renewcommand

\right

\rm Roman typeface.

\roman Causes a counter to be printed in roman numerals.

\rule Creates a line of specified width and height. See LaTeX/Rules and Struts.

11.6.19   S

\savebox Makes a box and saves it in a named storage bin.

\sbox The short form of \savebox with no optional arguments.

\sc Small caps.

\scriptsize Sets font size. See Text Formatting.

\section Starts a new section. See Document Structure.

\setcounter

\setlength

\settowidth

\sf Sans serif.

\shortstack

\signature In the *Letter* class, specifies a signature for later insertion by \closing.

\sl Slanted.

\slash See slash marks

\small   Sets font size. See Text Formatting.

\smallskip

\sout   Strikes out text. Requires ulem package. See Text 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.

\stepcounter   Increase the counter.

\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.

11.6.20   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 Text Formatting.

\textbf{}   Sets bold font style. See Text Formatting.

\textcolor{}{}   Creates colored text. See Entering colored text.

\textit{}   Sets italic font style. See Text Formatting.

\textmd{}   Sets medium weight of a font. See Text Formatting.

\textnormal{}   Sets normal font. See Text Formatting.

\textrm{}   Sets roman font family. See Text Formatting.

\textsc{}   Sets font style to small caps. See Text Formatting.

\textsf{}   Sets sans serif font family. See Text Formatting.

\textsl{}   Sets slanted font style. See Text Formatting.

\texttt{}   Sets typewriter font family. See Text Formatting.

\textup{}   Sets upright shape of a font. See Text Formatting.

\textwidth

\textheight

\thanks

\thispagestyle

\tiny   Sets font size. See Text Formatting.

\title

\today   Writes current day. See Text Formatting.

\tt

\twocolumn

\typeout

\typein

11.6.21   U

\uline   Underlines text. Requires ulem package. See Formatting.

\underbrace

\underline

\unitlength

\usebox

\usecounter

\uwave   Creates wavy underline. Requires ulem package. See Formatting.

11.6.22   V

\value

\vbox{text}   Encloses a paragraph's text to prevent it from running over a page break

\vcenter

\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.

Chapter 12

Text and image sources, contributors, and licenses

12.1 Text

- **LaTeX/Introduction** *Source:* https://en.wikibooks.org/wiki/LaTeX/Introduction?oldid=3090098 *Contributors:* Ævar Arnfjörð Bjarmason, Tualha, Iamunknown, 3mta3, Jraregris, Goodgerster, Derbeth, Graemeg~enwikibooks, Elwikipedista~enwikibooks, Withinfocus, Pdelong, Orderud, Whiteknight, Kernigh, Latexing, Sgenier~enwikibooks, Igjimh, Alejo2083, Chazz, Gronau~enwikibooks, Xania, Rehoot, Sjlegg, Conrad.Irwin, Urhixidur, Nbrouard, Vadik wiki, Pi zero, Dan Polansky, Waldir, Angus, QuiteUnusual, Piksi, Kri, Mckay, Tom Morris, Avila.gas, Nobelium, Ambrevar, Martinkunev, PAC2, Anarchyboy, Tomato86, Staticshakedown, E.lewis1, InverseHypercube, Polytropos Technikos, Torbjorn T., TinyTimZamboni, AllenZh, Ppadmapriya, Benson Muite, Steelangel, Reddraggone9, DoVlaNik993 and Anonymous: 75

- **LaTeX/Installation** *Source:* https://en.wikibooks.org/wiki/LaTeX/Installation?oldid=3138585 *Contributors:* 3mta3, Koavf, Michael-Billington, Mwtoews, Chazz, Bryant1410, Pi zero, Marcus Cyron, Gyro Copter, Kazkaskazkasako, QuiteUnusual, Nobelium, Ambrevar, PAC2, Jason barrington~enwikibooks, Mfwitten, Tomato86, Benregn, Torbjorn T., LlamaAl, Razr Nation, Johannes Bo, RainCity471, Leaderboard, Inferno986return, Fdar, Ah3kal, Dplaza000, Simplelatex, Alansandiego, DoVlaNik993, Metis spawn, Ptaherkhani and Anonymous: 49

- **LaTeX/Installing Extra Packages** *Source:* https://en.wikibooks.org/wiki/LaTeX/Installing_Extra_Packages?oldid=3038524 *Contributors:* Alejo2083, Mwtoews, Az1568, Xania, Pi zero, Zylorian, QuiteUnusual, Ollydbg, Mabdul, Ambrevar, Lotus noir, Netheril96, DarkSheep, Dplaza000, DoVlaNik993 and Anonymous: 23

- **LaTeX/Basics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Basics?oldid=3146892 *Contributors:* Krischik, Chuckhoffmann, 3mta3, ABCD, Derbeth, Jonathan Webley, Withinfocus, Snaxe920~enwikibooks, Jtwdog~enwikibooks, Tpr~enwikibooks, Jguk, Bilbo1507, Igjimh, Dilaudid, Alejo2083, Mwtoews, Thenub314, Rehoot, Vadik wiki, Qeny, Atallcostsky, Pi zero, Waldir, MER-C, TomyDuby, CrazyTerabyte, Kri, Adrignola, Constantine, Ambrevar, PAC2, Chisophugis, Immae, Janskalicky, Tomato86, Robbiemorrison, E.lewis1, BrettMontgomery, Guyrobbie, Patuck, ATC2~enwikibooks, BwDraco, Guzo, Wickedjargon, Pater Christophorus, Johannes Bo, Reddraggone9, Negative24, Dplaza000, Tropicalkitty, BbcNkl and Anonymous: 87

- **LaTeX/Document Structure** *Source:* https://en.wikibooks.org/wiki/LaTeX/Document_Structure?oldid=3137527 *Contributors:* Derbeth, Withinfocus, Orderud, Stephan Schneider, Snaxe920~enwikibooks, Jtwdog~enwikibooks, Jomegat, Nkour, Igjimh, Dilaudid, Alejo2083, JECompton~enwikibooks, Mwtoews, BiT, Thenub314, ConditionalZenith~enwikibooks, Xania, FlashSheridan, Spelemann~enwikibooks, Kovianyo, Recent Runes, Astrophizz~enwikibooks, Spag85~enwikibooks, Pi zero, Tully~enwikibooks, Waldir, Nux, Jan Winnicki, QuiteUnusual, Neet, Kri, Lovibond, Adrignola, Hosszuka, Wdcf, Ambrevar, DagErlingSmørgrav, Lucasreddinger, Keplerspeed, Dirk Hünniger, Frap, Tomato86, Derwaldrandfoerster, Echeban, Torbjorn T., LlamaAl, Ntypanski, Wickedjargon, Genethecist, Johannes Bo, Reddraggone9, Kurlovitsch, Kw CUACS.TOPS, Vanjanssen, BbcNkl and Anonymous: 80

- **LaTeX/Text Formatting** *Source:* https://en.wikibooks.org/wiki/LaTeX/Text_Formatting?oldid=3033246 *Contributors:* Derbeth, Withinfocus, Igjimh, Thenub314, Pi zero, ChrisHodgesUK, Adrignola, Ambrevar, Dirk Hünniger, Robbiemorrison, Fishix, Bgeron, Johannes Bo, RubensMatos, Leaderboard, Kurlovitsch, Lindhea, Maths314, Kevin Ryde, Paulgush, BbcNkl and Anonymous: 21

- **LaTeX/Paragraph Formatting** *Source:* https://en.wikibooks.org/wiki/LaTeX/Paragraph_Formatting?oldid=3132903 *Contributors:* Ish ishwar~enwikibooks, ABCD, Derbeth, QuantumEleven, Jonathan Webley, Withinfocus, Orderud, Jtwdog~enwikibooks, Andyr~enwikibooks, Igjimh, Alejo2083, Mwtoews, Thenub314, ConditionalZenith~enwikibooks, Mcld, Kovianyo, MartinSpacek, Crasshopper, Recent Runes, Pi zero, Tully~enwikibooks, Owl-syme, Hjsb, Yez, Waldir, CarsracBot, Hannes Röst, Rror, Adrianwn, ChrisHodgesUK, Pstar, QuiteUnusual, Neet, LR~enwikibooks, Adrignola, Nixphoeni, PatrickDevlin21, Ambrevar, Zzo38, Cdecoro, Keplerspeed, Zrisher, GPHemsley, Henrybissonnette, TorfusPolymorphus, Smobbl Bobbl, ToematoeAdmn, Dirk Hünniger, Mouselb, EvanKroske, Tomato86, Fishpi, Robbiemorrison, Tazquebec, Brammers, Listdata, Neoriddle, Xonqnopp, C3l, Harrikoo, Gmh04~enwikibooks, InverseHypercube, Halilsen, RealSebix, Akim Demaille, Dlituiev, SamuelLB, CD-Stevens, Incognito668, Gmacar, Dplaza000 and Anonymous: 148

- **LaTeX/Colors** *Source:* https://en.wikibooks.org/wiki/LaTeX/Colors?oldid=3133469 *Contributors:* Alejo2083, Az1568, Xania, Pi zero, Dan Polansky, Daniel Mietchen, Waldir, Gms, JackPotte, White gecko, ChrisHodgesUK, Kazkaskazkasako, Trace, QuiteUnusual, Kri, Adrignola, Benjaminevans82~enwikibooks, ChristianGruen, Sanderd17, Scorwin, Ambrevar, PAC2, Velociostrich, Dirk Hünniger, Tomato86, Conighion, Qzxpqbp, SamuelLB, Henry Tallboys, Nicolas Perrault III, Tisep, Johannes Bo, Tpapastylianou, Honza889, Dplaza000, David Gaudet, Ks-M9, DoVlaNik993 and Anonymous: 64

- **LaTeX/Lengths** *Source:* https://en.wikibooks.org/wiki/LaTeX/Lengths?oldid=3119815 *Contributors:* Panic2k4, Derbeth, Withinfocus, Hokiehead~enwikibooks, Igjimh, Alejo2083, Pi zero, Scruss, Waldir, Ollydbg, Kri, Adrignola, Ambrevar, Dendik, Crissov, Xonqnopp, Ediahist, Atcovi, Yeshua Saves, Strpeter and Anonymous: 21

- **LaTeX/Counters** *Source:* https://en.wikibooks.org/wiki/LaTeX/Counters?oldid=3007015 *Contributors:* Ambrevar, Yeshua Saves and Anonymous: 4

- **LaTeX/Boxes** *Source:* https://en.wikibooks.org/wiki/LaTeX/Boxes?oldid=3104885 *Contributors:* Selfworm, LivingShadow, Kri, Ambrevar, Kundor, AthanasiusOfAlex, Atcovi, Yeshua Saves, Toogley and Anonymous: 10

- **LaTeX/Rules and Struts** *Source:* https://en.wikibooks.org/wiki/LaTeX/Rules_and_Struts?oldid=3057919 *Contributors:* Ollydbg, Ambrevar, Johannes Bo, DrTobbe and Anonymous: 1

- **LaTeX/Mathematics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Mathematics?oldid=3133441 *Contributors:* Insaneinside, 3mta3, Juliusross, Atiq ur Rehman, Derbeth, Jonathan Webley, Withinfocus, Orderud, Mecanismo, Ans, Jtwdog~enwikibooks, Jomegat, Krishnavedala, Robin~enwikibooks, Cameronc~enwikibooks, Nigels~enwikibooks, Pirround, Hagindaz, Igjimh, Dilaudid, Ms2ger, Alejo2083, Basenga, Mwtoews, Swift, BiT, DavidMcKenzie, Conrad.Irwin, Dncarley, Elliptic1, Unco, Mcld, Recent Runes, Bonuama, Adam majewski, Pi zero, Jeff G., Waldir, Marra, This, that and the other, JackPotte, Mhue, TomyDuby, Geminatea~enwikibooks, ChrisHodgesUK, Whym, Germanzs, Cícero, QuiteUnusual, Winfree, Joe Schmedley, Neet, Asmeurer, Kri, Adrignola, Kwetal, Tork73, Prispartlow, Sargas~enwikibooks, Götz, YuryKirienko, Avila.gas, Karcih, Ambrevar, Lucasreddinger, Keplerspeed, Kwpolska, Xnn, Merciadriluca, Wmheric, Kubieziel, Dirk Hünniger, Jodi.a.schneider, Justin W Smith, Tom.marcik, Tomato86, Uuhuuӊ, Robbiemorrison, Netheril96, Kejia, Karper, Etoombs, Clebell, InverseHypercube, Krst, Greenbreen, Karthicknainar~enwikibooks, Fsart, Chafe66, Tgwizard, DmitriyZotikov, McSaks, SamuelLB, Jluttine, Comput2h, Wp4bl0, Volvens, Mandriver, Anthony Deschamps, Ntypanski, Razr Nation, Theemathas, Franklin Yu, Johannes Bo, Unlikelyuser, Leaderboard, Briancricks, Strpeter, Komputerwiz, Hapli, Mgkrupa, Syum90, Bigwyrm~enwikibooks, Yotann, User000name, Vioricavinersan, Infinite0694, Mpvharmelen, Mhartl, Ks-M9, Anareth and Anonymous: 222

- **LaTeX/Advanced Mathematics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Advanced_Mathematics?oldid=3131923 *Contributors:* Sbeyer, 3mta3, Jonathan Webley, Ynhockey, Inductiveload, Chazz, BiT, Thenub314, Ojan, Pi zero, Leyo, ChrisHodgesUK, QuiteUnusual, Control.valve, Adrignola, Nixphoeni, Silca678, LinuxChristian~enwikibooks, Avila.gas, TorfusPolymorphus, Simonjtyler, Tomato86, Zaslav, Kcho, Geetha nitc, MagnusPI~enwikibooks, Jaleks, Codairem, Krst, Arthurvogel, SamuelLB, Escalator~enwikibooks, Volvens, Garfl, Paul2520, Johannes Bo, Atcovi, Leaderboard, JW 00000, Dizzyzane, Kevin Ryde and Anonymous: 50

- **LaTeX/Theorems** *Source:* https://en.wikibooks.org/wiki/LaTeX/Theorems?oldid=3039766 *Contributors:* 3mta3, Derbeth, Raylu, Alejo2083, Mwtoews, Kovianyo, Pi zero, Mhue, Tosha, Juliabackhausen, Ambrevar, Tomato86, SamuelLB, Ajmath62, CtrlAltCarrot, DoVlaNik993 and Anonymous: 25

- **LaTeX/Chemical Graphics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Chemical_Graphics?oldid=3135053 *Contributors:* Chazz, CommonsDelinker, Ysangkok, QuiteUnusual, Clapsus, Ambrevar, Dirk Hünniger, Andrea09falco, Wickedjargon, 2147483647, Johannes Bo, Daviewales, DrSarah Calumet, Doking23, Mullins CZ and Anonymous: 8

- **LaTeX/Algorithms** *Source:* https://en.wikibooks.org/wiki/LaTeX/Algorithms?oldid=3119510 *Contributors:* ZeroOne, Derbeth, Alejo2083, Mwtoews, Mcld, Webinn, Pi zero, Dan Polansky, Thefrankinator, Lavaka, Nemti, Whym, Tauriel-1, Adrignola, امیر اعوانی, Maartenweyn, Gkc~enwikibooks, Sanderd17, Sargas~enwikibooks, Saehrimnir, Ambrevar, JenVan, Debejyo, Mrt doulaty, Brevity, Writalnaie, Tomato86, Fishpi, Pandora85, Bhanuvrat, Nsda, SamuelLB, Glad~enwikibooks, Karategeek6, Szellmann, Jimmaykeepsitreal, Johannes Bo, Harish victory, Lteu, Uwe Hartwig, Doking23 and Anonymous: 94

- **LaTeX/Source Code Listings** *Source:* https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings?oldid=3141115 *Contributors:* Panic2k4, Jomegat, Alejo2083, Nicolasbock, Pi zero, Vaucouleur, Drevicko, QuiteUnusual, Hosszuka, Bunyk, Gladiool, Ambrevar, Leal26, Arthurchy, Helptry, Evin~enwikibooks, Tomato86, Robbiemorrison, Diomidis Spinellis, Wenzeslaus, Tau Lambda, Sonic the goliath, Yeshua Saves, RasmusWriedtLarsen, Felipecarres, Dieguico, Doking23 and Anonymous: 54

- **LaTeX/Linguistics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Linguistics?oldid=3016826 *Contributors:* Neatnate, Xania, Matěj Grabovský, ChrisHodgesUK, Joe Schmedley, Ambrevar, Dirk Hünniger, SynConlanger, Ngoclong19, DmitriyZotikov, Ghoti, PhilJohnG, Obelyaev, Olesh, Matej.korvas, Hankjones, SamuelLB, Masterpiga, Columbus240 and Anonymous: 14

- **LaTeX/Indexing** *Source:* https://en.wikibooks.org/wiki/LaTeX/Indexing?oldid=3075752 *Contributors:* Itai, Derbeth, Mecanismo, Stephan Schneider, Alejo2083, Mwtoews, Kovianyo, Uluboz, He7d3r, Pi zero, Neoptolemus, Dan Polansky, Tully~enwikibooks, Morelight~enwikibooks, Petter Strandmark, Neet, Kri, Adrignola, Sargas~enwikibooks, WardMuylaert, Dirk Hünniger, Tomato86, Semperos, Nsuwan, Halilsen, SamuelLB, Toothbrushbrush, Bombcar, DoVlaNik993 and Anonymous: 16

- **LaTeX/Glossary** *Source:* https://en.wikibooks.org/wiki/LaTeX/Glossary?oldid=3095175 *Contributors:* Panic2k4, Recent Runes, Dan Polansky, Redirect fixer, QuiteUnusual, Speravir, Pmlineditor, Ambrevar, Topodelapradera, Dirk Hünniger, Tomato86, Robbiemorrison, Bro4, RealSebix, Lancioni, SamuelLB, MoMaT, Hendiadyon, Jessevanassen, Berettag, Tlinnet, Ffavela, Tanzaho, Gmacar, Tonda, Donok11, DoVlaNik993, Anamma06 and Anonymous: 39

- **LaTeX/Bibliography Management** *Source:* https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management?oldid=3109760 *Contributors:* ZeroOne, Derbeth, Withinfocus, Orderud, Stephan Schneider, Jtwdog~enwikibooks, Igjimh, Alejo2083, Mwtoews, Braindrain0000, Qwertyus, Rondenaranja~enwikibooks, Thenub314, Drewbie, Oderbolz, Spelemann~enwikibooks, Amamory, Kovianyo, Nbrouard, MartinSpacek, Blaisorblade, He7d3r, Spag85~enwikibooks, Pi zero, Kevang, Waldir, Mhue, MichaelSchoenitzer, ChrisHodgesUK, Kazkaskazkasako, Kroolik, Arbitrarily0, QuiteUnusual, Kri, Vermiculus~enwikibooks, Silca678, Rafaelgr, Rbonvall, Eyliu~enwikibooks, StevenJohnston, Sandman10000, Sargas~enwikibooks, Nobelium, Ambrevar, Tdomhan, Lucasreddinger, John1923, Pilosa.Folivora, TorfusPolymorphus, MichaelBueker, Topodelapradera, Dirk Hünniger, Lbailey45, Robbiemorrison, Rossdub, Crissov, Xonqnopp, Jevon, Norbert.beckers, Mr. Stradivarius, Bro4, RealSebix, FranklyMyDear..., Gillespie09, Literaturgenerator, QUBot, PatrickGalyon, SamuelLB, Jluttine, Austinmohr, Johannes Bo, Ch605852, Gmacar, Kurlovitsch, Jpoosterhuis, Epic Wink, Syum90, Crabpot8, Lindhe94, Fdar, Kdonavin, Infinite0694, AdhillA97, Einjohn, Springthyme and Anonymous: 162

- **LaTeX/More Bibliographies** *Source:* https://en.wikibooks.org/wiki/LaTeX/More_Bibliographies?oldid=3136439 *Contributors:* Adrignola, Ambrevar, Jimbotyson, Billy the Goat II, Remsirems, Wootery, Johannes Bo, Savicivas and Anonymous: 15

- **LaTeX/Letters** *Source:* https://en.wikibooks.org/wiki/LaTeX/Letters?oldid=3082288 *Contributors:* Derbeth, Edudobay, Mwtoews, Xania, Jld, Garoth~enwikibooks, Pi zero, Neoptolemus, CarsracBot, Gms, ChrisHodgesUK, QuiteUnusual, Ambrevar, GavinMcGimpsey, Tomato86, Neoriddle, E.lewis1, SamuelLB, Empirical bayesian, TheAnarcat, Atcovi, VitoFrancisco, Savicivas, DoVlaNik993 and Anonymous: 32

- **LaTeX/Presentations** *Source:* https://en.wikibooks.org/wiki/LaTeX/Presentations?oldid=3133291 *Contributors:* Ish ishwar~enwikibooks, 3mta3, Derbeth, Xania, Kovianyo, Pi zero, Ramac, Dan Polansky, Flal, Jayk~enwikibooks, ChrisHodgesUK, FredrikMeyer, Eyliu~enwikibooks, Dreaven3, PAC, Sanderd17, Sargas~enwikibooks, Avila.gas, Ambrevar, Gryllida, Stuples, PAC2, NqpZ, Helptry, Grj23, Mezzaluna, Flip, Robbiemorrison, Neoriddle, Xonqnopp, Ghoti, SamuelLB, Dr0pi, Espinozahg, Goldkatze, Hapli, Lindhe94, Aeonblue158, Jerome.dequeker, Irenas996, DoVlaNik993, Drasquared and Anonymous: 55
- **LaTeX/Teacher's Corner** *Source:* https://en.wikibooks.org/wiki/LaTeX/Teacher'{}s_Corner?oldid=2983311 *Contributors:* PAC, Ambrevar, PAC2, Tomato86, SamuelLB, Defender, Hapli and Anonymous: 10
- **LaTeX/Curriculum Vitae** *Source:* https://en.wikibooks.org/wiki/LaTeX/Curriculum_Vitae?oldid=3121316 *Contributors:* Jomegat, QuiteUnusual, Ambrevar, Reyk, Mariafenrinha, Johannes Bo, Jianhui67, CallumPoole, DoVlaNik993 and Anonymous: 13
- **LaTeX/Introducing Procedural Graphics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Introducing_Procedural_Graphics?oldid=3040466 *Contributors:* 3mta3, Kenyon, Jonathan Webley, Jomegat, Alejo2083, Mwtoews, Qwertyus, Rajkiran g, Pi zero, Dan Polansky, Thefrankinator, Jacobrothstein, Niel.Bowerman, Simeon, ChrisHodgesUK, Tosha, Kri, Пика Пика, Adrignola, Ambrevar, Keplerspeed, Unbitwise, Rnddim, Dirk Hünniger, Jdgilbey, Jflycn, Krisrose~enwikibooks, Tomato86, Kuer.gee, Wikieditoroftoday, Pmillerrhodes, Atulya1988, Franzl aus tirol, Wxm29, SamuelLB, Doking23 and Anonymous: 23
- **LaTeX/MetaPost** *Source:* https://en.wikibooks.org/wiki/LaTeX/MetaPost?oldid=2473798 *Contributors:* Ambrevar
- **LaTeX/Picture** *Source:* https://en.wikibooks.org/wiki/LaTeX/Picture?oldid=3040253 *Contributors:* Svick, Ambrevar, Wxm29, Tentotwo, Doking23 and Anonymous: 9
- **LaTeX/PGF/TikZ** *Source:* https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ?oldid=3145208 *Contributors:* JonnyJD, Krishnachandranvn, QuiteUnusual, Kri, Nobelium, Ambrevar, Franzl aus tirol, Dlituiev, Haneluya, Dizzyzane, KlasN, Cameronc, Ccnu2012 and Anonymous: 12
- **LaTeX/PSTricks** *Source:* https://en.wikibooks.org/wiki/LaTeX/PSTricks?oldid=3087909 *Contributors:* ChrisHodgesUK, Ambrevar, Ngoclong19, Skim, Netheril96, Karlberry, Doking23 and Anonymous: 3
- **LaTeX/Xy-pic** *Source:* https://en.wikibooks.org/wiki/LaTeX/Xy-pic?oldid=3040389 *Contributors:* ChrisHodgesUK, Kri, Ambrevar, Dirk Hünniger, Ngoclong19 and Doking23
- **LaTeX/Creating 3D graphics** *Source:* https://en.wikibooks.org/wiki/LaTeX/Creating_3D_graphics?oldid=3040392 *Contributors:* Kri, Nobelium, Ambrevar, Doking23 and Anonymous: 1
- **LaTeX/Macros** *Source:* https://en.wikibooks.org/wiki/LaTeX/Macros?oldid=3078278 *Contributors:* Risk, Mecanismo, Alejo2083, Mwtoews, Pi zero, Tully~enwikibooks, Hsmyers~enwikibooks, Waldir, ChrisHodgesUK, LR~enwikibooks, Kri, Adrignola, Hosszuka, Ambrevar, Go.pbam., Dirk Hünniger, Robbiemorrison, Aadornellesf, SamuelLB, Funkenstern, Johannes Bo, AsphyxiateDrake, Yeshua Saves, Efex3, Mrwil222 and Anonymous: 41
- **LaTeX/Plain TeX** *Source:* https://en.wikibooks.org/wiki/LaTeX/Plain_TeX?oldid=3120241 *Contributors:* Blaisorblade, Ambrevar, Yeshua Saves, Bamgooly, Sheep0x and Anonymous: 11
- **LaTeX/Creating Packages** *Source:* https://en.wikibooks.org/wiki/LaTeX/Creating_Packages?oldid=3064616 *Contributors:* Ambrevar, Johannes Bo, HansCronau and Anonymous: 6
- **LaTeX/Themes** *Source:* https://en.wikibooks.org/wiki/LaTeX/Themes?oldid=2504675 *Contributors:* Ambrevar and Anonymous: 1
- **LaTeX/Modular Documents** *Source:* https://en.wikibooks.org/wiki/LaTeX/Modular_Documents?oldid=3144786 *Contributors:* Derbeth, Alejo2083, Mwtoews, Mike.lifeguard, Pi zero, ChrisHodgesUK, Neet, Adrignola, Nixphoeni, Liiiii, Frakturfreund, Zyqqh~enwikibooks, Ambrevar, Robbiemorrison, Neoriddle, Ediahist, MaBoehm, SamuelLB, CD-Stevens, Vinaisundaram, Stefan.qn~enwikibooks, Johannes Bo, Wgjbeek, RP87, MfR, David Ollodart and Anonymous: 27
- **LaTeX/Collaborative Writing of LaTeX Documents** *Source:* https://en.wikibooks.org/wiki/LaTeX/Collaborative_Writing_of_LaTeX_Documents?oldid=3109763 *Contributors:* Derbeth, Arnehe, Pi zero, Madskaddie, Schaber, Waldir, ChrisHodgesUK, QuiteUnusual, Tosha, Kpym, Tim Parenti, Ambrevar, Keplerspeed, Jason barrington~enwikibooks, Dirk Hünniger, Bamgooly~enwikibooks, Robbiemorrison, Glosser.ca, Wikieditoroftoday, Jmcdon10, Hermine potter, SamuelLB, Jbsnyder, Fdar and Anonymous: 37
- **LaTeX/Export To Other Formats** *Source:* https://en.wikibooks.org/wiki/LaTeX/Export_To_Other_Formats?oldid=3137145 *Contributors:* Derbeth, Jomegat, Alejo2083, Mwtoews, Az1568, Pi zero, ChrisHodgesUK, Bpsullivan~enwikibooks, Bakken, Nobelium, Ambrevar, Keplerspeed, SamuelLB, Wenzeslaus, Migueldvb, Xin-Xin W., Stefan.qn~enwikibooks, Maths314, Doking23 and Anonymous: 33
- **LaTeX/FAQ** *Source:* https://en.wikibooks.org/wiki/LaTeX/FAQ?oldid=2962525 *Contributors:* Xania, Kayau, Nixphoeni, Ambrevar, Ghostofkendo, Dredmorbius, Sobjornstad and Anonymous: 11
- **LaTeX/Tips and Tricks** *Source:* https://en.wikibooks.org/wiki/LaTeX/Tips_and_Tricks?oldid=3014469 *Contributors:* Derbeth, Withinfocus, Jguk, Alejo2083, Basenga, Mwtoews, Filip Dominec, Pi zero, Towsonu2003~enwikibooks, Brendanarnold, ChrisHodgesUK, Winniehell, Limpato, Vaffelkake, Sargas~enwikibooks, Ambrevar, Bumbulski, Tomato86, Robbiemorrison, Wikieditoroftoday, Jamoroch~enwikibooks, Russell208, SamuelLB, Johannes Bo and Anonymous: 44
- **LaTeX/Authors** *Source:* https://en.wikibooks.org/wiki/LaTeX/Authors?oldid=2472489 *Contributors:* Derbeth, Alejo2083, Pi zero, Ambrevar and Anonymous: 1
- **LaTeX/Links** *Source:* https://en.wikibooks.org/wiki/LaTeX/Links?oldid=3009072 *Contributors:* Derbeth, Jonathan Webley, Withinfocus, Orderud, Dmb, Alejo2083, Jasu, Pi zero, Ambrevar, RaymondSutanto, Rotlink, Eselmeister and Anonymous: 8
- **LaTeX/Package Reference** *Source:* https://en.wikibooks.org/wiki/LaTeX/Package_Reference?oldid=3134256 *Contributors:* Dan Polansky, Ambrevar, Wickedjargon, Dredmorbius, Johannes Bo and Anonymous: 1
- **LaTeX/Sample LaTeX documents** *Source:* https://en.wikibooks.org/wiki/LaTeX/Sample_LaTeX_documents?oldid=2465029 *Contributors:* Thenub314, Vesal, Pi zero, Ambrevar and Anonymous: 2
- **LaTeX/Index** *Source:* https://en.wikibooks.org/wiki/LaTeX/Index?oldid=2768358 *Contributors:* Derbeth, Jonathan Webley, Pi zero, Ramac, Dan Polansky, Owl-syme, TomyDuby, Avila.gas, Henridv, Ambrevar, Dirk Hünniger, Kevin Ryde and Anonymous: 1
- **LaTeX/Command Glossary** *Source:* https://en.wikibooks.org/wiki/LaTeX/Command_Glossary?oldid=3155667 *Contributors:* Robert Horning, Derbeth, Withinfocus, Orderud, Naught101, Dmb, Igjimh, IrfanAli, Spelemann~enwikibooks, Paxinum, Pi zero, Dan Polansky, TomyDuby, ChrisHodgesUK, Pstar, Adrignola, Tuetschek, Kpym, Royote, Stoettner, Dirk Hünniger, Robbiemorrison, Abustany, Xeracles, Avanhel and Anonymous: 30

12.2 Images

- **File:50_percents.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/62/50_percents.svg *License:* CC0 *Contributors:* File: 50%.svg *Original artist:* Ftiercel

- **File:Acetate-ion.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/23/Acetate-ion.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Pmillerrhodes

- **File:Acetate-ion2.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b3/Acetate-ion2.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Pmillerrhodes

- **File:Alltt.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/5c/Alltt.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Dirk Hünniger

- **File:BibDesk-1.3.10-screenshot.png** *Source:* https://upload.wikimedia.org/wikibooks/en/c/ce/BibDesk-1.3.10-screenshot.png *License:* Public domain *Contributors:*

  own work

  Original artist:

  Mij

- **File:BibDesk1.3.8.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/63/BibDesk1.3.8.jpg *License:* Public domain *Contributors:* My computer *Original artist:* Myself

- **File:Blocks_beamer_example.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/20/Blocks_beamer_example.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Israel Buitron

- **File:Book_important2.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/91/Book_important2.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* darklama

- **File:Bordermatrix.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/cc/Bordermatrix.png *License:* Public domain *Contributors:* Own work *Original artist:* Winfree

- **File:Butane-skeletal.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/eb/Butane-skeletal.png *License:* Public domain *Contributors:* self-made in ChemDraw *Original artist:* Ben Mills

- **File:Carbon_Lewis_Structure_PNG.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/5b/Carbon_Lewis_Structure_PNG.png *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Daviewales

- **File:Cat-eats-cream.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/16/Cat-eats-cream.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Olesh

- **File:Chemfig_Arrow_Examples.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bb/Chemfig_Arrow_Examples.png *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Clapsus

- **File:Chemfig_angles.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/0f/Chemfig_angles.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Pmillerrhodes

- **File:Chemfig_bonds.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/0f/Chemfig_bonds.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Pmillerrhodes

- **File:Chick1.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/54/Chick1.png *License:* CC-BY-SA-3.0 *Contributors:* Transferred from en.wikibooks to Commons by Sanderd17 using CommonsHelper. *Original artist:* The original uploader was Jtwdog at English Wikibooks

- **File:Clipboard.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/1f/Clipboard.svg *License:* GPL *Contributors:* Own work, based on File:Evolution-tasks-old.png, which was released into the public domain by its creator AzaToth. *Original artist:* Tkgd2007

- **File:Corticosterone_(1).png** *Source:* https://upload.wikimedia.org/wikipedia/commons/7/7c/Corticosterone_%281%29.png *License:* Public domain *Contributors:* Transferred from en.wikipedia to Commons by Sfan00_IMG using CommonsHelper. *Original artist:* The original uploader was Iorsh at English Wikipedia

- **File:ESvn-texmf.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/15/ESvn-texmf.png *License:* GPL *Contributors:* Transferred from en.wikibooks; transferred to Commons by User:Adrignola using CommonsHelper. *Original artist:* Original uploader was Arnehe at en.wikibooks

- **File:Emph.png** *Source:* https://upload.wikimedia.org/wikibooks/en/a/a8/Emph.png *License:* GFDL *Contributors:* ? *Original artist:* ?

- **File:Envelope.jpg** *Source:* https://upload.wikimedia.org/wikibooks/en/e/ef/Envelope.jpg *License:* Public domain *Contributors:* ? *Original artist:* ?

- **File:Example_of_French_quotation_marks.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/01/Example_of_French_quotation_marks.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Karl Scheel

- **File:Example_of_German_quotation_marks.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/55/Example_of_German_quotation_marks.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Karl Scheel

- **File:Frametitle_keyword_example.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/8c/Frametitle_keyword_example.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Israel Buitron

- **File:Gb4e1.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/67/Gb4e1.png *License:* CC0 *Contributors:* Own work *Original artist:* jeg

- **File:Gb4e2.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/0c/Gb4e2.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Hankjones

- **File:Gb4e3.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/67/Gb4e3.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Hankjones

12.3 Content license