

The Kile Handbook

**Jonathan Pechta
Federico Zenith
Holger Danielsson
Thomas Braun
Michel Ludwig
Felix Mauch**



The Kile Handbook

Contents

1	Preface	9
1.1	Requirements	9
1.2	Intended Audience	9
2	Introduction	10
2.1	Basic facts	10
2.1.1	About Kile	10
2.1.2	Kile and the Kate Editor Component	10
2.1.3	What is L ^A T _E X?	10
2.1.4	How do you pronounce it? Why that strange typesetting?	10
2.2	L ^A T _E X 101	11
2.3	Kile's Main Features	11
2.3.1	QuickStart Wizard	11
2.3.2	Predefined Templates	12
2.3.3	Syntax Highlighting	12
2.3.4	Auto-Completion of Environments	12
2.3.5	Jump to Structure Element	13
2.3.6	Inverse Search	13
2.3.7	Forward Search	13
2.4	The Toolbar	13
3	Quickstart	17
3.1	Writing a L ^A T _E X Document with Kile for Beginners	17
3.2	Environments	18
3.3	Using Kile	18
3.4	DVI Files	19
3.4.1	Viewing a DVI	19
3.4.2	Printing a DVI	19
3.4.3	Converting DVI files	19
3.5	Forward Search between Kile and Okular	19
3.6	Inverse Search between Kile and Okular	19
3.7	Resolving Errors	20

4	Starting a New Document	21
4.1	Templates	21
4.1.1	Create a New Template	21
4.1.2	Configuring Automatic Substitutions	22
4.1.3	Create a Template from the Wizard	22
4.1.4	Creating a Template from any File	22
4.1.5	Removing a Template	22
5	Editing L^AT_EX Documents	23
5.1	The L ^A T _E X Reference	23
5.2	Cursor Movements	23
5.3	Brackets	24
5.4	Highlighting	24
5.5	Bullets	24
5.6	Select	24
5.6.1	Select L ^A T _E X commands	25
5.7	Delete	26
5.8	Environment	26
5.9	T _E X Group	27
5.10	Double Quotes	28
5.11	Smart Newline	28
5.12	Smart Tabulator	29
6	Code Completion	30
6.1	Automatic Environment Completion	30
6.2	L ^A T _E X Commands	30
6.3	Environments	32
6.4	Abbreviations	32
6.4.1	Abbreviations	33
6.5	Autocompletion Modes	33
6.5.1	L ^A T _E X Commands	33
6.5.2	Document Words	34
6.6	Writing Own Completion Files	34
7	Wizards and Dialogs	35
7.1	QuickStart Wizard	35
7.2	Include Graphics	35
7.3	Arrays and tabulars	37
7.4	Inserting floating elements	38
7.5	Inserting Math environments	39
7.6	PostScript [®] Utilities	39
7.7	PDF Utilities	43
7.7.1	Rearrangements	43
7.7.2	Properties	47
7.7.3	Permissions	48
7.8	Document Statistics	49

8	Special Tags in L^AT_EX	50
8.1	Using the L ^A T _E X Tag Library	50
8.2	Using Bibitems	52
9	User-Configurable Menu	54
9.1	Configuration	54
9.2	Wizard	55
9.3	Placeholders	59
9.3.1	Insert Text	59
9.3.2	Insert File Contents	61
9.3.3	Execute A Program	61
9.4	Parameter	61
9.5	Menu Definition Files	63
10	The Build Tools	64
10.1	Compiling, converting and viewing	64
10.1.1	BibT _E X	64
10.1.2	MetaPost and Asymptote	64
10.1.3	PDFL ^A T _E X	64
10.1.4	L ^A T _E X to Web	65
10.1.5	Passing Command Line Parameters	65
10.2	Quick Preview	65
10.2.1	Selection Mode	66
10.2.2	Environment Mode	66
10.2.3	Subdocument Mode	67
10.2.4	Mathgroup Mode	67
10.2.5	Quick Preview in Bottom Bar	67
10.3	Graphic File Formats	67
10.3.1	L ^A T _E X and PDFL ^A T _E X	67
10.3.2	Graphics Conversion	67
10.3.3	Use the right File for the right Graphic	68
10.4	EPS Graphics	68
10.4.1	L ^A T _E X and EPS Graphics	69
10.4.2	The PostScript [®] Way of Kile	69
10.4.3	The PostScript [®] Way and Bitmap Graphics	69
10.4.4	PDFL ^A T _E X and EPS Graphics	70
10.5	Master Document	71
10.6	Error Handling	71
10.7	The Watch File Mode	72

11 Navigating the L^AT_EX Source	73
11.1 Using the Structure View	73
11.1.1 Using the Context Menu	73
11.1.2 Updating the Structure View	75
11.2 Bookmarks	75
12 Projects	76
12.1 Working with Projects	76
12.2 Creating a Project	76
12.3 The Files and Projects View	77
12.4 Adding and Removing Files	78
12.4.1 Archiving your Project	79
12.5 Project Options	79
12.6 Closing a Project	80
13 Document Encoding	81
13.1 The ucs Package	82
13.2 XeLaTeX	82
13.3 CJK Support	82
13.3.1 CJK Troubleshooting	83
13.3.2 How do I input CJK in Unicode?	84
14 Scripting	85
14.1 Scripting in Kile	85
14.2 Executing a Script	86
14.3 API Reference	87
14.3.1 Global Functions	88
14.3.2 The Cursor Prototype	88
14.3.3 The Range Prototype	89
14.3.4 The View API	91
14.3.5 The Document API	93
14.3.6 The Kile API	103
14.3.6.1 Alert	103
14.3.6.2 Input	103
14.3.6.3 Wizard	104
14.3.6.4 Script	105
14.3.6.5 File	105
14.4 Examples	106
14.4.1 Example 1: replace environment name	106
14.4.2 Example 2: replace a L ^A T _E X font command	106
14.4.3 Example 3: surround selected text	107

The Kile Handbook

15 Help	109
15.1 Help Documents	109
15.2 Context Sensitive Help	109
15.3 Searching for Keywords	110
15.4 User Defined Help	111
16 Credits and License	114

Abstract

Kile is a T_EX and L^AT_EX source editor and shell.

Chapter 1

Preface

1.1 Requirements

To run Kile, you need to have the following components installed on your system:

- **K Desktop environment (KDE):** KDE is a popular open-source desktop environment.
- **Qt:** Qt™ is a C++ GUI and network library needed to compile Kile.
- **LATEX:** high-quality document typesetting program. Most likely you want the TeX Live (or on older systems the teTeX) package, if you are on a UNIX®-like system.

Most of these items might be included in your Linux® distribution; please refer to your distribution documentation, or refer to your installation CD or DVD, for adding these packages to your computer.

Kile might also be available as a pre-compiled package for your Linux® distribution already. Please check with the package manager of your distribution.

1.2 Intended Audience

This manual is intended for any individual, regardless of her or his experience with L^AT_EX, KDE, Kile or Linux®.

Advanced users are not likely to read this manual, but all suggestions on documentation will be considered. If you would like to contribute to this project or the documentation, please consult the [Kile web page](#).

Do you need answers about Kile? Are you stuck with the compilation process? Do you want to see a new feature implemented? The preferred way to ask technical questions or to start a discussion is to use our mailing list: kile-devel@lists.sourceforge.net.

Chapter 2

Introduction

2.1 Basic facts

2.1.1 About Kile

Kile is an integrated L^AT_EX environment for the KDE desktop. Kile gives you the ability to use all the functionality of L^AT_EX in a graphical interface, giving you easy, immediate, and customized access to all programs for L^AT_EX code-completion, compiling, postprocessing, debugging, conversion and viewing tools; you also get very handy wizards, a L^AT_EX reference and a powerful project manager.

2.1.2 Kile and the Kate Editor Component

Kile is based on the Kate editor component, i.e. a lot of its editing capabilities stem from the Kate editor component itself. Kile extends these capabilities with features to edit L^AT_EX documents. To learn more about the Kate editor component and its capabilities, see the [Kate webpage](#).

2.1.3 What is L^AT_EX?

L^AT_EX is a text-processing system derived from T_EX, a program developed originally in 1977 by Donald Knuth to help layout text in a professional way and obtain a layout quality that is on a par with the work of a professional typesetter. L^AT_EX was created by Leslie Lamport to give authors an automatic typesetter, especially to ease the expensive and painstaking process of typesetting of mathematical formulas and expressions, which are enclosed within dollar signs in L^AT_EX *for a reason*. Today, word-processing programs let any user act as typesetter, but what is often needed is a document that simply looks good without having to spend hours to bring it into shape. L^AT_EX takes that burden on its shoulders, and lets you concentrate on the document instead of on the layout. And yes, it *will* look good!

2.1.4 How do you pronounce it? Why that strange typesetting?

There is a funny tradition of T_EX-related packages to have the strangest pronunciation and typesetting possible. T_EX was supposed to be brought in from the Greek τεχ, in Latin letters *tech*. There are a lot of explanations why, but most likely it is because T_EX was originally conceived for technical reports, and indeed its foremost ability was the correct and easy typesetting of mathematical formulae, then an extremely expensive, time-consuming and frustrating business.

The pronunciation is supposed to be as follows: *T* as you would expect, *E* as in *get*, and *X* as in the German *ich*. If you do not know what *ch* sounds like, it is more or less like the sound a hissing cat produces; the IPA symbol is /ç/. Many people report a different pronunciation of *ach* (IPA symbol /x/), but according to some Greeks, the first version is indeed correct. You should be aware that a lot of people mispronounce T_EX as /teks/ or /tek/.

Last, in L^AT_EX the first L^A is pronounced as *lay*: the idea being, while raw T_EX is difficult, even a *layman* can use L^AT_EX macros. A less inspiring, but more realistic explanation is that it stems from the surname of Leslie Lamport, the creator of L^AT_EX. Now you know!

2.2 L^AT_EX 101

The L^AT_EX typesetting system is similar to other markup languages such as XML, which is used in many types of documents (including the one you are reading), or HTML, which is used for web pages. The general idea behind markup languages is to have special keywords, called *tags*, that tell a program (a word processor, a web browser, or the L^AT_EX compiler) how the text enclosed within the tags is to be interpreted. Kile offers a number of such tags in the **LaTeX** menu in the menu bar.

While we will try to give you a good idea of what L^AT_EX is, this document is, of course, not The Definitive Book on L^AT_EX. If you want to learn L^AT_EX in depth, you may want to borrow a specialized book from your local library.

As with any other markup language, L^AT_EX documents contain a *preamble*, which defines global properties, such as paper size, page numbering, dimensions of the text on the page, and a document *body*, which contains the text of the document. The preamble is composed at least of the `\documentclass` command. It precedes the document body, which starts with the command `\begin{document}` and is concluded with the command `\end{document}`.

2.3 Kile's Main Features

2.3.1 QuickStart Wizard

The QuickStart wizard built into Kile is a useful feature to quickly start creating documents in Kile. Choosing the wizard from the menubar gives you several choices for the creation of your document. You can also specify some options related to the document right away.

Class options:

- **Document Class:** choose the type of document you want to create: article, book, letter, report, scrartcl, screprt, scrbook, prosper, beamer or other custom-defined.
- **Typeface Size:** tell Kile what point size (pt) you want to use.
- **Paper Size:** choose the size or style of sheets.
- **Encoding:** In general it is a good idea to use your system's standard encoding. Modern systems now move more and more to UTF-8 as the standard encoding. If you can, use utf8 or utf8x (which is indeed the correct spelling for L^AT_EX documents).
- **Other options:** this allows you to set further options such as printing, draft, and others.

Packages

This lists some of the most common additional packages used in L^AT_EX. Select the check box to include it.

Document Properties:

- **Author:** put your name here.
- **Title:** add the document title here.
- **Date:** specify the date.

2.3.2 Predefined Templates

The predefined templates in Kile are:

- Empty document: real freaks start from scratch!
- Article: sets the article format, for a document short enough not to be broken down to chapters.
- Report: sets the report format, for a middle-sized document, with, for example, page numbering on the page's outer edge.
- Book: sets the book format, a full-fledged flavor, so powerful that it is used to write many university textbooks.
- Letter: sets the letter format.
- Beamer, HA-Prosper: create nice presentations in PDF with a superior look and all L^AT_EX power.
- Powerdot: Powerdot is the follower of the packages **seminar** and **HA-Prosper**. It does not have as many options as Beamer, but it is easy to use and it can create really nice presentations in PDF.
- Scartcl, Scrbook, Scrpert, Scrltr2: the KOMA-Script document classes, especially adapted to German typography. Use them whenever you write German texts.
- Xelatex: a modified **Article** template to use with **XeLaTeX**.

Note that all of these templates can be adjusted to the user's requirements.

New users need not worry: this list is just a brief description of the available features, and a more detailed description can be found in chapter 3.

2.3.3 Syntax Highlighting

Kile is similar to other programs that deal with source code and editing, and will automatically highlight commands, options and items that are used (and abused). Kile makes it possible to easily spot problematic areas: for example, if you see major areas of text turn green, it is likely that you forgot to close a math environment somewhere.

2.3.4 Auto-Completion of Environments

The auto-completion of environments means that, when you begin a new environment by typing `\begin{environment}`, Kile will automatically insert a matching `\end{environment}` command, with a line in between them for your text. You can of course deactivate it if you want in **Settings** → **Configure Kile...** → **LaTeX+Environments**.

2.3.5 Jump to Structure Element

All documents are normally structured in a hierarchy of some type. L^AT_EX allows you to break up documents into the following hierarchy (part being highest in the hierarchy, and subparagraph being lowest):

- `\part`
- `\chapter`
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`

When viewing a document in the **Structure** view, you can jump between elements by clicking on the element you would like to view.

2.3.6 Inverse Search

When creating your own L^AT_EX files, inverse search can be very helpful. Once you have created a DVI file (DeVice Independent File) or PDF file, you can click the left mouse button while pressing **Shift** in the viewer and Kile will jump to the corresponding line in the L^AT_EX source code.

A DVI is a type of file containing a description of a formatted document, along with other information including character font, and is besides PDF the usual output of T_EX or L^AT_EX. A number of utilities exist to view, convert and print DVI files on various systems and devices.

2.3.7 Forward Search

When using inverse search, the selection of items in the DVI or PDF file is associated with the editor, so when you click on the DVI or PDF file, the main window jumps to the corresponding section of L^AT_EX code in the editor. Forward search is the exact opposite of this. Forward search will allow you to click on a specific section of text in the L^AT_EX code, and jump to the associated position in the viewer window.

2.4 The Toolbar

- **New:** begin a new document.
- **Open:** open a new document.
- **Close:** close your document.
- **Define document as master:** this is used when working with multiple files. Having a master document will let you work more easily with other `.tex` files included in your document. If you are using projects, you can also set in **Project** → **Project Options** a project-wide master document.
- **Quickbuild:** compiles your L^AT_EX source code and displays the results automatically unless there are errors contained in the document.

The Kile Handbook

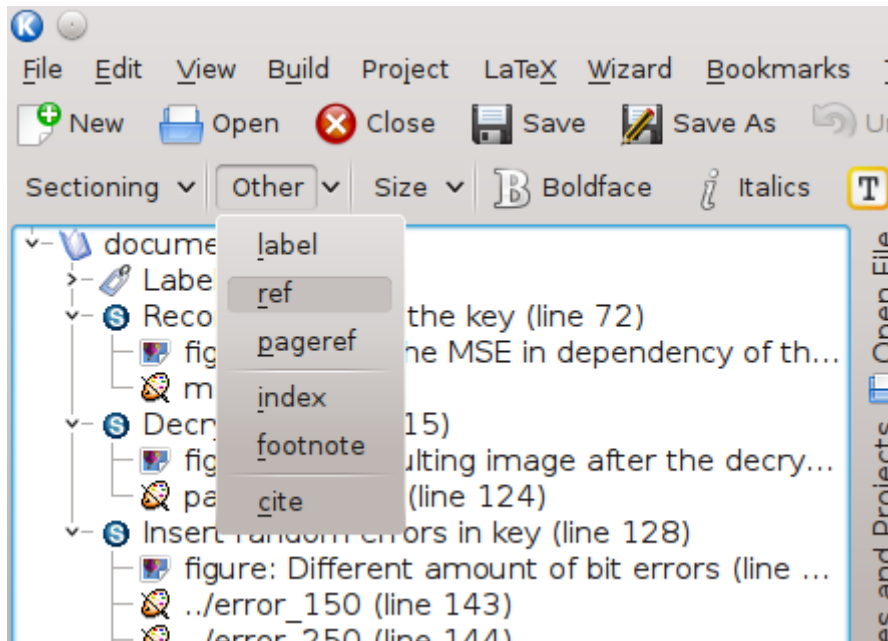
- **Watch file mode:** this mode will “watch” the DVI file for changes, and will not launch a new session of Okular after **Quickbuild**.
- **View logfile:** views the `.log` file, so you can spot errors.
- **Previous error:** jumps backward through the `.log` file and highlights errors in the source.
- **Next error:** jumps forward through the `.log` file and highlights errors in the source.
- **Stop:** halts current tool.
- **LaTeX:** runs $L^A T_E X$ on the active document.
- **ViewDVI:** launches DVI viewer.
- **DVItO[®]PS:** converts a DVI to a PostScript[®] (PS).
- **ViewPS:** launches PostScript[®] (PS) viewer.
- **PDFLaTeX:** runs $P D F L^A T_E X$ on the active document.
- **ViewPDF:** launches the PDF viewer.
- **DVItO[®]PDF:** converts a DVI to a PDF.
- **PStoPDF:** converts a PS to a PDF.
- **ViewHTML:** views HTML created.
- **ForwardDVI:** jump to the page of the DVI file that corresponds to the current line in the editor.
- **ForwardPDF:** jump to the page of the PDF file that corresponds to the current line in the editor.

If you look at the **Edit** toolbar, you will notice three large drop-down menus. The drop-down menus were designed for you to be able to quickly add certain common features into your document. The first drop down box is used for quickly dividing your document by parts, chapter, sections and so on; the available commands to add segments to your $L^A T_E X$ source code are:

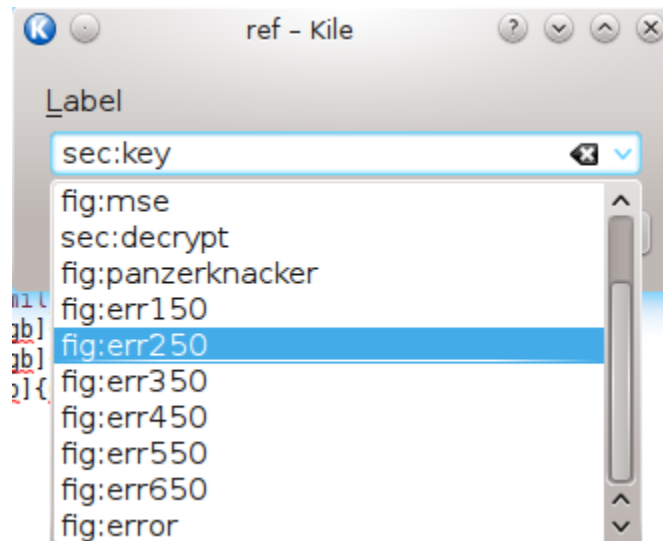
- **part:** highest level of sectioning for a document.
- **chapter:** starts a new chapter.
- **section:** create a new section.
- **subsection:** create a new subsection.
- **subsubsection:** a secondary section between subsection and paragraph.
- **paragraph:** create a new paragraph.
- **subparagraph:** create a new subparagraph.

The drop down box named **Other** is used to insert items into your document such as indexes, footnotes, and references; the available commands are:

- **label:** a command that produces a label for a chapter, a figure or another element.
- **index:** creates an entry for the index.
- **footnote:** creates a footnote in your document.
- **ref:** used to refer to a predefined label, which you can choose from a drop-down list.
- **pageref:** just like **ref**, but refers to a page instead of a structure element.
- **cite:** create a reference with data from a bibliography.



The *Other* drop-down menu



Selecting the label for a reference

When using `cite`, you are presented with a drop-down list of bibitems, but if you are using Bib_TE_X this will only work if the file belongs to a Project. For editing Bib_TE_X files the usage of specialized editors is recommended. The author has had good results with KBib_TE_X. Of course you can also write the Bib_TE_X files by hand inside Kile.

The last drop down box labeled `tiny` is used to set the size of the text. You can set the size of the main text, of footnotes, and so on. The available commands are:

- `tiny`: smallest.
- `scriptsize`: very small.
- `footnotesize`: smaller.
- `small`: small.

The Kile Handbook

- **normalsize**: normal.
- **large**: large.
- **Large**: larger.
- **LARGE**: even larger.
- **huge**: still larger.
- **Huge**: largest.

Chapter 3

Quickstart

3.1 Writing a L^AT_EX Document with Kile for Beginners

Users of Kile have two choices when starting a new document: they can use the **Wizard** to begin a new document, select the type of document they would like to create and options such as font size, paper size, and so on; otherwise, they can write the code by hand.

```
\documentclass[12pt]{article}
\begin{document}
  Here is a bunch of text coded in \LaTeX.
\end{document}
```

Every document in L^AT_EX begins with the command `\documentclass[optional argument]{class}`, where class specifies the document type.

Typing in the code example above from the text box gives you the following output:

Here is a bunch of text coded in L^AT_EX.

Compiled text in DVI output

The brackets that come after the command `\documentclass` contain the options for the command. The option `[12pt]` sets the size of the font for your article; if you do not set the font size in the beginning, you can set it later in the text.

Once you have typed in the code example from the box above, you will need to compile your L^AT_EX source code. The easiest way for you to compile L^AT_EX is to use the **Build** menu, or using the **Quickbuild** button.

Alt-2 is the keyboard shortcut to compile your source code.

You have to save your source code before you can compile; Kile will do this automatically for you.

If your document did not compile, check the log for errors. When using the **Quickbuild** key, the Okular viewer should be launched automatically; if it does not, look at the log.

3.2 Environments

An environment is a segment of text that is managed differently from the rest of the document. For example, you create a report with font size 12, but you need to change your font size for a few sentences. The commands `\begin{environment}`, `\huge` and `\end{environment}` will let you temporarily alter the text inside the environment commands to be size huge.

Changes are only effective from `\begin{environment}` to `\end{environment}`. There are no limits as to how many changes you can make inside an environment.

There are many features you can add to your document that will make it more readable and user-friendly. You can add features such as specific fonts, bold, italics, underline etc. to your document, and these commands will end with either an `\end` command, or at the end of your environment.

- `\begin{emph}`: this command makes text italicized, and is valid until the code comes across a `\end{emph}`, or another environment. To italicize one word in a sentence, you can use the syntax: this is `\emph{my}` sentence.
- `\textbf{I am making this text inside the brackets bold}`: this command makes your text bold.
- `\quote`: to create a quote inside your document; begin your quote with `\begin{quote}` and end it with `\end{quote}`.
- `\center`: centers the text.
- `\verse`: creates offset text for poems.
- `\itemize`: makes an itemized list.

3.3 Using Kile

Now that we have given you some background about how to write code using the L^AT_EX markup language, we will show you how to create a document using Kile step-by-step.

1. Start Kile.
2. Select **Wizard** → **Quick Start**, then choose a format, and set your preferences in the wizard.
3. Once the wizard has entered text, do some customization to make the document more readable, add a minimum of one quote, some bold text, italics, and a verse to see the difference between the commands.
4. Save your file, and give it the name `intro.tex`.
5. Build your document using **Alt-2**, or the button labeled **LaTeX**.
6. Select **ViewDVI**.
7. Check out all your new text.
8. When you are done viewing your document, click the **Editor View** button or press **Ctrl-E** to return to the editor if you are using the embedded viewer, or close the viewer window if you are using a separate viewer.

That's it! You have just created your first L^AT_EX document!

Once you have created your DVI, you will be able to print your document, or change it into a PostScript[®] or PDF file if you want. Experiment and have fun!

3.4 DVI Files

DVI stands for *DeVice Independent* file. These files are produced by T_EX or L^AT_EX to be read by a driver of some sort on your computer. There are many different types of output that a `.dvi` can be sent to, such as a printer, PostScript[®] or PDF file converter, or your computer screen.

3.4.1 Viewing a DVI

You have already seen how to view a DVI file on the screen by using the **ViewDVI** button in the toolbar.

3.4.2 Printing a DVI

To print a DVI, you can use the same process that you used to create your document earlier (see Section 3.3). At step 7, after clicking **ViewDVI**, select **File** → **Print** in the viewer, and if you have your printer properly configured, you will be able to print the DVI.

3.4.3 Converting DVI files

The toolbar gives the options of Converting a DVI to other formats. Once you have created a DVI from your L^AT_EX source code, you will be able to export it to a format of your choice using the toolbar buttons.

3.5 Forward Search between Kile and Okular

The forward search functions allow you to jump from your editor directly to the associated position of the DVI or PDF file.

Kile offers a configuration with this option for all L^AT_EX binaries. Go to **Settings** → **Configure Kile...** → **Tools+Build** and always choose the **Modern** configuration.

To execute a forward search, position the cursor on a line of source code, and click with the left mouse button while pressing **Shift**, or click **Forward DVI/Forward PDF** to jump to the associated position in the DVI or PDF viewer window.

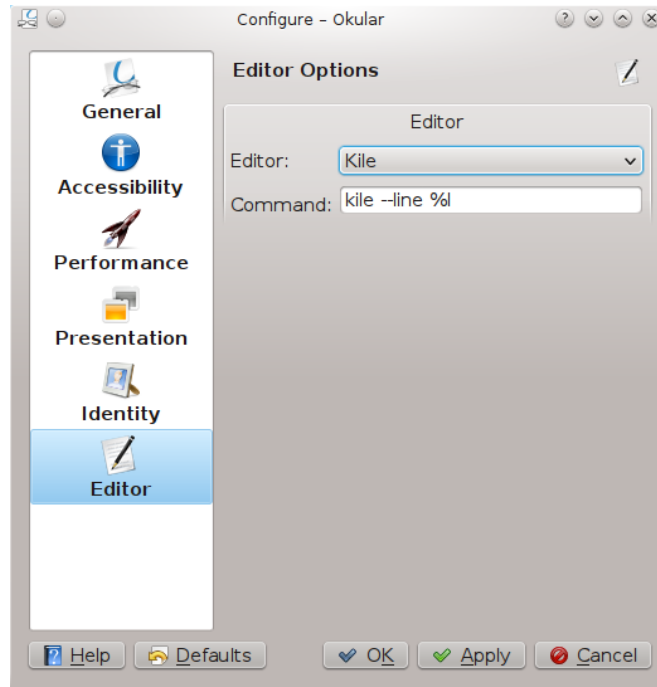
3.6 Inverse Search between Kile and Okular

Inverse search is a very useful feature when you are writing a L^AT_EX document yourself. If everything is set up properly, you can click into Okular's window with the left mouse button while pressing **Shift**. After that Kile loads the L^AT_EX source file and jumps to the proper paragraph. To use inverse search, you have to compile your L^AT_EX file with the **Modern** configuration.

Inverse search cannot work unless:

- The source file has been compiled successfully.
- Okular knows which editor you would like to use.

With this feature of Okular, a left mouse click while pressing **Shift** in the DVI or PDF document will result in Kile opening the corresponding L^AT_EX document and attempt to go to the corresponding line. Remember to tell Okular to use Kile as a text editor, in Okular's menu item **Settings** → **Configure Okular...** (on the page **Editor**).



Configuring Okular

3.7 Resolving Errors

If you are trying to use quickbuild, and the DVI viewer does not open, chances are you have an error. If you have an error, it will be visible in the log file / message area, and the summary of the error will be given.

The log file will explain the source of the error in your code. In the editor, you can use the buttons in the toolbar labeled **Previous LaTeX Error** and **Next LaTeX Error** to jump to and from errors. The log file always states in which line the error occurred. To view the line where an error occurred, click on the error in the log window, and Kile will take you to the error's line.

Chapter 4

Starting a New Document

When you click the button in the toolbar to begin a new document a dialog appears, asking which type of template you would like to use to write your document. The default choices are:

- Empty document
- Article
- Beamer
- Book
- HA-Prosper
- Powerdot
- Letter
- Report
- Scartcl (from the KOMA-Script package)
- Scrbook (from the KOMA-Script package)
- Scltr2 (from the KOMA-Script package)
- Scrrprt (from the KOMA-Script package)
- PDF
- XeLaTeX

If you selected an **Empty document**, you can either start writing a document from scratch, or you can use the wizard to quickly start a new document (see Section 2.3.1).

4.1 Templates

Frequent users of L^AT_EX typically use the same preamble for almost every document they use. Templates can be created, saved and loaded within Kile to make it easier to start a new document.

4.1.1 Create a New Template

To create a new template, you must first either open a T_EX / L^AT_EX file, or create a file of your own. Kile can generate a template from an existing document by opening the desired document and selecting **File** → **Create Template from Document**.

4.1.2 Configuring Automatic Substitutions

When creating a new document by selecting a template from **File** → **New**, certain character combinations will be replaced by data such as your name, or the character encoding you are using. These variables can be configured in **Settings** → **Configure Kile...** → **Settings+General**.

When designing your own template, it is useful to know which character combinations are replaced by which template variables:

- **\$\$AUTHOR\$\$**: This string will be replaced by the author variable.
- **\$\$DOCUMENTCLASSOPTIONS\$\$**: This string will be replaced by the documentclass options variable. Typically this is used as follows: `\documentclass[$$DOCUMENTCLASSOPTIONS$]{article}`.
- **\$\$INPUTENCODING\$\$**: If the inputencoding variable is set to, say, `latin1` this string is replaced by `\input[latin1]{inputenc}`.

4.1.3 Create a Template from the Wizard

The easiest way to create a new template is to start the wizard, and then add commands in the editor. Once you have your document set up the way you like:

1. Save your file;
2. Go to **File**;
3. Choose **Create Template from Document**;
4. Make any corrections necessary to the template;
5. Enter a name for your new template;
6. Click **OK** to add your template to the menu.

Next time you start up a new document, you will be able to choose your customized template instead of the default ones.

4.1.4 Creating a Template from any File

A template can be created from any L^AT_EX file. If you are looking for an easy way to configure a template, go find one you like on the Internet and follow the same steps as listed in Section 4.1.3.

For instance, you may want to create a full-fledged A0 poster. These posters are usually seen at scientific conferences, and L^AT_EX will help you make an attractive, catchy poster. Remember that you will need the `a0poster` package, which is normally not included in standard T_EX distributions. Download it from [here](#) and place it in the same folder as your L^AT_EX file.

4.1.5 Removing a Template

To remove a template from Kile, do as follows:

1. Go to **File** → **Remove Template...**;
2. A dialog box will appear with all templates listed: select a template;
3. Click **OK**, and your template will be removed.

Templates marked with an asterisk (*) cannot be removed without the proper permission.

Chapter 5

Editing L^AT_EX Documents

The internal editor that Kile uses is Kate. Kate is a text editor created for programmers, which incorporates the ability to read and highlight many different types of text files, among which are L^AT_EX and BibT_EX; you can access many options for Kate directly from Kile's **Tools** menu.

To learn more about Kate and its capabilities, see the [Kate Handbook](#). Kile users can start reading from the chapter 'Working with the Kate Editor'.

5.1 The L^AT_EX Reference

Kile features a very practical L^AT_EX tag reference, which you can access by choosing **Help** → **LaTeX Reference**. It contains a thorough description of almost all the commands that you may use in L^AT_EX and their syntax.

5.2 Cursor Movements

To select text, you have the following options:

- Hold left mouse button, and drag mouse to highlight text.
- Click once on a word to move the cursor to a new area.
- Click twice on a word to select the whole word.
- Click twice on a word and pressing **Ctrl** to select the whole T_EX word. This means clicking in this way on `\par` from `\par\bigskip` only select `\par`.
- Click three times to select the whole sentence.

Holding the left mouse button, and dragging the text you want to select, automatically copies the selected text to the clipboard.

Holding **Shift** and using the arrow keys allows you to select portions of the source code in the editor window.

5.3 Brackets

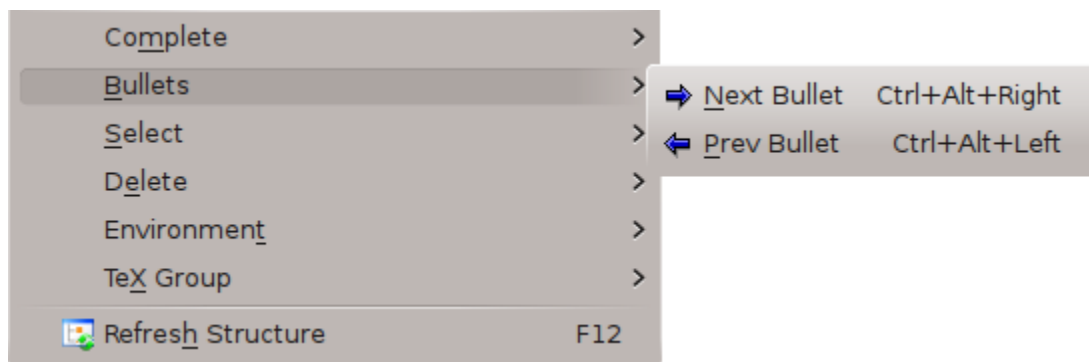
Bracket completion is a visual tool that the editor view uses to indicate to you which bracket matches which. If you open any `.tex` file, and select any bracket, whether it be a parenthesis `()`, square brackets `[]` or braces `{}`, the editor will highlight the bracket and its match in yellow (this default color can be changed). So, for example, if you position the cursor on the braces in `\section{Introduction}`, you would see `\section{Introduction}` in the default yellow highlight, showing you the location of the beginning and ending brackets.

5.4 Highlighting

Kile has the ability to look for and highlight different types of code. For example, $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ commands are distinguished from normal text, and math formulas are also highlighted in a different color.

5.5 Bullets

Many wizards can insert optional bullets, a special kind of bookmark within the text. The menu entries **Edit** → **Bullets** or the corresponding keyboard shortcuts will allow you to jump to the next or last bullet. This will also highlight this bullet so that it will be deleted automatically, when you enter your first letter.



Next Bullet (Ctrl+Alt+Right)

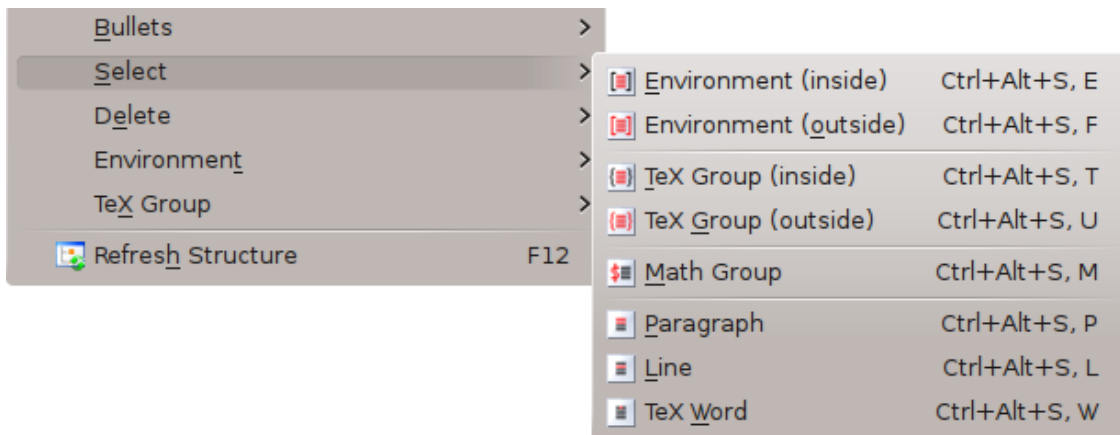
Jump to the next bullet in the text if there is one.

Last Bullet (Ctrl+Alt+Left)

Jump to the previous bullet in the text if there is one.

5.6 Select

Editing is of course one of the main aspects when you use a program like Kile. Although Kate already has great capabilities, Kile adds some important features, which are especially needed to write $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ source. $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ always needs a lot of environments and groups, so Kile supports very special commands to select them. Under **Edit** → **Select** you will find the following commands to select text.

**Environment (inside) (Ctrl+Alt+S,E)**

Select an environment without the surrounding tags. If this command is called, when an environment is already selected, Kile will expand the selection to the next surrounding environment.

Environment (outside) (Ctrl+Alt+S,F)

Select an environment including the surrounding tags. This selection can also be expanded with a second call of this command.

TeX Group (inside) (Ctrl+Alt+S,T)

Select a TeX group inside the surrounding braces.

TeX Group (outside) (Ctrl+Alt+S,U)

Select a TeX group including the surrounding braces.

Math Group (Ctrl+Alt+S,M)

Select the current math group including the math commands.

Paragraph (Ctrl+Alt+S,P)

Select a whole paragraph, i.e. a group of text lines separated on both sides by empty lines. A paragraph does not mean just continuous lines of text, as it is in other text editors. This extended meaning also includes tables, L^AT_EX commands and all other lines of source. The only important thing for Kile is that this kind of paragraph is separated by two empty lines.

Line (Ctrl+Alt+S,L)

Select the text line of the current cursor position.

TeX Word (Ctrl+Alt+S,W)

Select the word under the current cursor position. This selection has also an extended meaning, because this command can also select L^AT_EX commands, which begin with a backslash and may also have an optional star at the end.

5.6.1 Select L^AT_EX commands

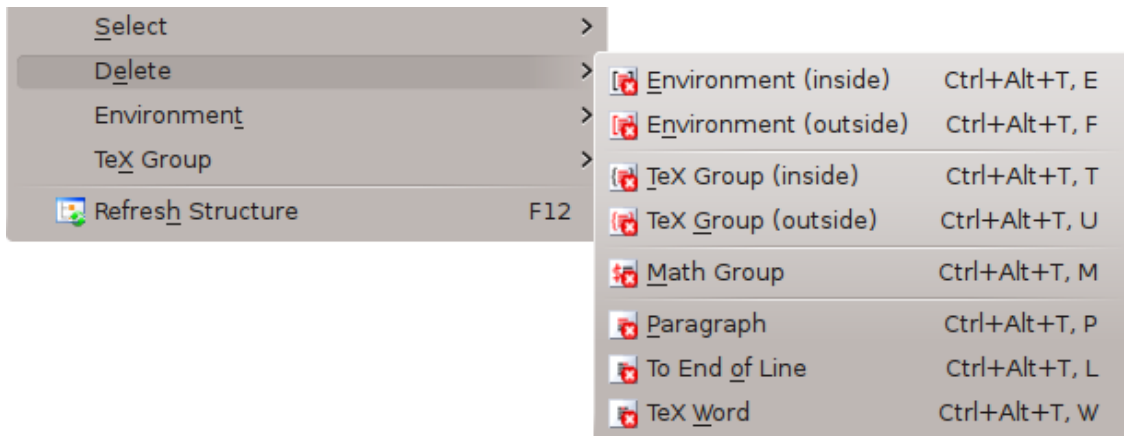
Kile has an extended feature to select L^AT_EX commands. If you for example have typed

```
text \bfseries\itshape more text
```

and double click on one of the L^AT_EX commands, both will be selected. But often you only want to select one of two or more commands. This can be done using the **Ctrl** key. You only have to press the **Ctrl** key and a double click will only select the desired command.

5.7 Delete

To delete some parts of a document you can of course select them, and then use the **Delete** key. Kate also offers the command **Ctrl-K** which deletes the whole line. But Kile offers a faster way with its own delete commands. Under **Edit** → **Delete** you will find the following commands to delete text.



Environment (inside) (Ctrl+Alt+T,E)

Delete an environment without the surrounding tags.

Environment (outside) (Ctrl+Alt+T,F)

Delete an environment including the surrounding tags.

TeX Group (inside) (Ctrl+Alt+T,T)

Delete a T_EX group inside the surrounding braces.

TeX Group (outside) (Ctrl+Alt+T,U)

Delete a T_EX group including the surrounding braces.

Math Group (Ctrl+Alt+T,M)

Delete the current math group including the math commands.

Paragraph (Ctrl+Alt+T,P)

Delete a whole paragraph. Look at the **Select** → **Paragraph** command, how a paragraph is defined in Kile.

To End of Line (Ctrl+Alt+T,I)

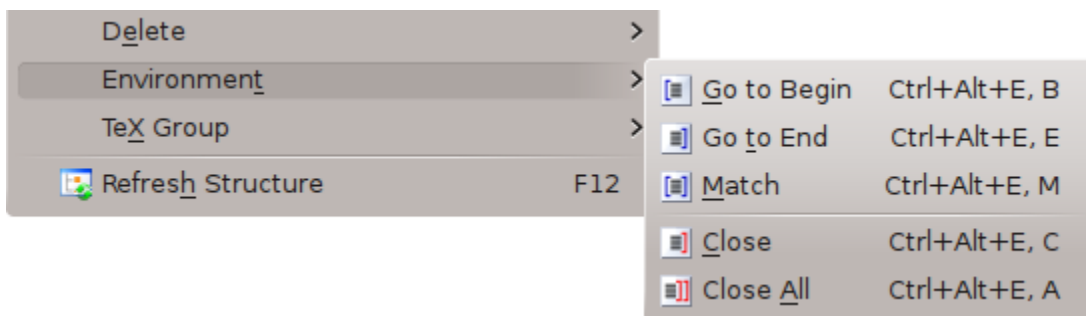
Delete the text from the current cursor position to the end of the line.

TeX Word (Ctrl+Alt+T,W)

Delete the word or L^AT_EX command under the current cursor position.

5.8 Environment

It has already been mentioned that environments are a central point in L^AT_EX. So Kile offers five other commands to make the work with L^AT_EX as easy as possible under submenus **Edit** → **Environment**.

**Go to Begin (Ctrl+Alt+E,B)**

This command will jump to the beginning of the current environment, wherever your current position is. The cursor will be placed directly in front of the opening environment tag.

Go to End (Ctrl+Alt+E,E)

This command will jump to the end of the current environment, wherever your current position is. The cursor will be placed directly behind the closing environment tag.

Match (Ctrl+Alt+E,M)

When your cursor is placed in front of or above the `\begin{environment}` tag, it will be moved to the opposite end of the environment and vice versa.

Close (Ctrl+Alt+E,C)

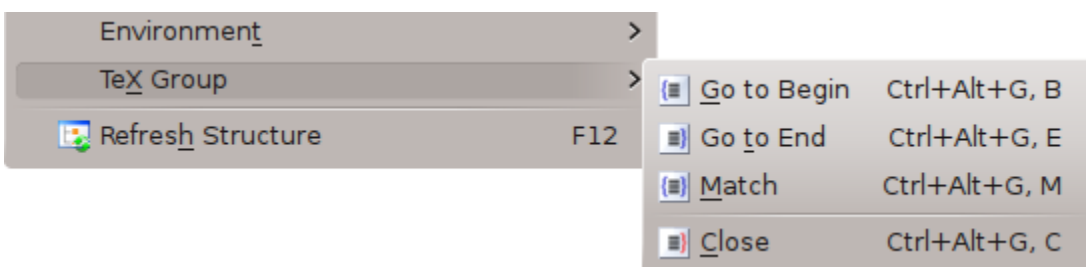
Typing a lot of nested environment tags, you may lose control of all those environments. This command will close the last opened environment, so that the nested structure of environments will not be broken.

Close All (Ctrl+Alt+E,A)

This closes all open environments, not only the last opened environment.

5.9 T_EX Group

Kile also offers some special commands for L^AT_EX groups, which are determined by braces `{ . . . }`. In submenu **Edit** → **TeX Group** you will find some important commands, which correspond to those from **Edit** → **Environment**.

**Go to Begin (Ctrl+Alt+G,B)**

This command will jump to the beginning of the current group, wherever your current position is. The cursor will be placed directly in front of the opening brace.

Go to End (Ctrl+Alt+G,E)

This command will jump to the end of the current group, wherever your current position is. The cursor will be placed directly behind the closing brace.

Match (Ctrl+Alt+G,M)

When your cursor is placed in front of or behind an opening brace of a T_EX group, it will be moved to the opposite end of the group and vice versa.

Close (Ctrl+Alt+G,C)

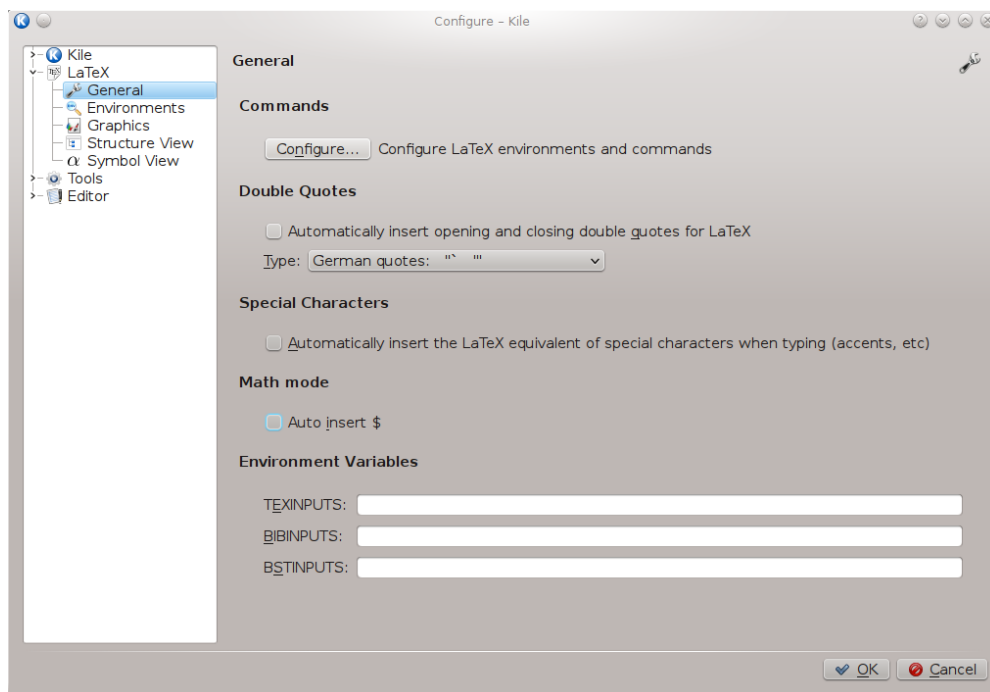
Typing a lot of nested group braces may be hard work. This command will close the last opened group, so that the nested structure of T_EX groups will not be broken.

5.10 Double Quotes

In L^AT_EX, two single quotes are used as double quotes. To help you insert these efficiently, Kile allows you to press " to insert two opening single quotes. Furthermore, if you want to close a quotation, you also have to press ". Kile will be smart enough to recognize this situation and inserts two closing quotes for L^AT_EX.

To get a literal double quote on the other side, press " twice.

You can enable or disable this auto insertion of opening and closing double quotes in section **Settings** → **Configure Kile...** → **LaTeX**.



If you also include language-specific options like **ngerman** or **french**, you will also be able to use German or French double quotes. Many more languages are available.

5.11 Smart Newline

If you press **Ctrl-Return**, Kile inserts an intelligent newline. If your current position is inside a list environment, like **enumerate** or **itemize**, Kile will not only insert a newline, but also add a `\item` command.

If you are inside a tabular environment, Kile will finish the current line with `\\`, followed by the newline.

If you are inside a L^AT_EX comment, Kile will start the next line with a `%`.

Even better, Kile is smart enough to support predefined L^AT_EX and user defined environments, which can be added in section **Settings** → **Configure Kile...** → **LaTeX**.

5.12 Smart Tabulator

Some users like to arrange columns in tabular environments and put all ampersand characters **&** beneath each other. Kile tries to support this. If you press **Alt-Shift-&**, Kile will look for the next tab in the row above. Although this tab may not be the corresponding tab, Kile will add some spaces to adjust the column position with the current tab.

Chapter 6

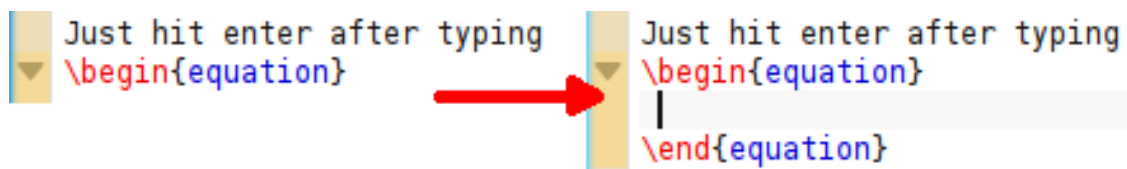
Code Completion

Although Kate already offers a good completion mode, Kile extends code completion to support some special methods especially for L^AT_EX. Five different modes are integrated. Three of them work on demand, the other two are autocompletion modes. All modes can be configured to work very differently at **Settings** → **Configure Kile...**

6.1 Automatic Environment Completion

When you begin a new environment, typing `\begin{environment}`, Kile will automatically add an `\end{environment}` command, with a line in between for your text.

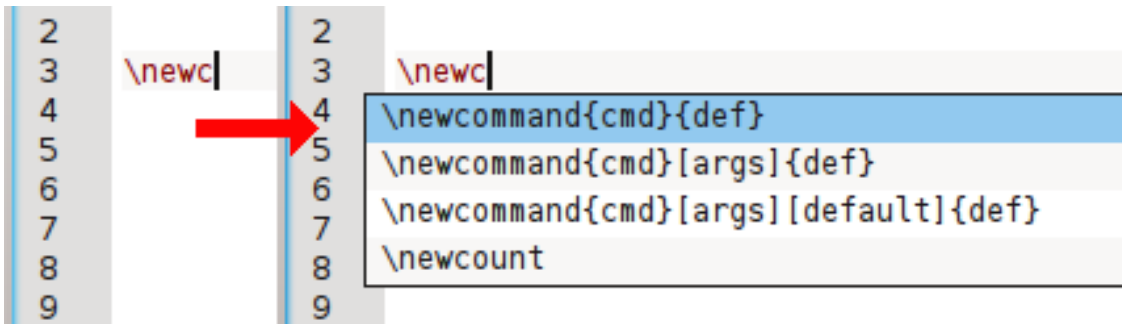
Autocompletion can be turned off in the L^AT_EX section of **Settings** → **Configure Kile...** → **LaTeX+Environments**.



Completing an Equation Environment

6.2 L^AT_EX Commands

When you type some letters, you can activate this completion mode for L^AT_EX commands and normal words with **Edit** → **Complete** → **(La)TeX Command** or the keyboard shortcut **Ctrl-Shift-Space**. Kile first reads the letters from the current cursor position to the left and stops at the first non-letter character or a backslash. If this pattern begins with a backslash, Kile will enter completion mode for T_EX or L^AT_EX commands. Otherwise it enters normal dictionary mode, where you will not find any L^AT_EX commands. Depending on the chosen mode, a completion box will be opened. You will see all commands or words whose beginning matches the current pattern. You can navigate with the cursor keys through this list and select one entry with **Enter** or a double click with the mouse.



When you push the **Backspace** key, the last letter of your pattern will be deleted, and the completion list may grow. On the other hand, if you type another letter it will expand the pattern and the visible word list may shrink.

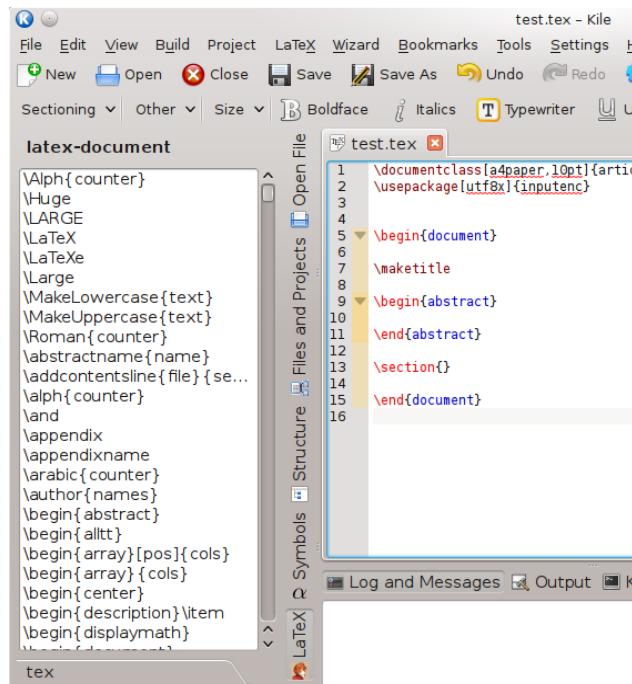
If you decide not to select any of the suggestions, you can leave this dialog with **Esc**.

You will see that all commands are written with a short description of their parameters. These descriptions are of course stripped when you select a command. Optionally you can let Kile insert bullets at these places, so that you can easily jump to these positions with **Edit** → **Bullets** → **Next Bullet** and insert the parameter you want.



Go to **Settings** → **Configure Kile...** → **Kile+Complete** to configure one or more of these lists. You can choose different word lists for T_EX and L^AT_EX commands and dictionary mode for normal words.

If you choose the option **Show Latex commands**, the entries of all chosen compressed word list (cwl) files for L^AT_EX command completion are shown in a separate view of Kile's sidebar. You will see which commands are available and what parameters and options must or can be given for a completion. You can also simply select one entry with a mouse click and it will be inserted into the document, with all named parameters and options stripped.



As each chosen word list will be shown in a separate view of its own, there could be too many views, so that Kile's main window may be larger than a small screen allows. As this looks very

ugly, Kile works with a maximum number of allowed views, which by default is set to 10. If this value is too big for your screen, you should reduce it.

6.3 Environments

The *command mode* is not useful for code completion of environments. You always have to type some letters of `\begin`, and invoking the completion mode will result in a huge list of environment tags. On the other hand, environments are so often used that Kile offers a special mode for code completion of environments. Forget the opening tag and write, for example, `a1`.

When you call the completion mode with **Edit** → **Complete** → **Environment** or keyboard shortcut **Alt-Shift-Space**, the opening tag is automatically added and you will see `\begin{a1}`. After this change, the completion list is much less cluttered.

```

17
18 a1|
19
20
21
22
23
24
25
26
27
28
29
30
31

```

Completion list:

- `\begin{a1|`
- `\begin{align}`
- `\begin{align*}`
- `\begin{alignat}{n}`
- `\begin{alignat}[alignment]{n}`
- `\begin{alignat*}{n}`
- `\begin{alignat*}[alignment]{n}`
- `\begin{aligned}`
- `\begin{aligned}[alignment]`
- `\begin{alignedat}`
- `\begin{alltt}`

Now select an environment, and you will see that it is also automatically closed. Even more, if Kile recognizes it as a list environment, it will also insert a first `\item` tag.

```

17
18 \begin{align}
19 |
20 \end{align}
21

```

Go to **Settings** → **Configure Kile...** → **Kile+Complete** to configure one or more of these lists. This mode uses the same word lists as the completion mode for $\text{T}_\text{E}\text{X}$ and $\text{L}^\text{A}\text{T}_\text{E}\text{X}$ commands.

6.4 Abbreviations

Kile supports user defined lists of abbreviations, which are replaced on demand by longer text strings. Look at **Settings** → **Configure Kile...** → **Kile+Complete** to configure one or more of these lists. For the example given here, the abbreviation list in `example.cwl` must be chosen. In this file you will find, for example, the entry `L=\LaTeX`.

For example, type only the letter **L**. Now invoke the abbreviation mode of word completion with **Edit** → **Complete** → **Abbreviation** or keyboard shortcut **Ctrl-Alt-Space**, and the letter **L** is replaced by the string `\LaTeX`.

Abbreviation completion also supports newline `%n` and `%C` to place the cursor, if these characters are present in the expansion text. So if you have the entry

```
enl=\begin{enumerate}%n\item %C%n\end{enumerate}%n
```

in the completion file, and invoke the abbreviation completion, the expansion looks as below, where **x** shows the final cursor position.

```
\begin{enumerate}
  \item x
\end{enumerate}
```

6.4.1 Abbreviations

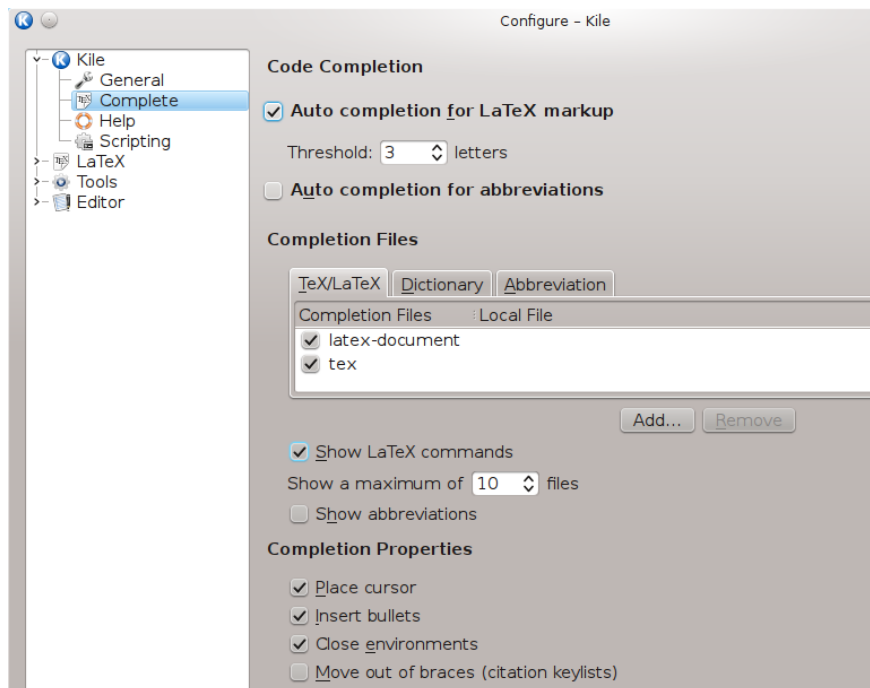
If you choose **Show abbreviations**, all possible abbreviations are shown in a view of the sidebar. So you will have a good survey of all possible abbreviations.

6.5 Autocompletion Modes

6.5.1 L^AT_EX Commands

You can also enable an autocompletion mode for L^AT_EX commands. When a given threshold of letters (default: 3) is entered, a popup window opens with a list of all matching L^AT_EX commands. You can select one of these commands, or ignore this window and type further letters. The entries of the completion box will always change and match your currently typed word.

Go to **Settings** → **Configure Kile...** → **Kile+Complete** to enable or disable this mode or to change the threshold.



6.5.2 Document Words

Large dictionaries are not useful in autocompletion mode. But, we have seen that a lot of words in a document are typed more than once. So Kile offers a completion for all words from the document that the user has already typed. You can manually invoke this completion, if you press **Ctrl-Space**. Note that this mode is different from the completion mode for L^AT_EX commands.

If you want to turn this mode on or off, go to **Settings** → **Configure Kile...** → **Editor** → **Editing+Auto Completion**. In this dialog you can configure if completion mode for document words should be enabled. There is also an additional autocompletion mode, where a completion box pops up, when a certain threshold is reached.

6.6 Writing Own Completion Files

The latest specification of the completion file format can found in the [CWL file format specification](#).

Completion files can be installed in a user's home folder under the `~/.kde/share/apps/kile/complete/<mode>/` subdirectory, where `<mode>` either stands for abbreviation, dictionary or tex.

Chapter 7

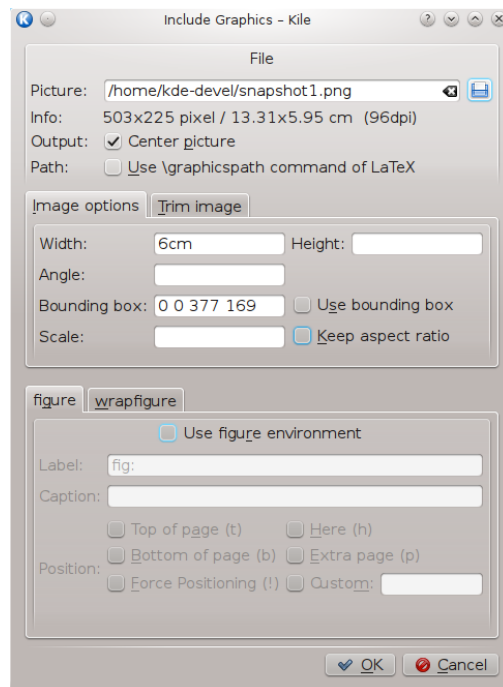
Wizards and Dialogs

7.1 QuickStart Wizard

This wizard has already been described in the section Section 2.3.1.

7.2 Include Graphics

The **Include Graphics** dialog makes insertion of graphics as easy as possible. You can reach it via the menubar with **LaTeX** → **Image Insertion**. Please take a look at Section 10.3 and Section 10.4 to get an overview of some basic facts concerning graphic formats.



1. Choose a graphics file. This can be a JPEG, PNG, PDF, EPS or even a zipped or gzipped EPS file. If you have installed [ImageMagick](#) and also configured Kile to use it (**Settings**

→ **Configure Kile...** → **LaTeX+Graphics**), the width and the height of the graphic is automatically shown. If ImageMagick can determine a resolution, the size of the graphics is also shown in centimeters.

2. Decide whether your image shall be centered on the page.
3. Choose whether you want the `\graphicspath` notation for your graphics file.

By default graphics files have to be in the same folder as your master document. However, it is possible to put them in other folders to make things tidier. Without a `\graphicspath` command, Kile would include the path for the graphics file. But if you use `\graphicspath` in your preamble like this:

```
\graphicspath{{/path/to/my/graphics}{other/path/to/more/graphics}}
```

and check this option, Kile will only use the base name of the graphics file.

Another example: if you set `\graphicspath` command like:

```
\graphicspath{{.}{camera/}{images/}}
```

L^AT_EX will search in the current folder, then in `camera` and finally in `images` to find your graphics file.

4. If you choose either a width or a height, the whole graphics will be proportionally scaled. If you set two values for width and height at the same time, width and height may be scaled with different factors, and this could not be what you want. See also the information near the top of the dialog to know the original size of the graphics.
5. Insert an angle by which to rotate the graphics counterclockwise.
6. The bounding-box information is set automatically when you choose a graphics file. This information is only needed when you work with traditional L^AT_EX and bitmapped graphics. See the discussion of [EPS graphics](#).
If L^AT_EX needs a bounding box and you do not want to generate a `bb` file, Kile supports this option. On the other hand, PDFL^AT_EX will give a warning when you want to include a `png` or `jpg` graphics with this option. This checkbox enables or disables the bounding-box option.
7. Scale the image by the desired scale factor. e.g., 0.5 to reduce by half, or 2 to double. When you use this option, you do not have to set a width or height for the image.
8. In the **Trim Image** tab you can crop your image in all four directions.
9. Finally, you have to specify whether you want to embed this image into a figure environment. When you want the text to wrap around the figure, use the `wrapfigure` environment instead.

NOTE

When you choose the `wrapfigure` environment, you need to include the `wrapfig` package in your preamble.

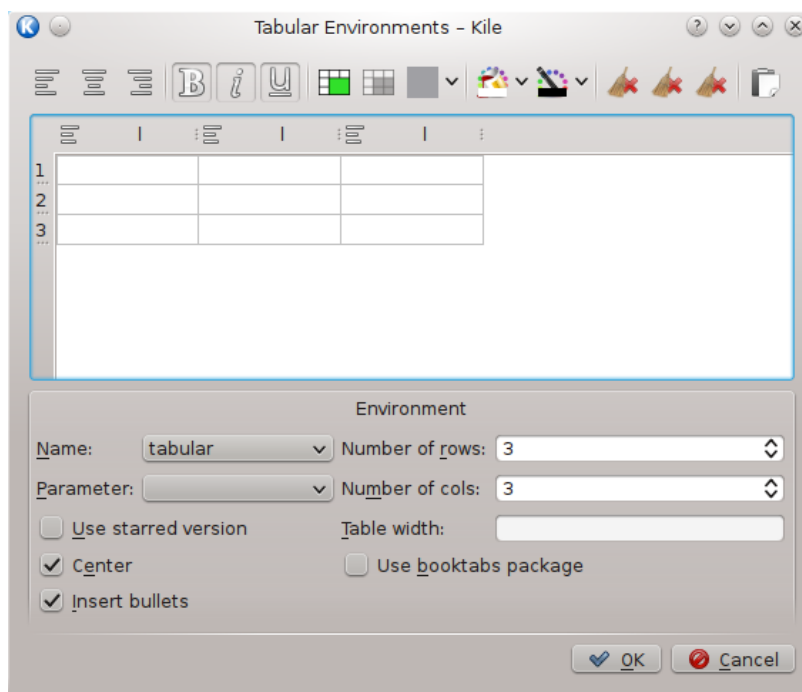
In either case you can insert a caption and a label for your image. Generally, it is a good idea to add a different prefix to each kind of label. It is common to use the prefix **fig:** for images.

10. If you pick the figure environment, you can choose where L^AT_EX should preferably position the figure.
11. In the `wrapfigure` environment you can:

- (a) Pick a placement rule for the figure and decide whether the figure should float or not. In a two-sided document you can define whether the figure should be on the inside or outside edge of the page.
- (b) Define how many shortened lines of the text are set alongside the figure. If you leave this empty, L^AT_EX will determine this itself as well as is possible.
- (c) Define an overhang to the chosen side. This is especially useful when you have columns in your document and you want a figure to span over more than just one column or you want shortened text on both sides of the figure.
- (d) Choose a width for the figure. This should be a bit bigger than the actual image width, so there will be some empty space between the figure and the text.

7.3 Arrays and tabulars

One of the most boring jobs one can do in L^AT_EX is to write a matrix or a tabular environment. One has to keep track of all the elements, ensure that the environment is well formed, and that all things are where they are supposed to be. Good indentation helps, but there is a simpler way: using Kile's **Wizard** → **Array** or **Wizard** → **Tabular** menu entries. You will then have a matrix-style input form that you can easily fill in with your entries. This dialog also offers some options to typeset the tabular material.



Using the toolbar on top of the dialog you can set the **align** of a cell, define a certain **font style**, **join** and **split** cells, choose a **border**, and specify background and font **colors**. On the extreme right there is a **Paste** button. With this button you can insert a table from the clipboard into the dialog, which allows you to copy and paste tables from a spreadsheet program, for example.

Below you can choose how many rows and columns you want, and you can tweak some more details about your array:

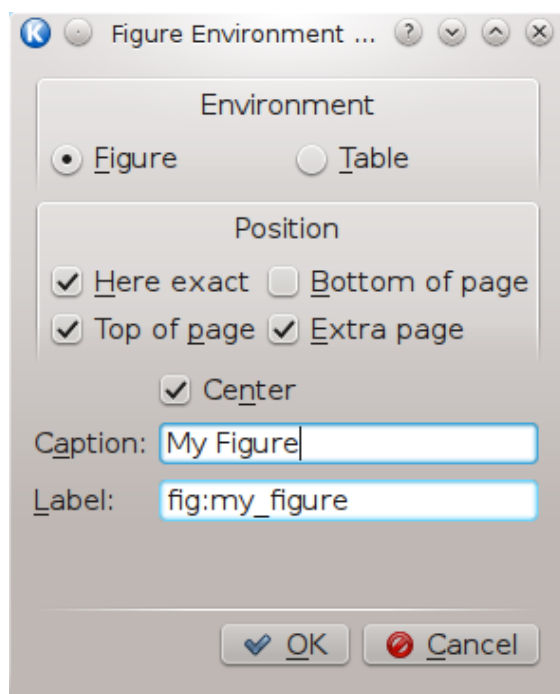
- With the **Name** option you can select which environment should be used for your array or tabular material.

- You can select the vertical cell alignment with the **Parameter** option. This is only enabled for environments which support that feature.
- If it exists for the selected environment, you can select **use starred version**. When you select this option, you also have to specify a **table width**.
- Tables sometimes look nicer when you select the **use booktabs package** option.
- Of course, you can also **Center** your whole array.
- **Inserting bullets** helps you when you want to fill in your content in the editor. With this option checked, Kile will insert bullet placeholders for each element of your array.

The **Wizard** → **Tabbing** option will display a simpler dialog to quickly set up a tabbing environment. It allows you to easily specify the number of rows or columns and the required spacing.

7.4 Inserting floating elements

Kile helps you with inserting your floating elements. With the **Wizard** → **Floats** wizard it is very simple to create a new figure or table environment.

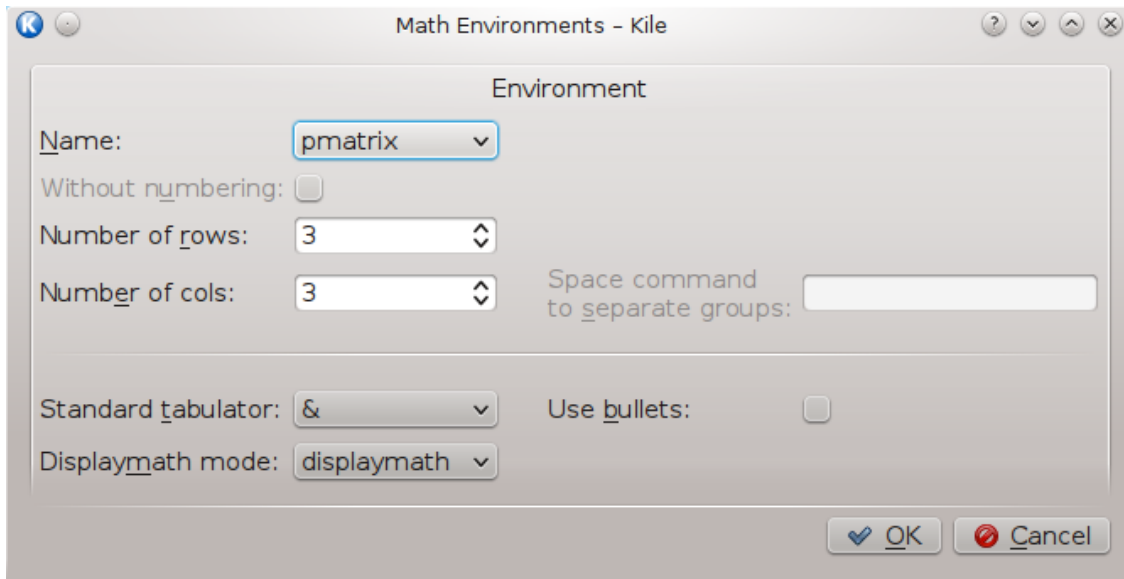


To insert a new floating environment just follow these steps:

1. Choose whether you want to insert a figure or a table.
2. Select the desired positioning rules.
3. Enter a caption for your floating element.
4. Type in a label for your new floating element. Kile will automatically suggest an appropriate prefix, e.g. "fig:" for figures and "tab:" for tables.

7.5 Inserting Math environments

Remembering how all the different math elements work can be really annoying. Of course Kile can do the magic for you here: **Wizard** → **Math**



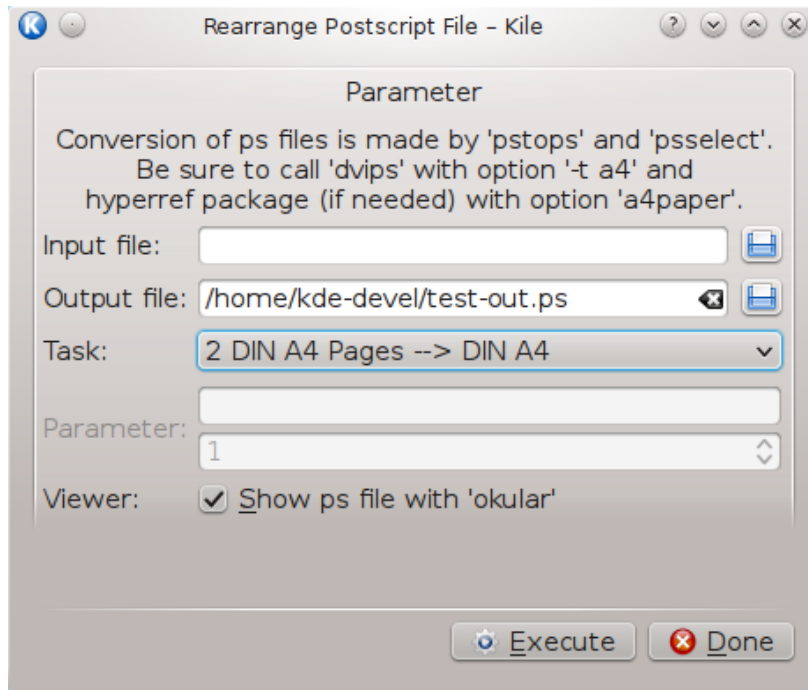
Options:

- **Name:** Choose the type of math element you want to create.
- **Without numbering:** This can switch numbering off for numbered elements like equations or aligns.
- **Space command to separate groups:** In an environment which supports several groups like alignat, you can define a space separator when you have more than one group. You can enter any space command here, which exists in mathmode, e.g. `\quad`.
- **Standard tabulator:** Select the tabulator which should be used. Kile should automatically pick the right one for you here.
- **Displaymath mode:** For environments like matrices or arrays you can choose which math environment your mathematical text should be displayed with.
- **Use bullets:** With this option checked, Kile will insert bullet placeholders for each element of your mathematical text.

7.6 PostScript[®] Utilities

PS files are not so popular as PDF files, but are an excellent base for manipulations and rearrangements of pages. If you need PDF output, you can rearrange pages with some PostScript[®] utilities and then convert it to PDF with `ps2pdf`.

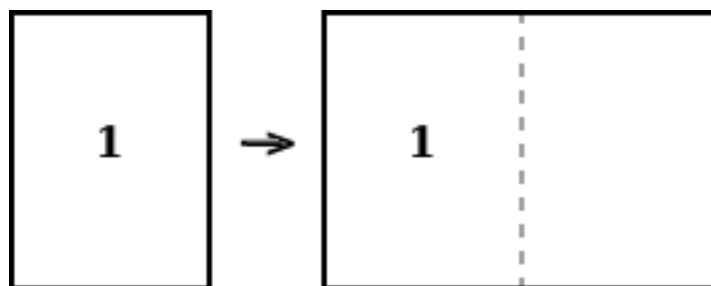
The *PostScript Wizard* under **Wizard** → **Postscript Tools** will suggest the most popular rearrangement. The conversion is done by the programs `pstops` and `pselect`, which can be found in most distributions in the package `psutils`. If one of these programs is not available, the corresponding item will not be visible.



First choose your input file. If Kile finds a PS file corresponding to your current master document, it is already filled in as the input file, but you are also free to choose another file. Then choose an output file, and select one of the tasks. Finally, you have to decide whether you want to do the conversion only, or also invoke Okular to view the result.

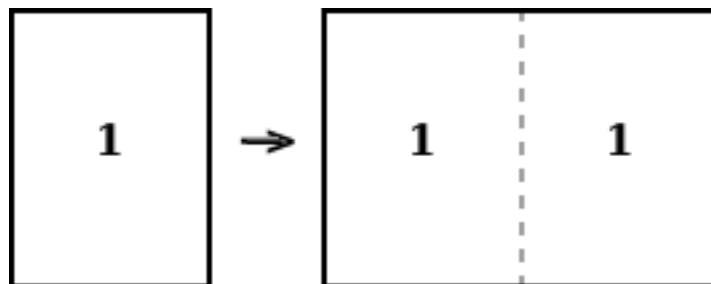
1 A5 page + empty page --> A4

Combine one A5 page together with one empty page on one A4 page. Whenever two A5 pages are combined together, they are rotated 90 degrees and will be arranged on an A4 page in landscape mode.



1 A5 page + duplicate --> A4

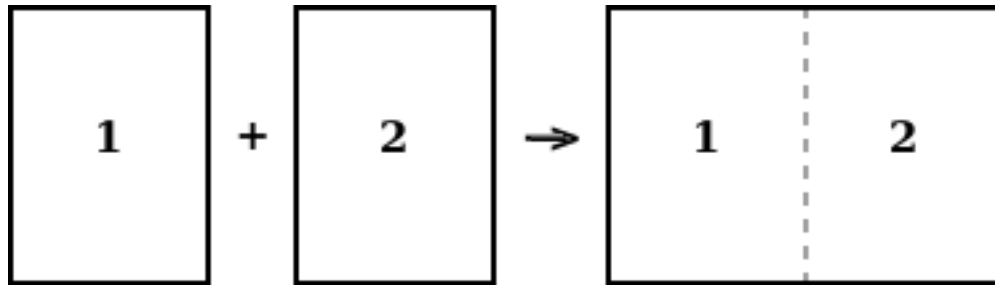
Put one A5 page and a duplicate together on one A4 page.



2 A5 pages --> A4

The Kile Handbook

Put two consecutive A5 pages together on one A4 page.

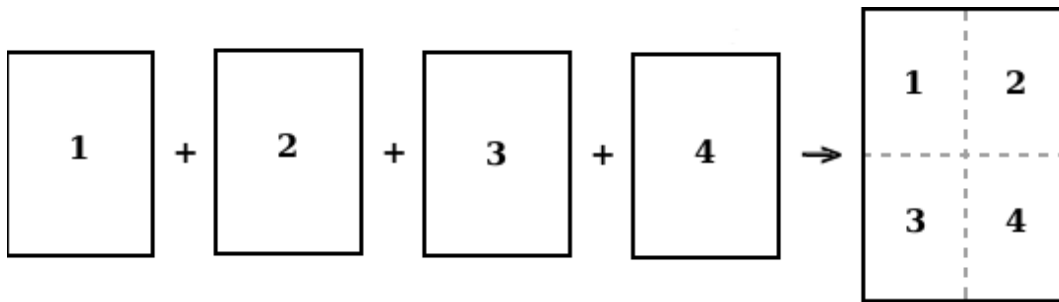


2 A5L pages --> A4

Put two consecutive A5 pages in landscape mode together on one A4 page.

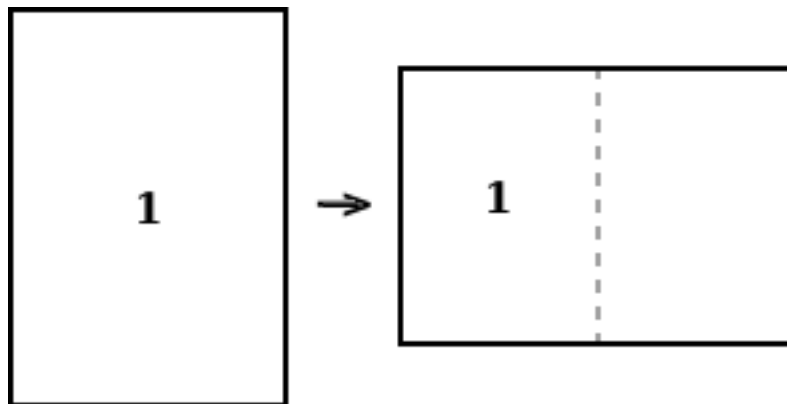
4 A5 pages --> A4

Combine four consecutive A5 pages together on one A4 page. The A5 pages have to be scaled with factor 0.7 to fit on the page.



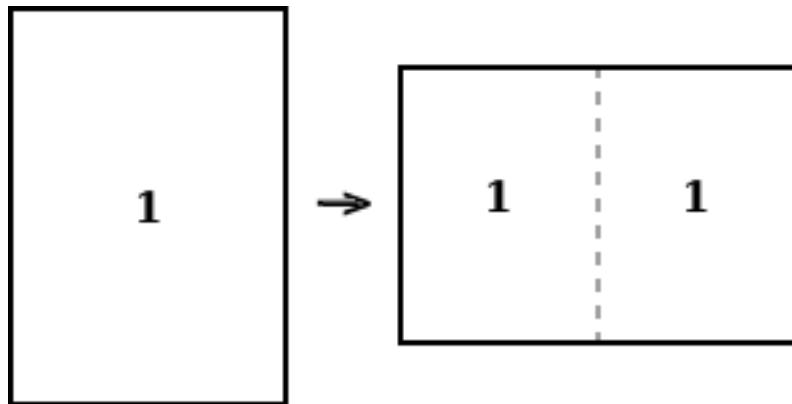
1 A4 page + empty page --> A4

Combine one A4 page together with one empty page on one A4 page. Whenever two A4 pages are combined together on one resulting A4 page, they have to be scaled with factor 0.7 and will be arranged in landscape mode.



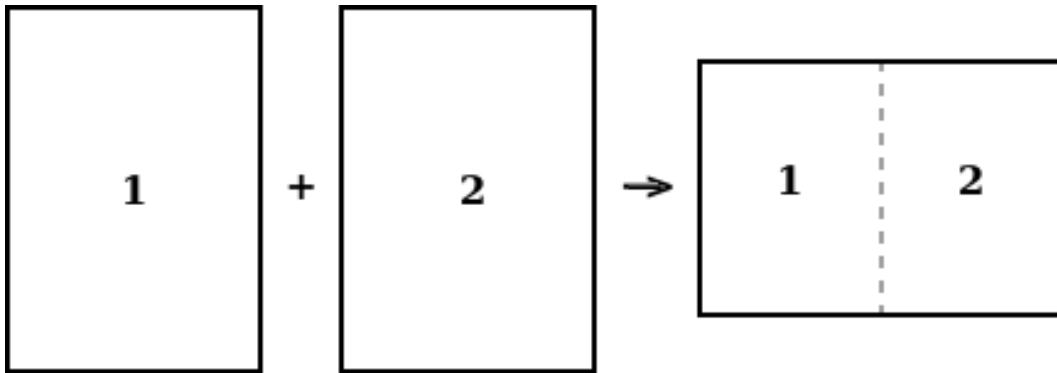
1 A4 page + duplicate --> A4

Put one A4 page and a duplicate together on one A4 page.



2 A4 pages --> A4

Put two consecutive A4 pages together on one A4 page.



2 A4L pages --> A4

Put two consecutive A4 pages in landscape mode together on one A4 page.

select even pages

Select all even pages of a document.

select odd pages

Select all odd pages of a document.

select even pages (reverse order)

Select all even pages of a document and reverse the order.

select odd pages (reverse order)

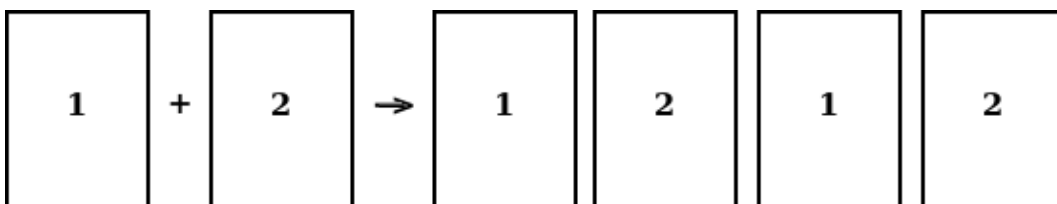
Select all odd pages of a document and reverse the order.

reverse all pages

Reverse all pages of a document.

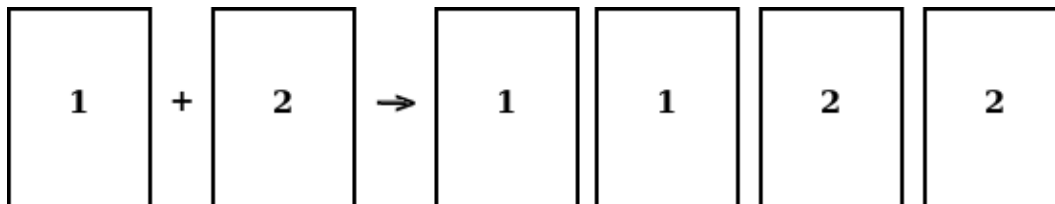
copy all pages (sorted)

Copy all pages of a document. You have to choose the number of sorted copies.



copy all pages (unsorted)

Copy all pages of a document. You have to choose the number of non-sorted copies.



pstops: choose parameter

There are many options for PostScript[®] utilities **pstops** and **psselect**. If you need a very special one, you can invoke **pstops** with an option of your choice. Please read the manual for all possible options.

psselect: choose parameter

You can invoke **psselect** with an option of your choice. Please read the manual for all possible options.

7.7 PDF Utilities

Many people think of PDFs as frozen files, which cannot be modified. But this is not true, as there exist excellent tools

- for manipulations and rearrangements of pages
- to read and update document info
- to read, set or change some permissions

of an existing PDF document.

Kile's *PDF wizard* under **Wizard** → **PDF Tools** uses two different methods to manipulate and rearrange PDF documents:

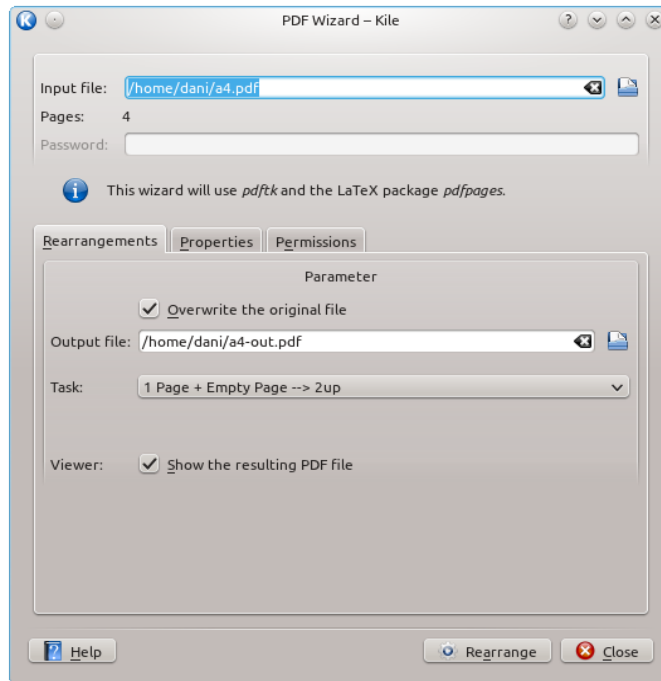
- L^AT_EX package **pdfpages**, which is part of each L^AT_EX distribution. **pdfpages** doesn't work with encrypted pages.
- **pdftk**, which is an excellent command line tool for doing everyday things with PDF documents (see [The PDF Toolkit](#)).

If one of these helpers, **pdfpages** or **pdftk**, is not present in your system, the corresponding items will not be visible. Furthermore, remember that only **pdftk** can work with encrypted files.

7.7.1 Rearrangements

If Kile's PDF wizard is called, it starts with the **Rearrangements** register card.

The Kile Handbook

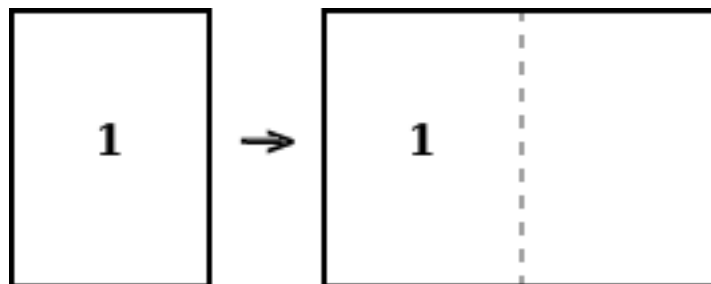


First choose your input file. If Kile finds a PDF file corresponding to your current master document, it will already be filled in as the input file, but you are also free to choose another file. Then choose an output file or overwrite the existing PDF file, and select one of the tasks. Finally, you have to decide whether you want to do the conversion only, or also invoke the viewer (e.g. Okular) to show the resulting document.

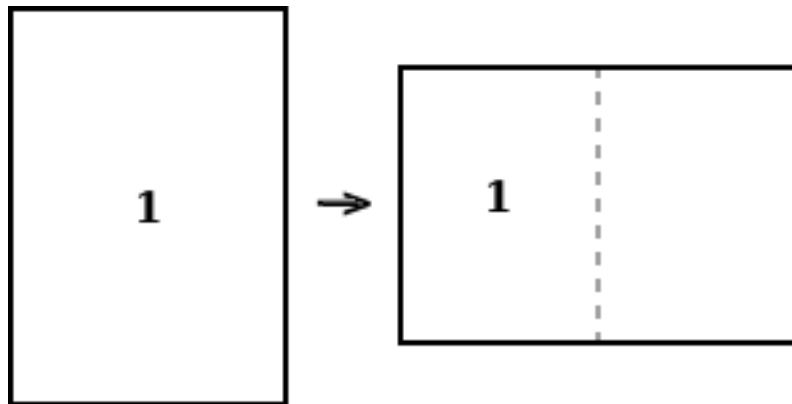
If the PDF file is encrypted, only **pdftk** will work and you have to give the password of this document to execute tasks.

1 page + empty page --> A4

Combine one page together with an empty page on one A4 page. Whenever two A5 pages are combined together, they are rotated by 90 degrees and arranged on an A4 page in landscape mode.

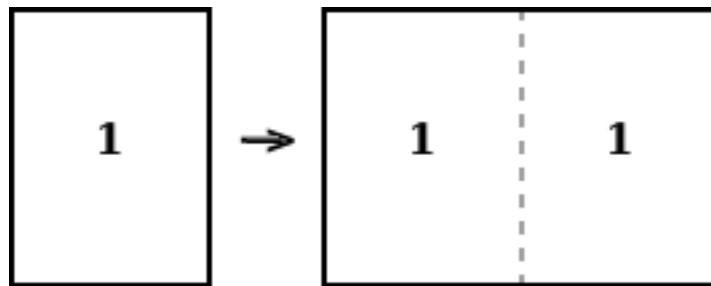


Whenever two A4 pages are combined together, they are scaled, rotated by 90 degrees and arranged on an A4 page in landscape mode.

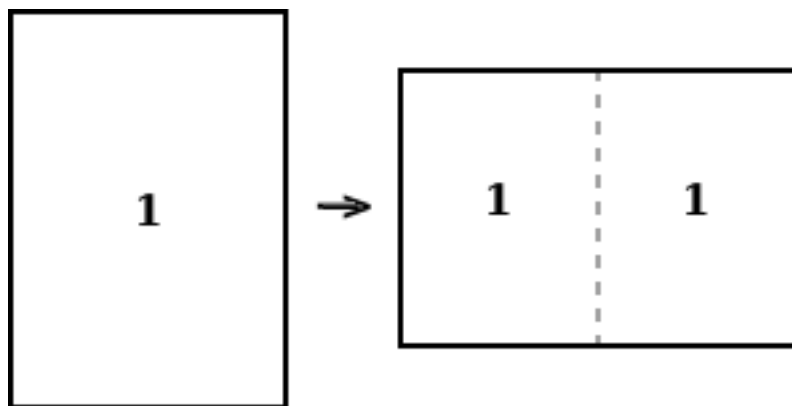


1 page + duplicate --> A4

Put one page and a duplicate together on one A4 page.

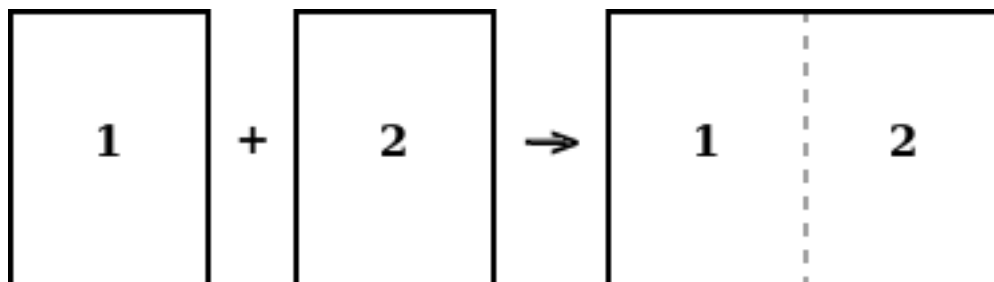


If the page to be duplicated has A4 size, it will be scaled to fit on the page.

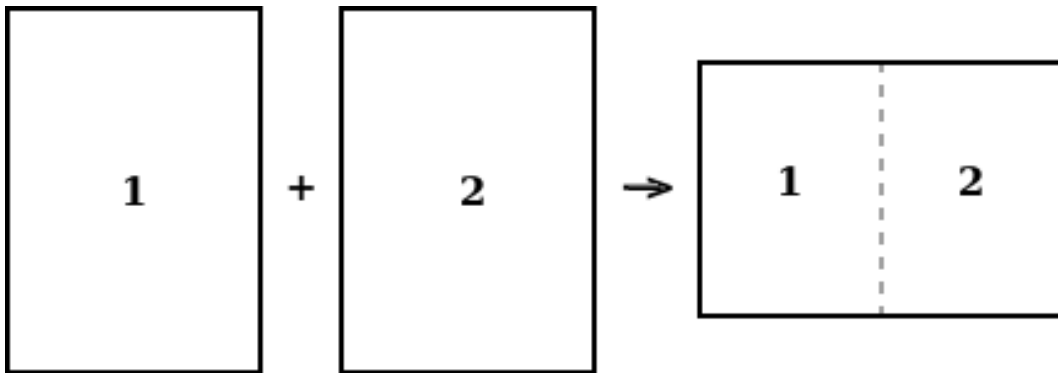


2 pages --> A4

Combine two consecutive pages together on one A4 page. Whenever two A5 pages are combined together, they are rotated by 90 degrees and arranged on an A4 page in landscape mode.

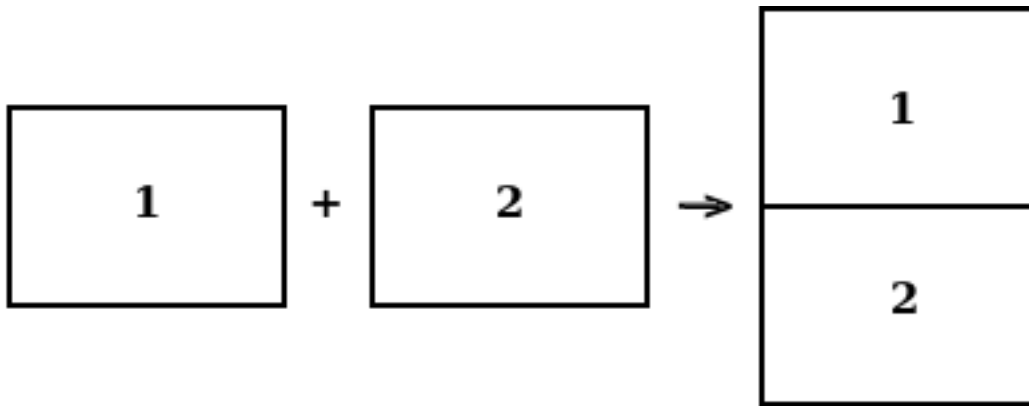


Whenever two A4 pages are combined together, they are scaled, rotated by 90 degrees and arranged on an A4 page in landscape mode.



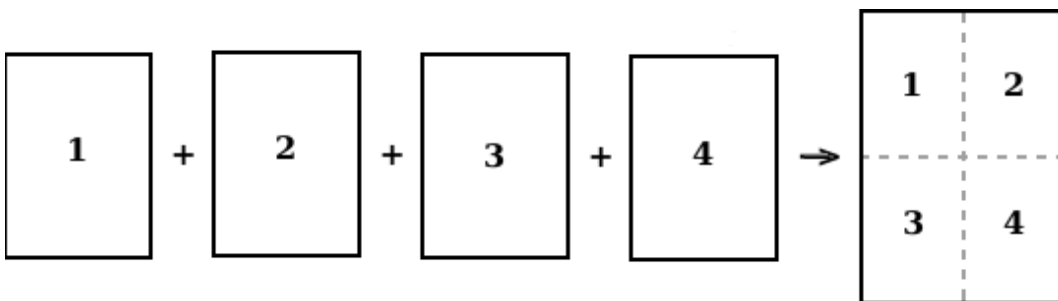
2 pages (landscape) --> A4

Put two consecutive pages in landscape mode together on one A4 page.



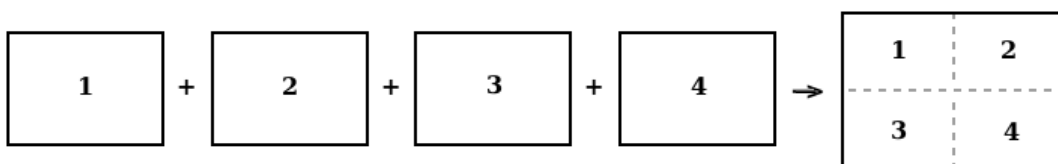
4 pages --> A4

Combine four consecutive pages together on one A4 page. The pages have to be scaled to fit on the page.



4 pages (landscape) --> A4

Combine four consecutive pages in landscape mode together on one A4 page. The pages have to be scaled to fit on the page.



select even pages

Select all even pages of a document.

select odd pages

Select all odd pages of a document.

select even pages (reverse order)

Select all even pages of a document and reverse the order.

select odd pages (reverse order)

Select all odd pages of a document and reverse the order.

reverse all pages

Reverse all pages of a document.

decrypt a file

If the PDF file is encrypted, you can decrypt it.

select pages

Add a comma separated list of pages or page ranges, e.g. 1,4-7,9. Only these pages will appear in the resulting PDF file.

delete pages

Add a comma separated list of pages or page ranges, which should be removed from the chosen PDF file.

apply a background watermark

Applies a PDF watermark to the background of a single input PDF. The wizard only uses the first page from the background PDF and applies it to every page of the input PDF. This page is scaled and rotated as needed to fit the input page.

apply a background color

Applies a background color to all pages of the current document. This can only be done once, as the second color will be put behind the first color and will not then be visible.

apply a foreground stamp

Applies a foreground stamp on top of the input PDF document's pages. The wizard uses only the first page from the stamp PDF and applies it to every page of the input PDF. This page is scaled and rotated as needed to fit the input page. This works best if the stamp PDF page has a transparent background.

pdftk: choose parameter

You can invoke **pdftk** with an option of your choice. Please read the manual for all possible options.

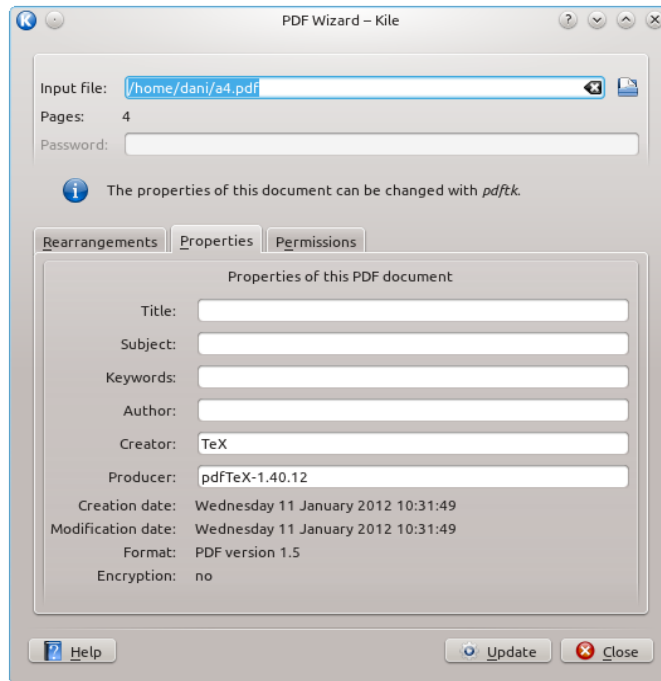
pdfpages: choose parameter

You can invoke **pdfpages** with an option of your choice. Please read the manual for all possible options.

7.7.2 Properties

The setting, changing and removing of properties will only be possible if **pdftk** is installed and if additionally Kile was compiled with the **libpoppler** library.

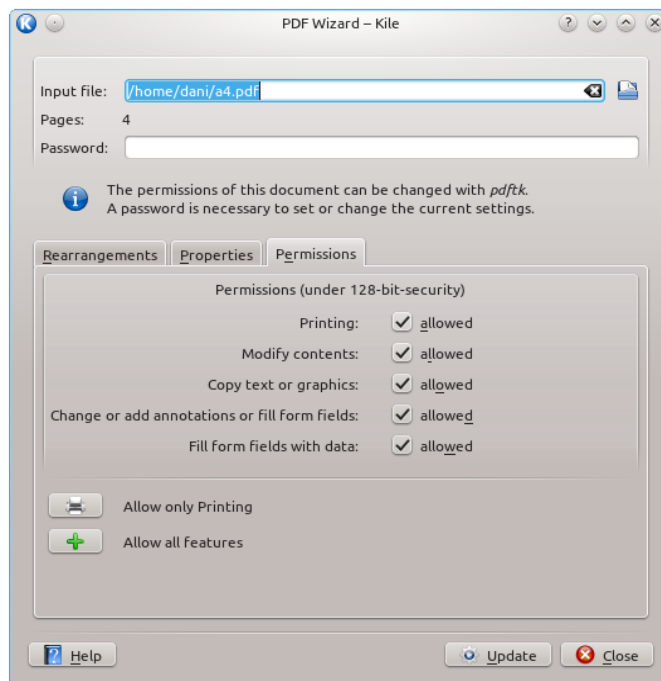
The Kile Handbook



Traditional PDF metadata includes the document's title, author, subject, keywords, creator, producer and the dates of creation and last modification.

7.7.3 Permissions

Also, the setting, changing and removing of permissions will be only possible if **pdftk** is installed.



A password is necessary to set or change these document settings. Additionally, PDF encryption is done to lock the file's content behind this password or to enforce lighter restrictions imposed by the author. So the author can allow or restrict:

- printing pages
- modifying pages
- copying text and graphics from pages
- changing or adding annotations
- filling form fields with data.

Changing permissions always forces encryption associated with 128-bit security of **Acrobat 5** and 6, and also needs a password.

But always remember: encryption and a password do not provide any real PDF security. The content is encrypted, but the key is known. You should see it more as a polite but firm request to respect the author's wishes.

7.8 Document Statistics

The statistics dialog in **File** → **Statistics** gives you a statistical overview for a selection, a document or a whole project. It includes the number of words, L^AT_EX commands/environments and also includes the number of characters for each type.

The statistics obtained can be copied as text or as a nicely formatted L^AT_EX table to the clipboard.

When you select a text and open the statistics dialog, you get the statistics for the currently selected text. If you open the dialog without any text selected, the statistics for all opened files are shown. If you want to get statistics for the whole project, you can use **Project** → **Open All Project Files** for an easy and quick way to open all source files of your project.

A note of caution has to be sounded about the accuracy of the numbers. We have included some logic to get a good estimate, e.g. K\"uhler gives one word and one command, with six and two characters respectively. But there are other combinations in which parts of commands are counted as words and vice versa. Please note that the algorithm was developed and tested for languages similar to English or German. So don't take the numbers for granted. If you write a report whose length has to be of a certain numbers of words or characters, please make some tests first in order to check whether Kile's accuracy satisfies your needs.

- Miscellaneous Text
- Delimiters
- Greek
- Special Characters
- Cyrillic Characters
- User Defined

The tooltips of the icons show the L^AT_EX commands and additionally needed packages.

Pressing **Shift** and clicking a symbol will result in $\$ \mathbf{\symbolcmd} \$$ being inserted. Similarly, pressing **Ctrl** inserts it in curly brackets.

If you insert a command which requires a package which is not included in your L^AT_EX document, you will see a warning message in the logview window.

The first list of symbols holds the **Most Frequently Used** symbols. Inserted symbols will be added to this list, for quick and easy reference. The ordering of the symbols will not be changed upon addition of new symbols, instead a reference counter is incremented. If the number of items exceeds 30 items, the item with the lowest count is removed.

The **User Defined** symbol list can hold your own symbols. To create your own symbols you need the program `gesymb` and the file `definitions.tex` from the kile source package. Additionally you need a L^AT_EX compiler (what a surprise) and `dvipng` (version 1.7 or later). The procedure is that you create a L^AT_EX file with `\input{definitions}`, which makes the commands listed below available, and let `gesymb mysymbols.tex user` (which calls L^AT_EX and `dvipng`) create the icons. After copying them to `$HOME/.kde/share/apps/kile/mathsymbols/user/` and restarting kile you can use your own symbols.

The following commands are defined in `definitions.tex`:

- `\command[\optarg]{\symbol}`: Include the symbol `\symbol` in the symbol list, the optional argument `\optarg` specifies the command which kile should insert. If it is not given the command in the mandatory argument is used.
- `\mathcommand[\optarg]{\symbol}`: Same as above, except that the command in the mandatory argument is inserted in math mode.
- `\pkgs[arg]{pkg}`: Declare that the command given in this line needs the L^AT_EX package `pkg` with the optional argument `arg`. This command has to be in front of the `\command` command and overrides any package specification by the `neededpkgs` environment.
- `\begin{neededpkgs}[pkgs-args]{pkgs} ... \end{neededpkgs}`: Has the same effect as above, but for all enclosed commands.

An example for completeness is given here:

```
\documentclass[a4paper,10pt]{article}
\usepackage{amssymb}
\input{definitions}
%
\begin{document}
\pagestyle{empty}
%
\begin{neededpkgs}{amssymb}
\mathcommand{\surd}
\pkgs{amsmath}\mathcommand[\ddddot{}]{\ddddot{a}}
\mathcommand{\angle}
\end{neededpkgs}
```

```

\command{"A}
\mathcommand{\exists}
\mathcommand[\stackrel{}{}]{\stackrel{abc}{=}}

%\begin{neededpkgs}[russian,koi8-r,T2C,]{babel,inputenc,fontenc,mathtext}
%
% \end{neededpkgs}
% this would need to include the packages
% \usepackage{mathtext}
% \usepackage[T2C]{fontenc}
% \usepackage[russian]{babel}
% \usepackage[koi8-r]{inputenc}
% just to explain the format
\end{document}

```

8.2 Using Bibitems

`\bibitem` is a command used to enter a reference in a `thebibliography` environment in your document. The syntax for using `\bibitem` is `\bibitem[label]{key}`.

The optional `[label]` is for you to add your own labeling system for the bibliography entry. If no label is set, the entries will be set in numerical order: [1], [2], [3], etc.

The argument `{key}` is used to reference and link the commands `\bibitem` and `\cite` to each other and the information they contain. The command `\cite` contains the label associated with the intended `\bibitem`, which is located inside a `thebibliography` environment, and contains the reference data. Both corresponding `\bibitem` and `\cite` must have the same `{key}`; the easiest way to organize keys is by the author's last name. The secondary braces in the `thebibliography` environment denote the longest bibliography label you expect to have. So, inserting `{foo}` means you can have any label shorter or as large as the expression `foo`. Failure to set this parameter correctly may result in a not so attractive indentation of your bibliography.

The bibliography is a section apart from your main document, and an example of code for the bibliography would look like the following:

```

\begin{thebibliography}{50}
  \bibitem{Simpson} Homer J. Simpson. \textsl{Mmmmm...donuts}. Evergreen ↔
    Terrace Printing Co.,
    Springfield, SomewhereUSA, 1998
\end{thebibliography}

```

Then, your main source code would contain the location of the information relating to the `\bibitem` using `\cite`. That source code would look similar to this:

```

My thesis, about the philosophy of The Simpsons\copyright comes from my ↔
  favorite book \cite{Simpson}.

```

As it is often difficult to remember the exact citation key once you have many references, Kile provides an easy way to insert a citation. Using **LaTeX** → **References** → **Cite** a list with all the citation keys pops up. Select the correct reference and a citation will be inserted into your document. To update the list of keys, either save the file, or **Edit** → **Refresh Structure**, or press **F12**. With code completion enabled, Kile will show you a list of all the `\bibitem`-labels as soon as you open up a `\cite` command.

The final result in your document's bibliography would then look like this:

```

[1] Homer J. Simpson. Mmmmm...donuts. Evergreen Terrace Printing Co., Springfield
, SomewhereUSA, 1998.

```

The Kile Handbook

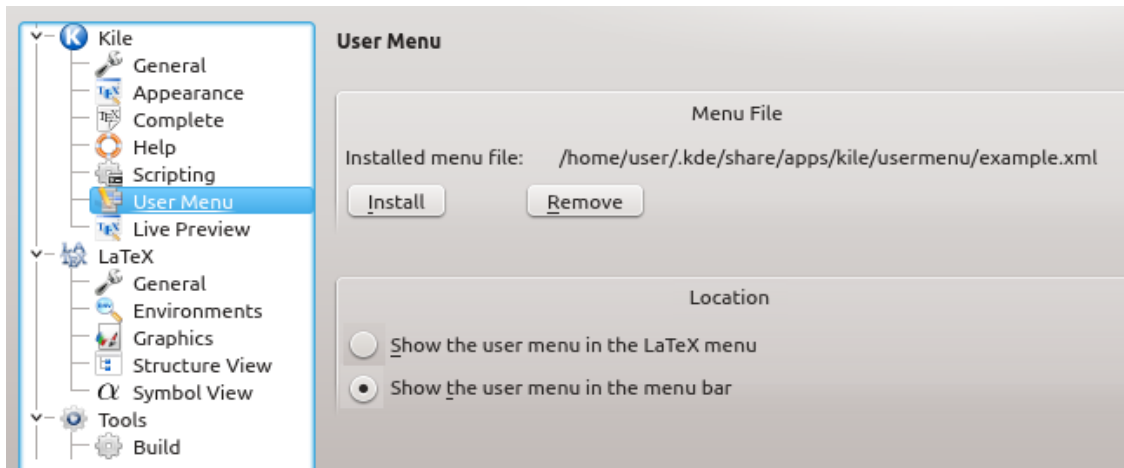
Kile can also work together with Bib_{T_EX} editors, such as KBib_{T_EX} to make it easier to enter citations. When a Bib_{T_EX} file is added to the project, Kile will help you complete citation commands, just as described above.

Chapter 9

User-Configurable Menu

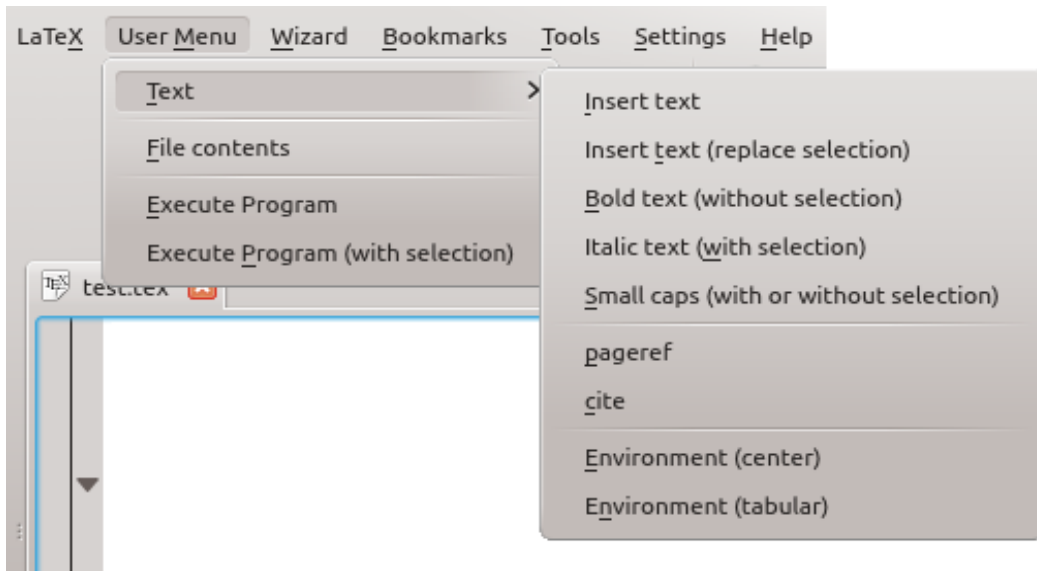
9.1 Configuration

Kile supports a user-configurable menu, which will appear as a part of Kile's menu. This menu can be configured using Kile's configuration dialog with **Settings** → **Configure Kile** → **User Menu**.

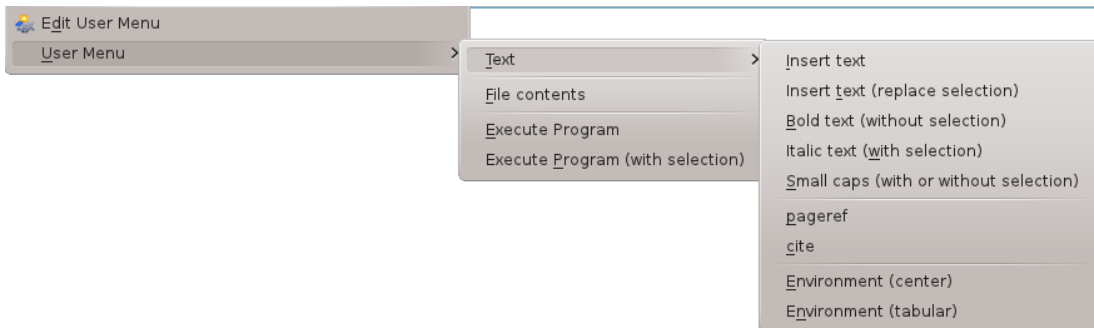


You have two options where to place this menu:

- either the menu **User Menu** will appear in the main menu bar between the menus **LaTeX** and **Wizard** and the configuration wizard **Edit User Menu** in the **Wizard** menu



- or both items will appear at the bottom of the LaTeX menu.



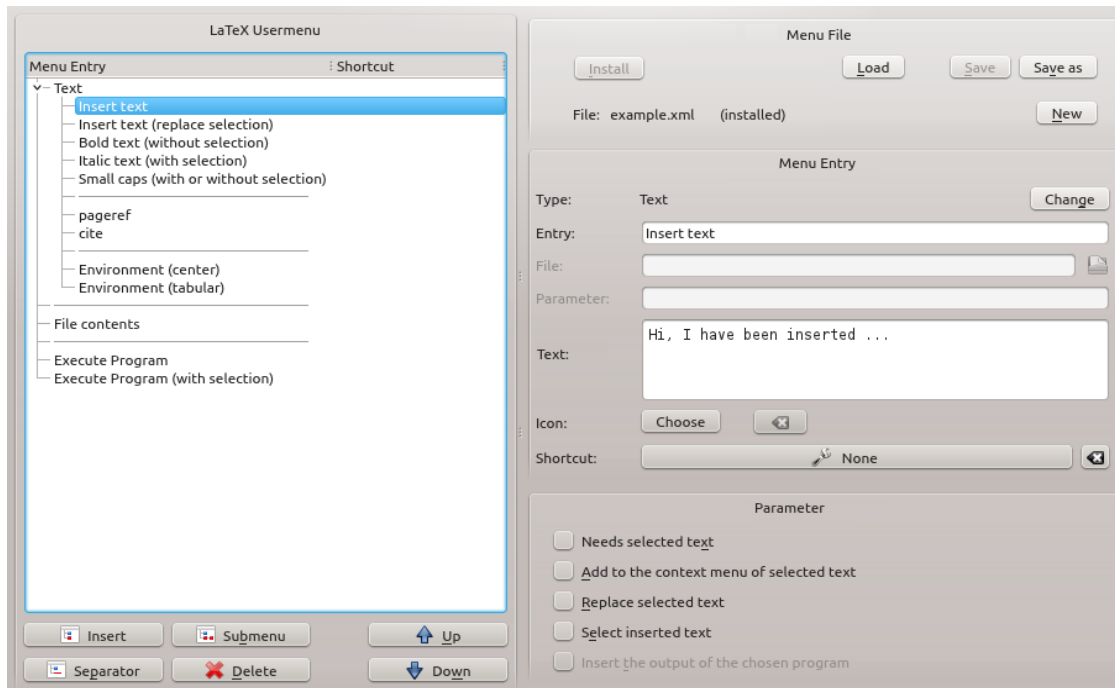
Already existing user-defined tags from older versions of Kile are automatically transformed into the new user-configurable menu. The tags are saved in a file called `usertags.xml` and like all menu definition files, they can be found in the local user menu directory of Kile: `KILE_APP_DIR/usermenu/`, e.g. `/home/user/.kde/share/apps/kile/usermenu/`.

You can use different menu definition files for different tasks. Call the user menu wizard **Wizard** → **Edit User Menu** or **LaTeX** → **Edit User Menu** to install or edit a menu definition file.

9.2 Wizard

You can create new or change existing menus with a comfortable user menu configuration wizard found at **Wizard** → **Edit User Menu**.

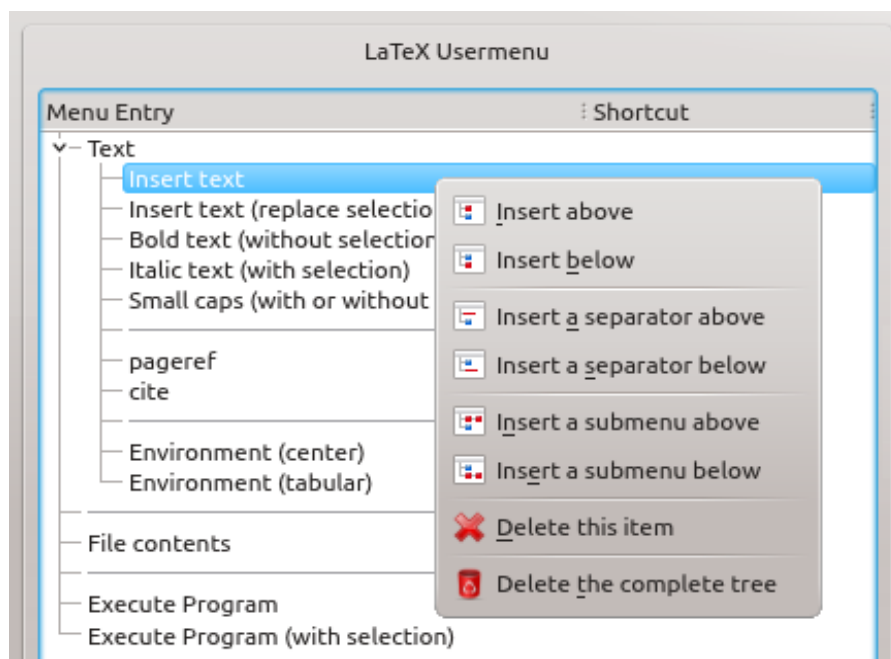
The Kile Handbook



On the left side you will see an existing menu-tree. Like a standard menu, three different kinds of menu items are available:

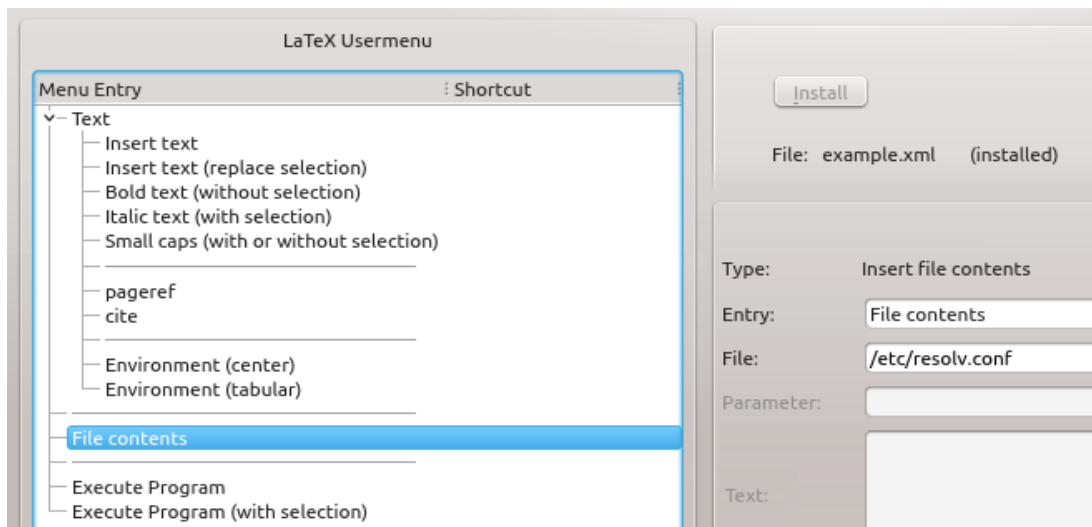
- standard menu entries, which are assigned to an action
- submenus, which contain more menu items
- separators, to get a visible structure of all entries.

To modify this menu, use the six buttons on the left side. More possible actions are available in the context menu of already existing menu items.

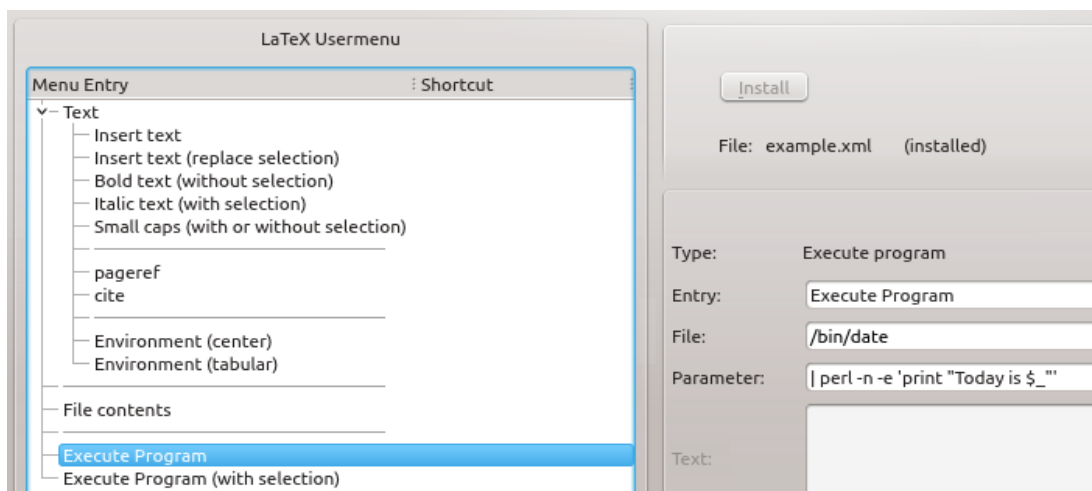


Each standard menu item is assigned to one of three action types, where each of them has different attributes, which could be set:

- **Text:** Kile gives you the ability to make your own tags. A tag is similar to a shortcut that launches some command or writes frequently-used texts, e.g. Joe Sixpack often uses the sentences **Hi, I have been inserted . . .** This tag will be inserted at the current cursor position, when this action is called (see above). Metachars are also available (see Section 9.3).
- **Insert file contents:** Inserts the complete contents of a given file.

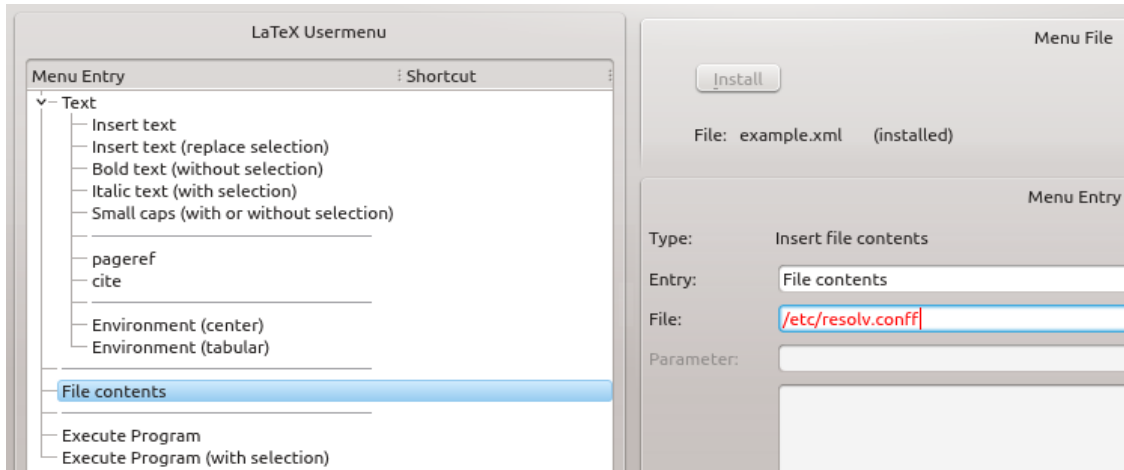


- **Execute an external program:** The output of this program can be inserted into the opened document. Metachar `%M` is also possible in the command line of this program, as the selected text will be saved in a temporary file. Use `%M` for the filename of this temporary file.

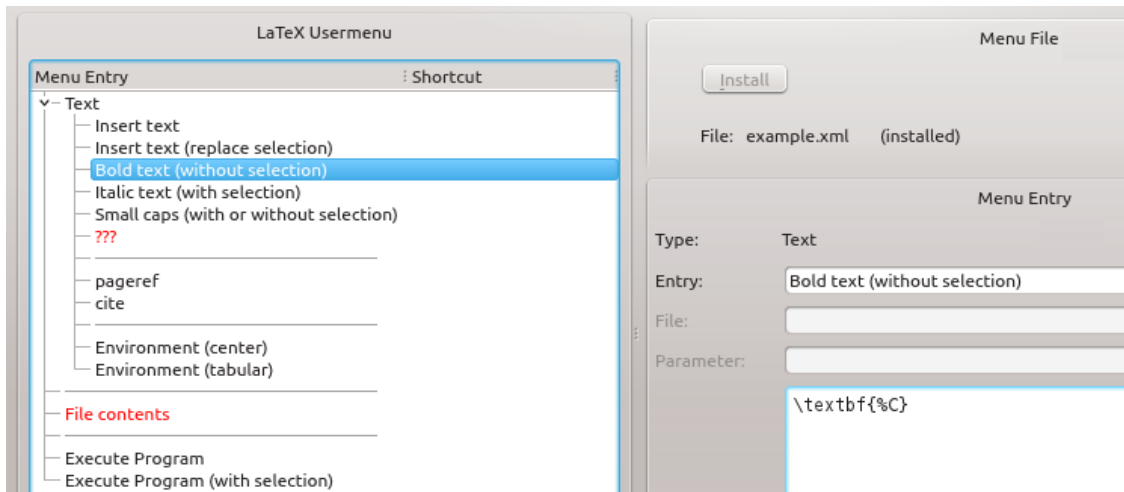


If some important information for an action is missing, menu items are colored red. This may be a nonexisting file

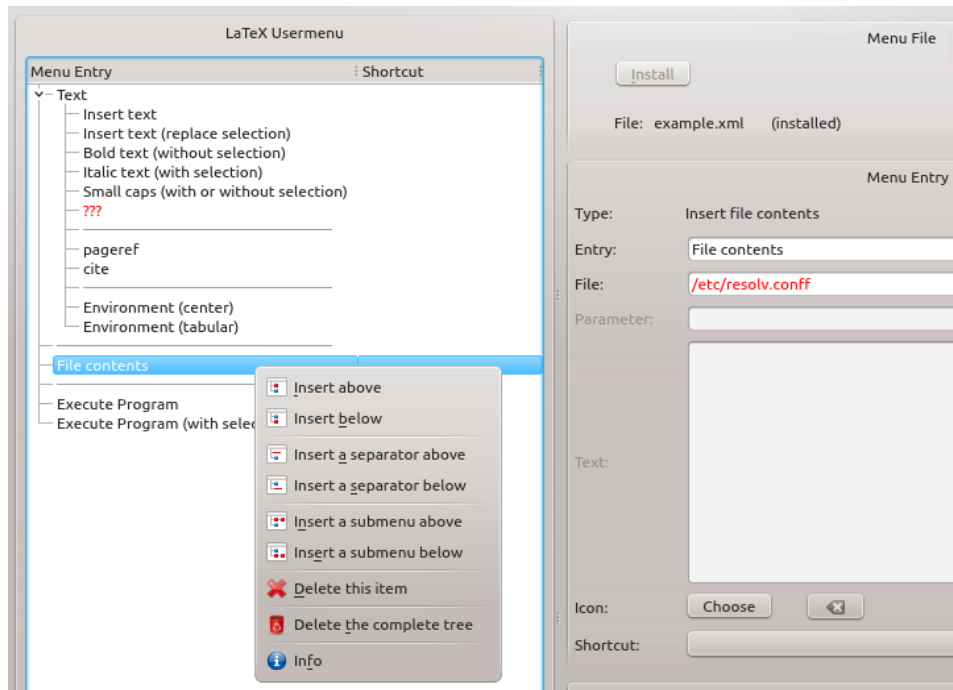
The Kile Handbook



or a missing title for the menu entry, which will be shown with question marks as ???.



If you open the context menu of such a red colored menu item, you will get an additional option for more information concerning this error.



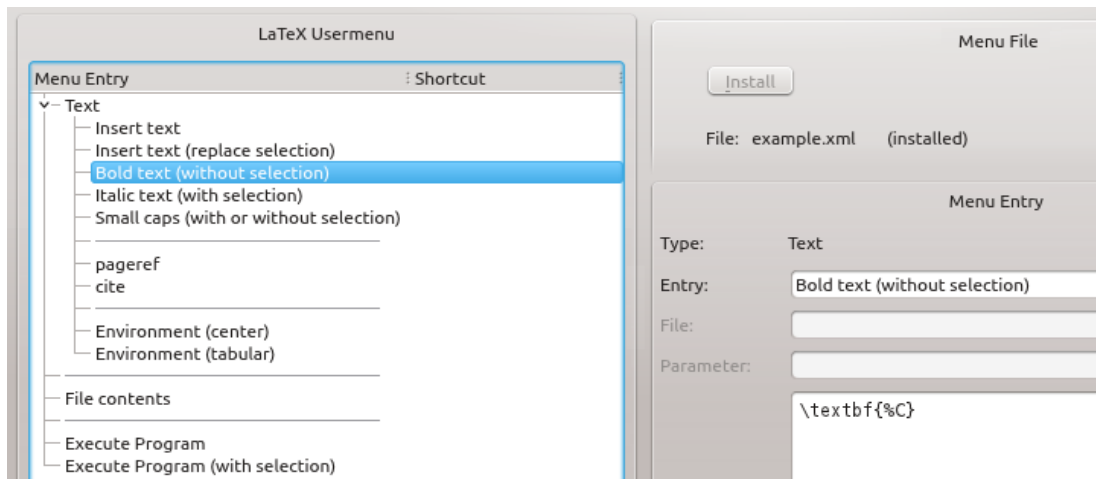
More information may also be available using the **What's this** feature of most widgets.

9.3 Placeholders

9.3.1 Insert Text

There are some placeholders you can use in your user-defined tags: **%C**, **%B**, **%M**, **%E**, **%R** and **%T**.

- **%C**: this is where the cursor will be placed after the insertion of a user-defined tag.



- **%B**: will be replaced by a bullet (see Section 5.5).

Type:	Text
Entry:	Environment (tabular)
File:	
Parameter:	
	<pre>\begin{tabular}{%B} %B \end{tabular}</pre>

- **%M**: will be replaced by the selected text.

Type:	Text
Entry:	Italic text (with selection)
File:	
Parameter:	
	<pre>\textit{%M}</pre>

- **%E**: denotes the indentation depth of text inside an environment.

Type:	Text
Entry:	Environment (center)
File:	
Parameter:	
	<pre>\begin{center} %E%C \end{center}</pre>

- **%R**: will call a reference-dialog to choose a label which has already been defined. This can be used to refer to a predefined label, which you can choose from a drop-down list (see also **LaTeX** → **References** → **ref** or **LaTeX** → **References** → **pageref**).

Type:	Text
Entry:	pageref
File:	
Parameter:	
	<pre>\pageref{%R}</pre>

- **%T**: will call a citation-dialog to choose an already defined citation. The same as using **LaTeX** → **References** → **cite** a list with all the citation keys pops up.

Type:	Text
Entry:	<input type="text" value="cite"/>
File:	<input type="text"/>
Parameter:	<input type="text" value="\cite{%R}"/>

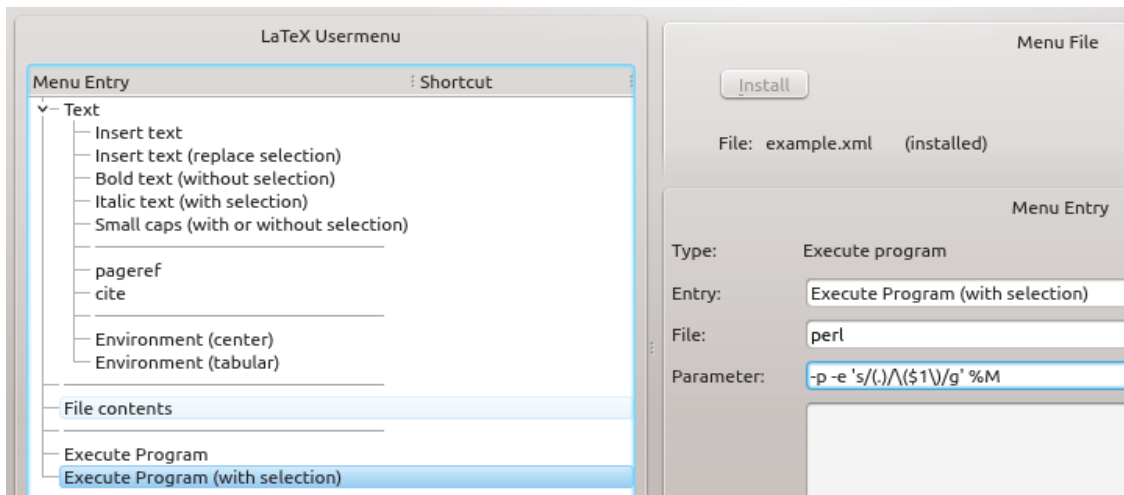
Let's consider another example, with the following macro `\frac{%M}{%C}`. First, we select a number in our text, let's say **42**. Now we invoke this macro and obtain `\frac{42}{ }` with the cursor located within the second pair of brackets.

9.3.2 Insert File Contents

If you want to insert the contents of a text file, you could use the same placeholders.

9.3.3 Execute A Program

If you want to execute an external program, only the `%M` for selected text is recognized in the command line. The selection will be saved in a temporary file and the placeholder `%M` is replaced with this filename.

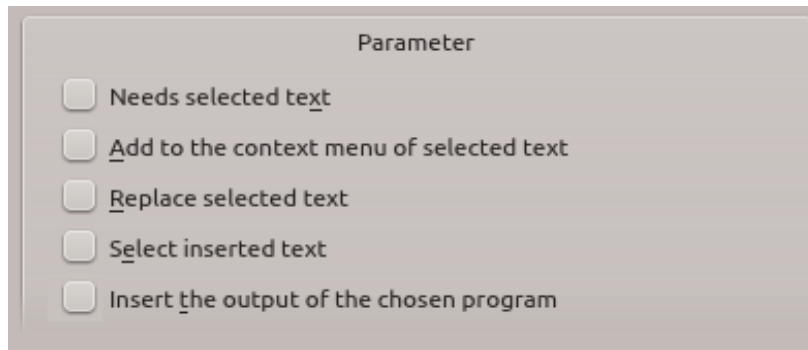


Another placeholder is `%S`, which is replaced by the complete base name of the current document without the path. This base name consists of all characters in the file up to (but not including) the last `'.'` character.

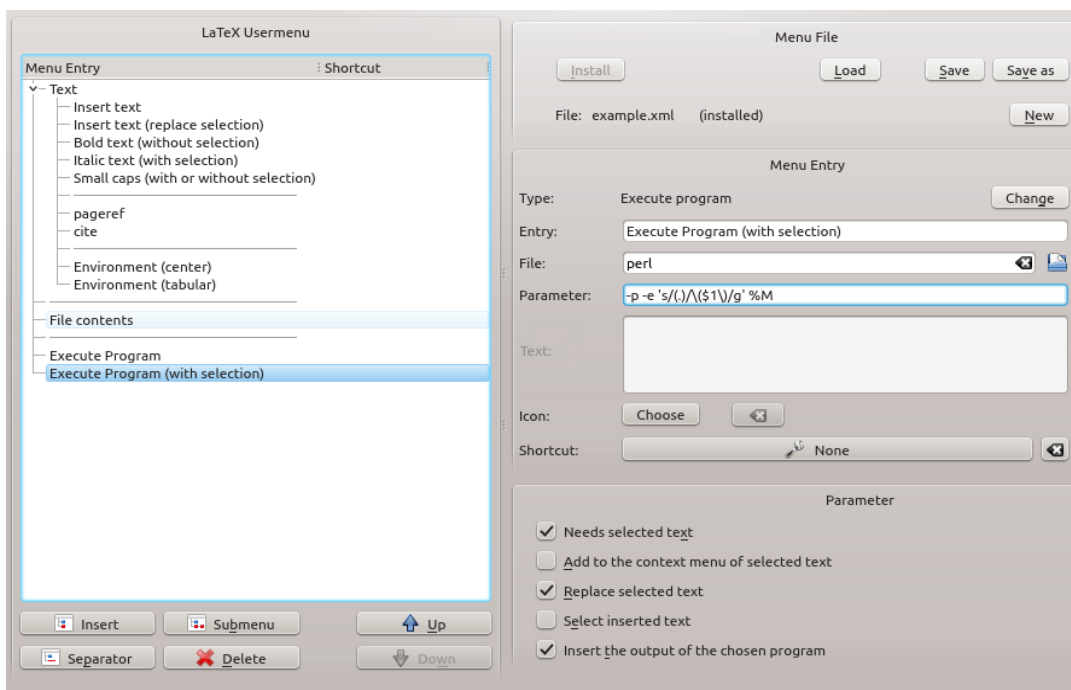
9.4 Parameter

Most menu entries may have additional self-explaining parameters, which may be checked. If some of these parameters are not available for some kind of action, they are disabled.

The Kile Handbook

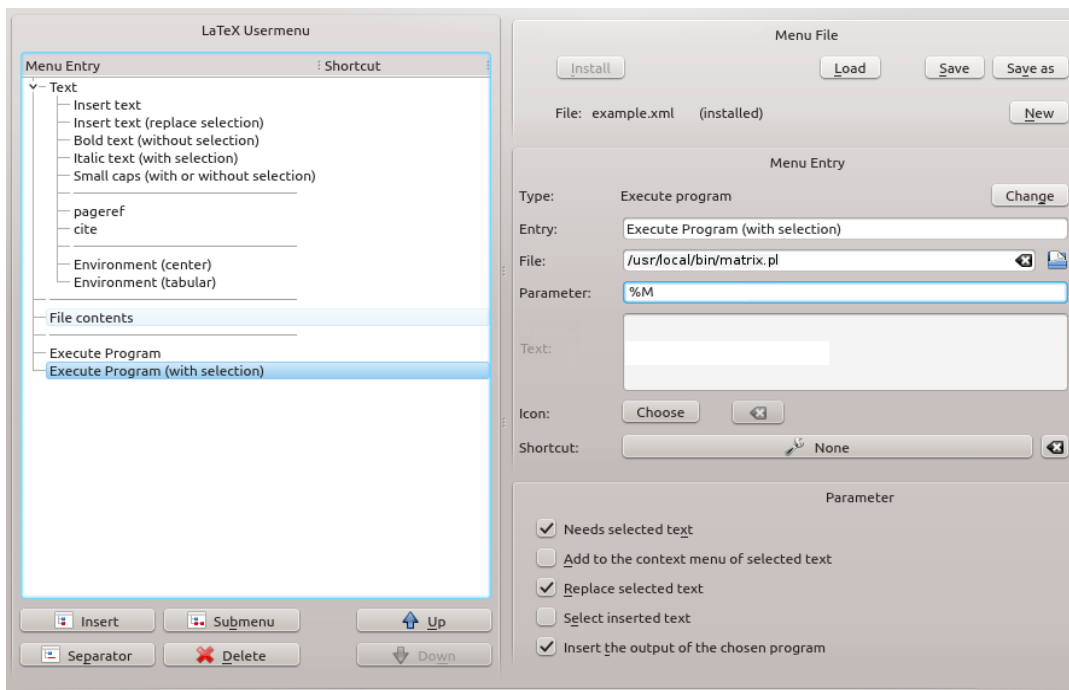


Here is one example for executing an external program:



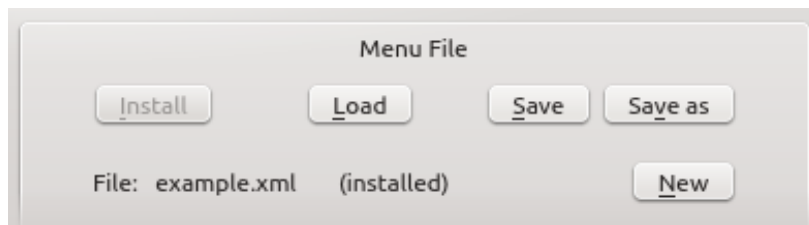
You can see that a **perl** script is called, which should work with current selection. The **Needs selected text** parameter is checked to assure a selection. The output of this script will be inserted (**Insert the output of the chosen program**) and replace the current selection (**Replace selected text**), but not selected itself.

Of course you can also call your own programs or scripts. For example select a list of numbers, separated by spaces, and call a script or Perl program, which transforms this selection into L^AT_EX code for a matrix. Whatever your ideas may be, you can realize them using the following usermenu entry.



9.5 Menu Definition Files

You can install different menus at runtime for different tasks. When you call the user menu wizard, the current menu definition file is loaded. If you modify it and close the dialog with **OK**, your changes will be saved and installed as the new user menu. Closing the dialog with **Cancel** will discard all changes.



You are also free to save the modified file in the user menu directory or to load another menu definition file and install it. All user menu definition files must be saved in the local user menu directory of Kile: `KILE_APP_DIR/usermenu/`.

Look at the example menu definition file **example.xml** to see more menu entries with their parameters.

Chapter 10

The Build Tools

10.1 Compiling, converting and viewing

To view the result of your work, you first need to compile the source. All the build tools are grouped closely together in the **Build** → **Compile**, **Build** → **Convert**, and **Build** → **View** menus.

To compile your source code for screen viewers like Okular or further conversion, you can use the shortcut **Alt-2**. Then you can view the DVI file using your default viewer with **Alt-3**, convert the DVI to a PS file with **Alt-4**, and view the PS file with **Alt-5**.

10.1.1 Bib_TE_X

If you are using **Bib_TE_X** for your bibliography entries, you usually have to follow a special compiling scheme. This means calling L^AT_EX and then Bib_TE_X and then L^AT_EX twice again. Fortunately Kile is clever enough to detect automatically if it is necessary to call additional tools like Bib_TE_X, makeidx and Asymptote. This logic is turned on by default and can be changed in **Settings** → **Configure Kile...** → **Tools+Build** in the **General** tab in the L^AT_EX and PDFL^AT_EX tools.

10.1.2 MetaPost and Asymptote

If you want to compile your document with MetaPost or Asymptote, picture drawing programs, you can do it with **Build** → **Compile** → **Metapost**, or **Build** → **Compile** → **Asymptote**.

10.1.3 PDFL^AT_EX

There is also another way to compile your document, if you want a PDF: you can run PDFL^AT_EX, that will compile the source directly into a PDF file, with **Alt-6**: you can then view the compiled file by pressing **Alt-7**.

Alternatively, you can convert a PS into a PDF with **Alt-8**, or a DVI directly into a PDF with **Alt-9**.

Using PDFL^AT_EX instead of L^AT_EX may be just a matter of simplicity or habit, but sometimes the behavior of the two programs can differ.

10.1.4 L^AT_EX to Web

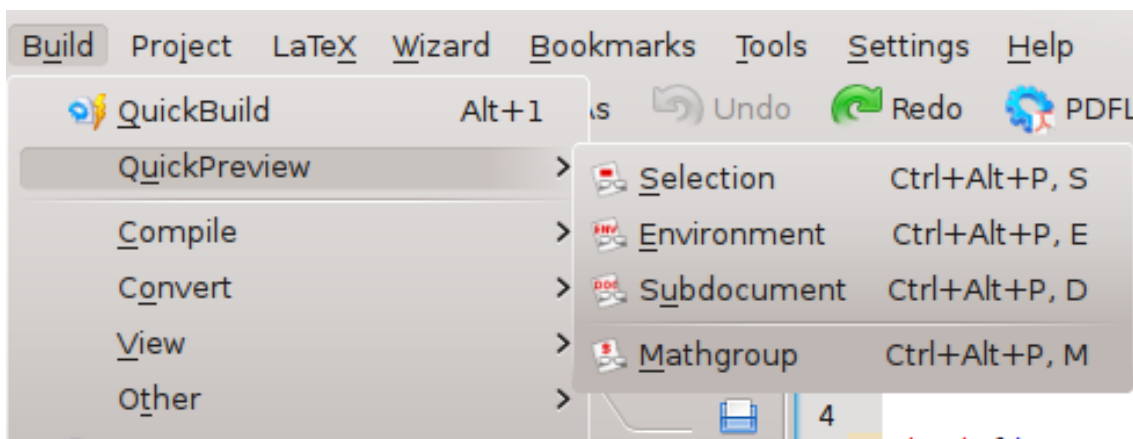
Finally, you may want to publish your work on the web and not just on paper. You may then use the latex2html program, that can be called from Kile's menu **Build** → **Convert** → **LaTeX to Web**. The result will be placed in a subfolder of the work folder, and you will be able to see the result of the conversion choosing the menu item **Build** → **View** → **View HTML**.

10.1.5 Passing Command Line Parameters

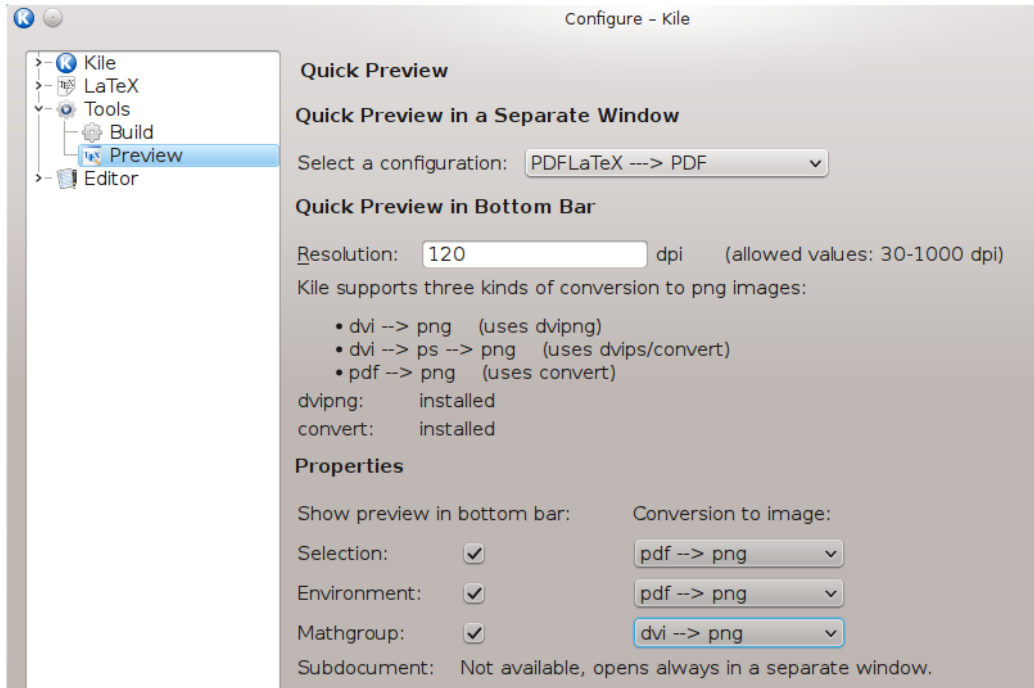
If you want to pass some specific command line parameters to the compile, convert or view tools, you can configure their call in **Settings** → **Configure Kile...** → **Tools+Build**.

10.2 Quick Preview

You will always need some time to view the result, when working with L^AT_EX. L^AT_EX has to compile the source and the viewer has to be called. This can be annoying if you only changed some letters in an equation difficult to typeset. Kile offers a *Quick Preview* mode, where you can compile only a part of a document and save a lot of time. It supports four different modes, which can be combined with seven configurations.



All settings must be done in **Settings** → **Configure Kile...** → **Tools+Preview**.



10.2.1 Selection Mode

The user has to select a part of the document. Menu entry **Build** → **QuickPreview** → **Selection** or the keyboard shortcut **Ctrl+Alt+P,S** will start the selected programs. Kile takes the preamble of the original text, so that all packages and user defined commands are included. The user can choose one of eight predefined configurations:

- LaTeX+DVI (embedded viewer)
- LaTeX+DVI (Okular)
- LaTeX+PS (embedded viewer)
- LaTeX+PS (Okular)
- PDFLaTeX+PDF (embedded viewer)
- PDFLaTeX+PDF (Okular)
- XeLaTeX+PDF (embedded viewer)
- XeLaTeX+PDF (Okular)

This should be sufficient for all situations for which a quick preview is needed.

10.2.2 Environment Mode

Very often you want to preview the current environment, and especially mathematical environments, which sometimes may be difficult to write. Kile offers a very fast way to do this. No selection is needed, just choose **Build** → **QuickPreview** → **Environment** or the keyboard shortcut **Ctrl+Alt+P,E** and the current environment will be compiled and shown.

10.2.3 Subdocument Mode

If you have a large project with a lot of documents, compiling the whole project is not a great idea, if you have made changes only in one single document. Kile is able to compile and show a preview of the current subdocument. It takes the preamble from the master document and only compiles the current part when you choose **Build** → **QuickPreview** → **Subdocument** or the keyboard shortcut **Ctrl+Alt+P,D**.

10.2.4 Mathgroup Mode

The mathgroup preview mode allows you to preview the mathgroup you are currently editing. Kile takes the preamble from the master document and only compiles the mathgroup the cursor is currently in when you choose **Build** → **QuickPreview** → **Mathgroup** or the keyboard shortcut **Ctrl+Alt+P,M**.

10.2.5 Quick Preview in Bottom Bar

Instead of showing the preview in a new document Kile can also be configured to use the bottom bar for preview compilations. You can activate this feature in the quick preview configuration panel.

10.3 Graphic File Formats

10.3.1 L^AT_EX and PDFL^AT_EX

PDFL^AT_EX, when used with **graphics** or **graphicx** packages, can correctly compile PNG and JPG files into DVI or PDF, but is not able to handle EPS files. Conversely, the process of compiling with L^AT_EX to DVI and converting to PS and eventually PDF does support EPS, but does not support PNG and JPG.

A lot of users want to create PDF documents, but also want to use the excellent Pstricks package to create PostScript[®] graphics, or they want to use the PostScript[®] output of mathematical and scientific software like Mathematica, Maple or MuPAD. These L^AT_EX users have to compile first in PostScript[®], even if they want to create PDF documents, because these programs produce PostScript[®] code which cannot be managed by PDFL^AT_EX. However, it is not so hard as it may sound, because Kile will help.

10.3.2 Graphics Conversion

To overcome this frustrating loop, when you want to include both PostScript[®] code and PNG or JPG files, you have a number of workarounds:

- If you need a file in PS format, but have JPG or PNG graphics, you can also simply use PDFL^AT_EX with DVI output first, and then run `dvips` to create the PS file. You see that PDFL^AT_EX is a very good choice, if your source contains no PostScript[®] code at all.
- You can convert EPS files to PNG or other formats with utilities as the [Gimp](#) or [ImageMagick](#) and use PDFL^AT_EX.
- A preferred way is to convert EPS graphics to PDF graphics with **epstopdf**, which comes with every T_EX distribution and then use PDFL^AT_EX. It produces high quality graphics, and you can even control the result with some of the following options:

```
-dAutoFilterColorImages=false
-dAutoFilterGrayImages=false
-sColorImageFilter=FlateEncode
-sGrayImageFilter=FlateEncode
-dPDFSETTINGS=/prepress
-dUseFlateCompression=true
```

Even better: if your system allows **shell-escape**, conversion can be done on the fly. All you have to do is to include the `epstopdf` package, which is part of all T_EX distributions, with the command `\usepackage{epstopdf}`. Assuming that your code is:

```
\includegraphics[width=5cm]{test.eps}
```

When you call PDFL^AT_EX with option `--shell-escape`, graphics `test.eps` is automatically converted into `test.pdf`.

This conversion will take place each time you run PDFL^AT_EX. If your graphics command is given implicitly:

```
\includegraphics[width=5cm]{test}
```

`epstopdf` checks whether `test.pdf` is already available, so that the conversion step can be skipped.

- You can convert the other way around, and use L^AT_EX and PS-PDF conversion. This is not always a good idea, since EPS encapsulation of JPG or PNG can yield larger files, that in turn yield unnecessarily large documents. This is however *highly* dependent on the graphic utility that you use, since EPS can encapsulate other graphics, but not all applications support this perfectly. Some might actually try to build your JPG image with vectors and various scripting, which will result in gigantic files. Conversion of all graphics formats to EPS can be done by [ImageMagick](#). Another simple program that does this process correctly is [jpg2ps](#).
- You can also use an automatic conversion. All graphics files are converted on the fly to EPS, and inserted into the PS document. This is a comfortable way, but you have to set up your system properly. This is discussed in the section [EPS Graphics](#).

10.3.3 Use the right File for the right Graphic

- EPS is sort of a graphic vector scripting language, describing all the lines and dots the graphic is made of; it looks good even when magnified beyond its default size, and suits best diagrams and vectorial graphics natively produced in EPS, which look very clear and sharp while maintaining a very small byte size.
- PNG (or the deprecated GIF) is a *non-lossy* file format, with good compression and quality. It is very good for diagrams, scans of drawings, or anything whose sharpness you do want to retain. It is sometimes overkill when used for photos.
- JPG is a *lossy* format, that compresses files better than PNG at the price of some loss in the picture detail. This is usually irrelevant for photos, but may cause bad quality for diagrams, drawings, and may make some thin lines disappear outright; in those cases use EPS or PNG.

But always remember: garbage in, garbage out! No conversion will make a bad picture good.

10.4 EPS Graphics

EPS graphics files are the traditional way to insert graphics files into L^AT_EX documents. As mailing lists are full with questions concerning EPS graphics, we will discuss some important aspects and demonstrate how Kile supports them.

10.4.1 L^AT_EX and EPS Graphics

If you decided to use the traditional L^AT_EX to produce PS or PDF output, you will probably run into some problems with graphics. You have to use EPS graphics (Encapsulated PostScript[®]); no JPEG or PNG files. This should be no problem, as there are a lot of [converters](#) like **convert** from the excellent [ImageMagick](#) package. But, it needs some time of course.

The EPS files are used by both L^AT_EX and the DVI-PS converter:

- L^AT_EX scans the EPS file for the bounding-box line, which tells L^AT_EX how much space to reserve for the graphics.
- The DVI-PS converter then reads the EPS file and inserts the graphics in the PS file.

This has some implications:

- L^AT_EX never reads the EPS file if the bounding-box parameters are specified in the graphics insertion command.
- Since L^AT_EX cannot read non-ASCII files, it cannot read the bounding-box information from compressed or non-EPS graphics files.
- The EPS graphics are not included in the DVI file. Since the EPS files must be present when the DVI file is converted to PS, the EPS files must accompany the DVI files whenever they are moved.

Now you can call L^AT_EX, and a DVI-PS converter like `dvips` to create your PostScript[®] document. If your goal is a PDF document, you should run **dvips** with option `-ppdf` and then call **ps2pdf**. You will find a lot of documents describing this solution.

10.4.2 The PostScript[®] Way of Kile

Kile helps you to get the bounding-box information. If you have installed [ImageMagick](#) package, Kile will extract this information from the EPS file and insert it as an option. This is done automatically when you select the graphics file. There are two advantages to proceeding like this:

- The information is already scanned in the dialog, and need not to be done by L^AT_EX later on.
- Even more important is that the width and height of the picture can be calculated, when the resolution is known. This information will be shown near the top of the dialog, and may serve as a clue when you want to scale the graphics.
- Kile can also support zipped or gzipped EPS files, which are much smaller than uncompressed EPS files. But, this feature can only be used with a special system setup and a change of your local graphics configuration, as it is described in the [Bitmap Graphics](#) section.

10.4.3 The PostScript[®] Way and Bitmap Graphics

If your system allows **shell-escape**, Kile also supports an easy way to include bitmap graphics, if you set up your T_EX system properly. There is no need to convert JPEG or PNG graphics, this can be done automatically when the DVI file is converted to PS.

L^AT_EX needs some information about the file suffixes. The package **graphicx** looks for a file `graphics.cfg`, which must be somewhere in your search path for L^AT_EX documents. Search for entries like:

```
\DeclareGraphicsRule{.pz}{eps}{.bb}{}%
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}{}%
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}{}%
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{}%
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{}%
```

and replace these lines with:

```
\DeclareGraphicsRule{.pz}{eps}{.bb}{}%
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}{}%
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}{}%
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{}%
% changed or new graphic rules
\DeclareGraphicsRule{.eps.zip}{eps}{.eps.bb}{`unzip -p #1}% zipped EPS
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}% gzipped ←
EPS
\DeclareGraphicsRule{.jpg}{eps}{}{`convert #1 eps:-}% JPEG
\DeclareGraphicsRule{.gif}{eps}{.bb}{`convert #1 eps:-}% GIF
\DeclareGraphicsRule{.png}{eps}{.bb}{`convert #1 eps:-}% PNG
\DeclareGraphicsRule{.tif}{eps}{.bb}{`convert #1 eps:-}% TIFF
\DeclareGraphicsRule{.pdf}{eps}{.bb}{`convert #1 eps:-}% PDF- ←
graphics
```

You will find this file, for example in Debian, at `/etc/texmf/latex/graphics.cfg`. The best way to proceed is to copy this file to your local `texpath` and then change this copy. See your $\text{T}_\text{E}\text{X}$ distribution manual to learn how to get a list of your $\text{T}_\text{E}\text{X}$ folders.

With this configuration file you are able to insert bitmap graphics and zipped or gzipped EPS files in $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$. The command for conversion is given by `dvips`. When you look at the conversion command you will see that no extra file is created. The result of the conversion process is directly piped into the PS file. The only thing $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ must know is the size of the graphics, and therefore we need the bounding box, which is provided by Kile.

Some say that this way is insecure; you have to decide on how to work. In any case, you need no bounding box, as Kile will extract this information from all types of graphics.

10.4.4 PDF $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ and EPS Graphics

As already stated, PDF $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ is not able to handle EPS graphic files, but converters like `epstopdf` will help. The best way is to include package `epstopdf`, which must follow the **graphicx** package.

```
\usepackage[pdftex]{graphicx}
\usepackage{epstopdf}
```

Now you can already include EPS graphics, if you run `pdflatex` with option `--shell-escape`, but we can make it even better and also handle zipped or gzipped EPS files. Again we have to change the graphics configuration file `graphics.cfg` as above. This time we search for:

```
% pdfTeX is running in pdf mode
\ExecuteOptions{pdftex}%
```

and simply add some lines:

```
% pdfTeX is running in pdf mode
\ExecuteOptions{pdftex}%
\AtEndOfPackage{%
```

```

\g@addto@macro\Gin@extensions{.eps.gz,.eps.zip}%
\@namedef{Gin@rule@.eps.gz}#1{{pdf}}{.pdf}{`gunzip -c #1 | epstopdf -f ←
>\Gin@base.pdf}}%
\@namedef{Gin@rule@.eps.zip}#1{{pdf}}{.pdf}{`unzip -p #1 | epstopdf -f ←
>\Gin@base.pdf}}%
}%

```

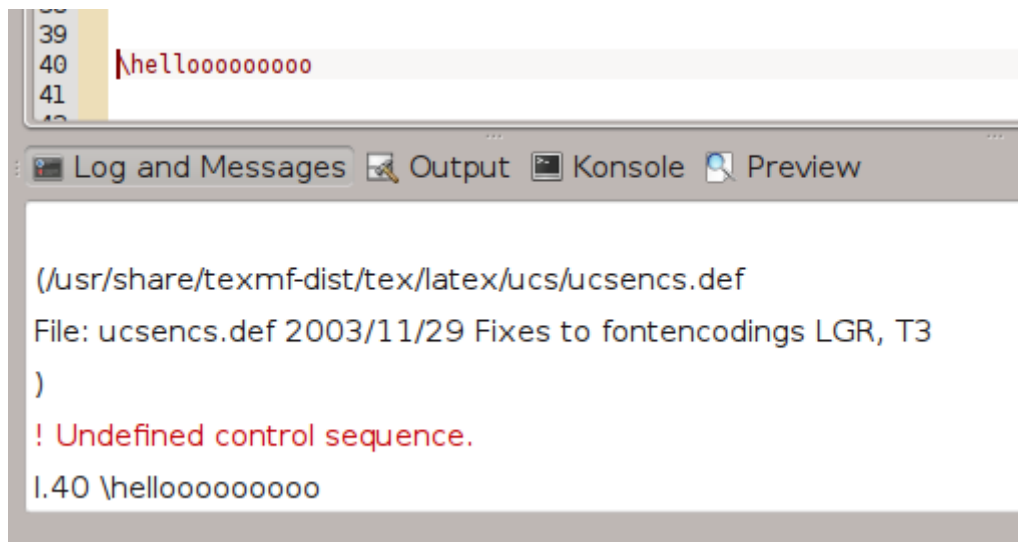
With these lines, PDFL^AT_EX is able to handle EPS files, and hopefully there should be no more issues concerning graphics.

10.5 Master Document

Defining your document as a master allows you to work with separate files, which gives you a parent document (or Master document), and child documents that make up a complete work. After having defined your Master document, with the corresponding command in the **Settings** menu, all the commands of the **Tools** menu will apply only to this document, even when you are working on the child documents. You can even close the Master document.

10.6 Error Handling

After you have compiled something, Kile takes a look at the error messages that were generated. If there are any errors or warnings, they will be briefly reported in the **Log and Messages** window. One can take a closer look at the messages by selecting **Build** → **View Log File**, or by using the keyboard shortcut **Alt-0**. The generated log is then displayed in the **Log and Messages** view; errors and warnings are highlighted.



Viewing the log

You can easily jump from one message in the log file to another by using the **Build** → **Next / Previous LaTeX Error / Warning** menu items, or by using the corresponding toolbar buttons.

To jump to the line in the L^AT_EX source where the error or warning occurred, click on the error or warning in the **Log and Messages** view. Kile will take you automatically to the offending line.

10.7 The Watch File Mode

When you launch the **Quickbuild** command, a viewer of some sort will normally be called after the compilation. If you are not using an embedded viewer, a new window will be opened every time.

If you are adjusting the look of your document, you might launch **Quickbuild** very often, and have many viewer windows open on your desktop; to avoid this confusion, you can activate the **Watch file** mode, that will prevent **Quickbuild** from launching a viewer.

This mode is of course useless with the embedded viewers, as you have to close them anyway to get back to editing the document and recompiling.

Chapter 11

Navigating the L^AT_EX Source

11.1 Using the Structure View

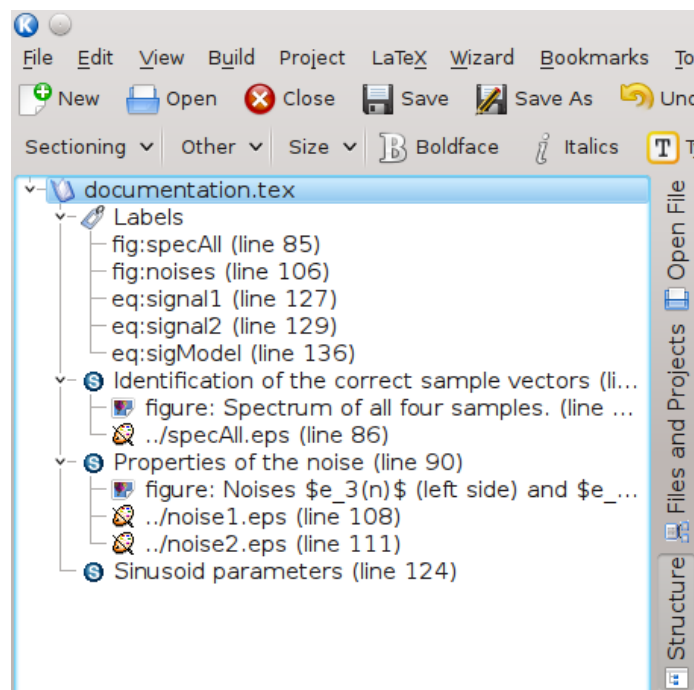
The **Structure** view shows the hierarchy of the document being created in Kile, and allows you to quickly navigate it, showing its segmentation. To navigate around your document, all you need to do is to left click on any label, chapter, section, subsection, etc., and you will be taken to the beginning of the corresponding area.

If you included a separate L^AT_EX file in your source using the `\input` or `\include` tags, these files will be referred to in the **Structure** view; double-clicking on their names will make Kile bring up the included file in the editor window.

The hierarchy tree also has a separate branch for labels used in the text.

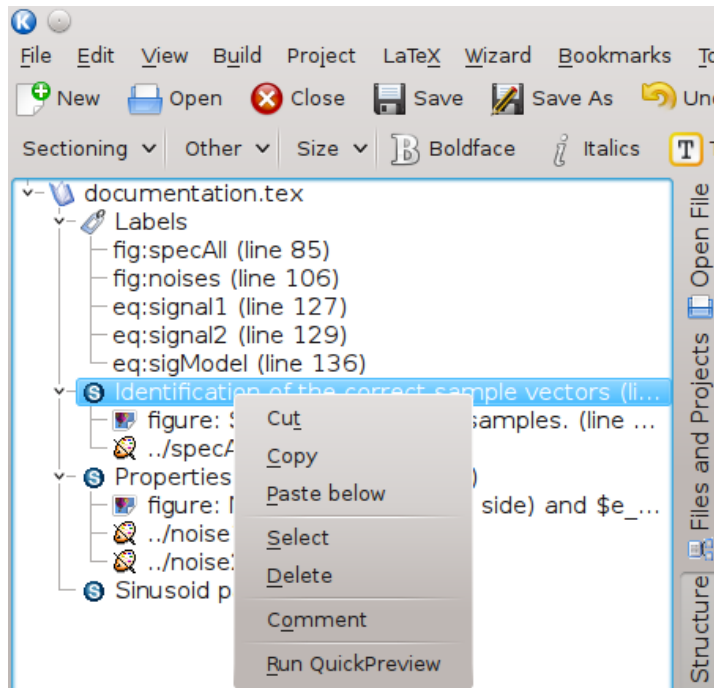
11.1.1 Using the Context Menu

Most of the entries in the structure view have a lot of entries in the context menu, which you can open with a right mouse click. So look at the structure view in the following picture.

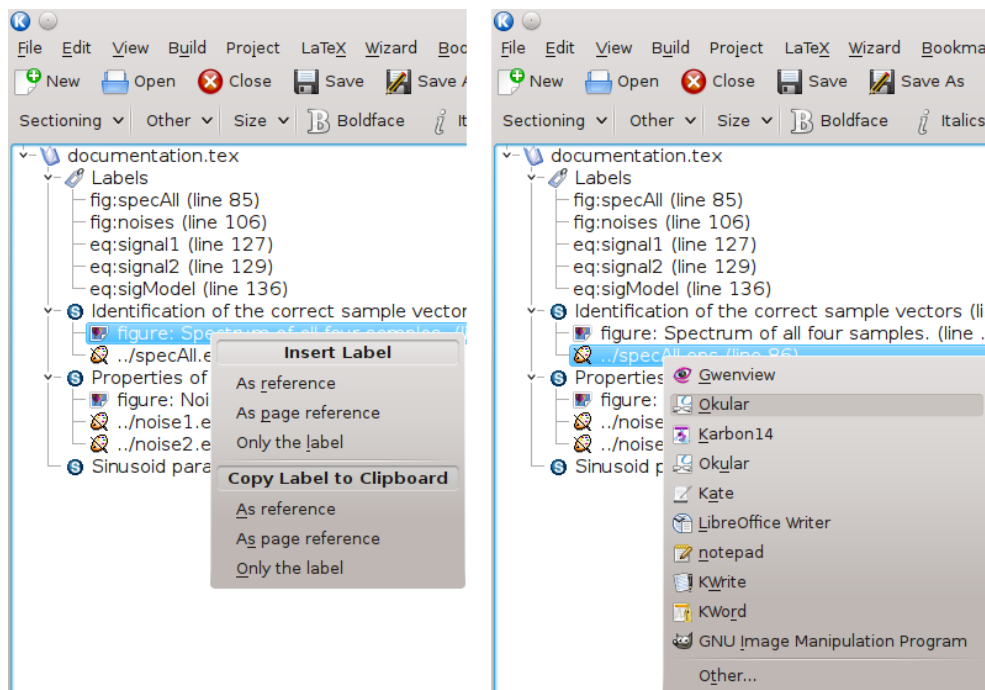


The Kile Handbook

You will find some labels, some sectioning entries, two figure environments and three pictures. If you right click on one of the sectioning entries, a menu with some useful commands will popup. All commands like **Select**, **Delete** or **Comment** will work with all lines, which belong to this section.



Clicking on a figure or table entry will offer some actions with respect to references and a right click on a graphics entry will offer some programs to open the graphics.



11.1.2 Updating the Structure View

To update your structure view you can either go to **Edit** → **Refresh Structure**, hit **F12**, or you can save your document, which will make Kile update its **Structure** view.

11.2 Bookmarks

Bookmarks are your reference to a segment of text or a line inside the Kile environment. To use a bookmark, select a specific line of your document you would like to return to, then press **Ctrl-B**, and Kile will add a bookmark to this line. Alternatively, you can also set a bookmark by highlighting a line and choosing the menu labeled **Bookmarks** → **Set Bookmark**.

To remove all your bookmarks, select **Bookmarks** → **Clear All Bookmarks**.

Chapter 12

Projects

12.1 Working with Projects

In Kile you can create and work with *projects*. A project is a group of L^AT_EX, graphic, BibT_EX, or other files that contain all the information that is used to build your complete document. A typical project would be a document consisting of several chapters, written in different `.tex` files; all of them could be included in a project, to make the whole document easier to manage. The specifications of the project are stored in a special file, with extension `.kilepr`.

A Project adds the following functionalities:

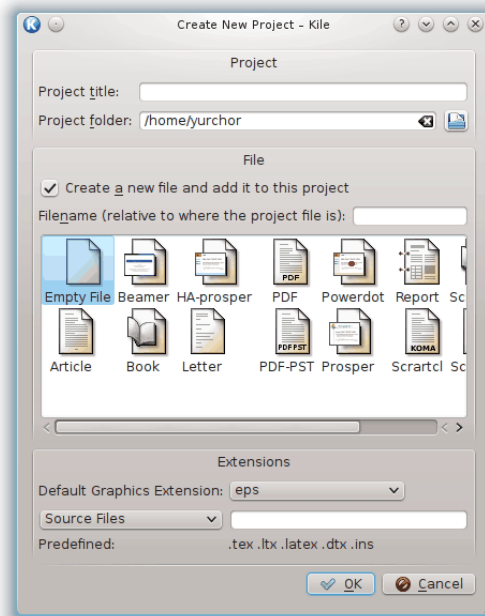
- You need not set a master document, Kile does this automatically.
- Project files can easily be archived together by going to **Project** → **Archive**
- The **Files and Projects** view shows which files are included in the project.
- After opening a project, any file that was previously opened will be restored with the original encoding and highlighting.
- Code completion works across all project files.
- Reference completion works across all project files.
- Citation completion works across all project files.
- Search in all project files.
- Specify custom quickbuild and makeidx command.

You can find all project related commands in the **Project**-menu. From there you can open, close and manage your projects.

12.2 Creating a Project

To create a project, select **Project** → **New Project...**

The Kile Handbook

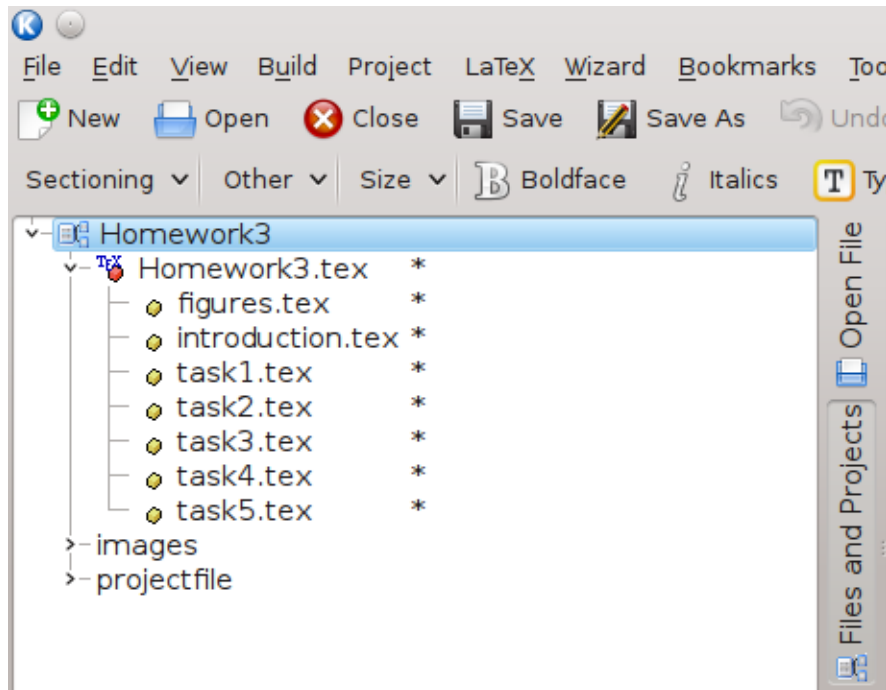


You will be asked to give the following information to create your project:

- Title of your project (**Project title** text field).
The title of the project will be used to create a name of the project file by transforming to lowercase and adding `.kilepr` extension.
- A folder where the project file will be stored (**Project folder** text field).
- If you want to create a new main file of the project check the **Create a new file and add it to this project** item.
- When you fill out the **Filename** box, you have to include a relative path from where the `.kilepr` project file is stored (see the **Project folder** item).
- Type of the created file, **Empty File**, **Article**, **Book**, **Letter**, **Report**, etc. can be chosen from a visual list at the bottom of the **File** panel.
- Extensions for the default files in the project can be selected using the **Extensions** panel. Your choice will be used to define the files that should be opened when you choose **Project** → **Open All Project Files** menu item and in the Kile wizards. The extensions in the text field should be separated with spaces.

12.3 The Files and Projects View

The **Files and Projects** view is a button of the sidebar menu. From this view, you can see the structure of your project, its files, and the name of the `.kilepr` file that stores the project information. Adding, removing, or changing options in your project is done via the **Files and Projects** view.

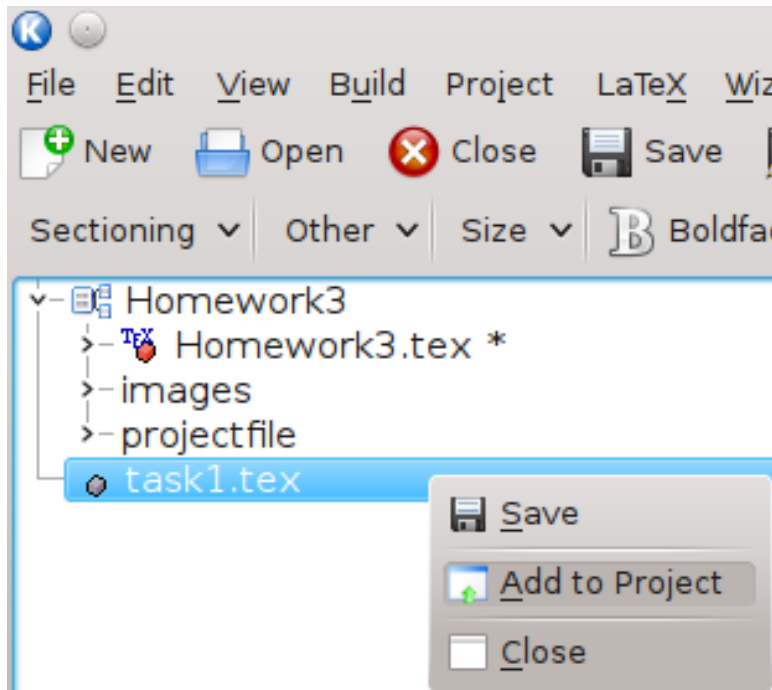


The Files and Projects View

12.4 Adding and Removing Files

To add a file to your project, open any $\text{T}_\text{E}\text{X}$ file, right click on its name in the **Files and Projects** view, and select **Add to Project**. If you have opened multiple projects, a dialog box will pop up in which you can specify to which project the file should be added.

You can also right-click on the project's name in the **Files and Projects** view, and select **Add Files...** to bring up a file selection dialog.



Adding a file to a project

To remove a file from a project, right-click on it and select **Remove File**. This does *not* delete your file (and also does not close it), but only removes it from the list of files contained in the `.kilepr` extension.

12.4.1 Archiving your Project

Kile allows you to easily backup your project by storing all its files into a single archive (often known as a *tarball*). To archive your project, right-click on its name in the **Files and Projects** view, or select **Project** → **Archive**.

By default, all files in a project are added to the archive. If you do not want to include a certain file in the archive, right-click on it in the **Files and Projects** view, and uncheck the **Include in Archive** option.

The archive operation is currently realized by executing the `tar` from the project folder (where the `.kilepr` file is located).

12.5 Project Options

Kile has a few options related to your project that can be set. To change them, right-click on the title of your project and select **Project Options**, and you will have the option of changing:

- The title of your project.
- Default file extensions.
- The Master document.
- The Quickbuild command.
- The `makeidx` options.

12.6 Closing a Project

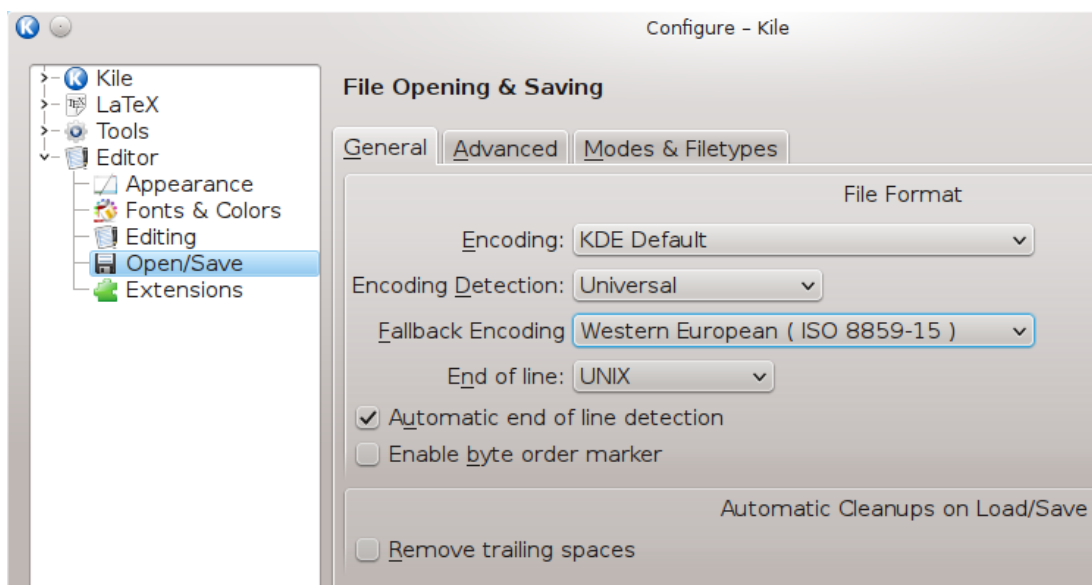
To close a project, select the **Files and Projects** view from the vertical toolbar, right click on your project title, and then select **Close**. This will close your project, all the files associated with your project, and will also add the name of the project you just closed to **Open Recent Project...** in the **Project** menu.

Chapter 13

Document Encoding

The Kile editor allows you to read, convert and save text in the character encoding your document needs. With this it is possible, for example, to use accented characters, such as are commonly used in Italian or French, directly in L^AT_EX documents. Selecting the encoding for your document can be done in two ways:

- One way to set the document encoding is by using the submenu **Settings** → **Configure Kile...** → **Editor**, where you can set the default character encoding for all files.



- A second way to set the encoding for a document is to choose the desired encoding within the wizard to create a new document.

L^AT_EX itself only understands ASCII, which represents a very limited set of characters. Hence, it is not possible to use accented characters directly. To use them nevertheless, a special syntax was invented: such as for example `\~e` for `ë`. The `inputenc` package is available to help you with this. It is included in the preamble using `\usepackage[latin1]{inputenc}`, where the optional argument specifies the encoding you would like to use (nowadays in most cases `utf8`). This tells L^AT_EX to translate all of the `ë`'s you wrote to `\~e`'s before compiling. Please refer to the `inputenc` documentation directly for more information. Last but not least: remember to make sure that your file is *actually* saved in the same encoding you specified for the `inputenc` package!

This multitude of different character coding tables has created numerous problems: for example, you cannot write a course of Turkish in French without losing one language’s special characters. There is general agreement that, sooner or later, everybody will switch to [Unicode](#). There are many implementations of Unicode, and UTF-8 is the most successful in Linux®; Windows®(R) relies instead on the more cumbersome and less flexible UCS-2. Most distributions have already begun setting their default encoding to UTF-8, and therefore you may be very interested in using the `utf8` argument to the `inputenc` package.

13.1 The ucs Package

If you don’t have the ucs package installed, you can proceed as follows:

- Get the ucs package from [CTAN](#).
- To install it, unpack the downloaded file and place it in a folder listed in your `$TEXINPUTS` environment variable. This can also be set inside Kile.

```
\usepackage{ucs}
\usepackage[utf8]{inputenc}
```

13.2 XeLaTeX

If you are using **XeLaTeX**, you can simply load the `xltxtra` package. It will additionally load all the required packages.

```
\usepackage{xltxtra}
```

13.3 CJK Support

Adding support for ideographic languages is quite tricky. However, once you are done with it, it will work quite well. Other than installing packages, there is some extra configuration work to do.

TIP

Your Linux® distribution might already have a CJK (Chinese, Japanese, Korean) package ready for you, so you might be saved the hassle of manually installing everything. Do check before going forward!

There is the possibility of using the ucs package in order to write short snippets of CJK text, but that option is seriously limited as it does not handle, among other things, newlines. Instead, we will install the complete CJK- $L^A T_E X$ package and make it work for both $L^A T_E X$ and $P D F L^A T_E X$. A lot of this material has been inspired by [Pai H. Chou’s page about how to setup PDFLATEX](#).

1. Download the [CJK](#) package. Copy its unpacked files to an appropriate subfolder of `$TEXMF`, just as you did with the ucs package before (see Section 13.1). The files will be unpacked in a `CJK/X_Y.Z` folder; it is not important that you take them out, though it will probably be tidier and easier for you to maintain.

- Now you have to download a font that supports all the CJK characters you need. You can choose any *.ttf file that covers them, but in this walkthrough we will use [Cyberbit](#). Unzip the file and rename `Cyberbit.ttf` to `cyberbit.ttf`, since uppercase might confuse your system.

Place `cyberbit.ttf` in a folder together with [Unicode.sfd](#), and generate the *.tfm and *.enc files with the command `$ ttf2tfm cyberbit.ttf -w cyberbit@Unicode@`. For some reason, sometimes this does not produce the hundreds of files it should. Should that happen in your case, you can download both *.tfm and *.enc files.

Place the *.tfm files in an appropriate folder, say `$TEXMF /fonts/tfm/bitstream/cyberbit/`; the *.enc files may be installed in `$TEXMF /pdftex/enc/cyberbit/`.

- Now we need a map file to connect the *.enc files to the font. Download [cyberbit.map](#) and install it in `$TEXMF /pdftex/config/`.
- Download another file, [c70cyberbit.fd](#), and place it in an appropriate folder. You may choose, for example, `$TEXMF /tex/misc/`.
- The last file we have to generate is a PostScript[®] Type 1 font, necessary to read DVI files generated with L^AT_EX. Run the command `$ ttf2pfb cyberbit.ttf -o cyberbit.pfb`, and copy the resulting `cyberbit.pfb` to a folder such as `$TEXMF /fonts/type1/cyberbit/`.
- Let's now place `cyberbit.ttf` among the fonts where L^AT_EX can find it. You could place it in a folder named `$TEXMF /fonts/truetype/`.
- Check the configuration file you find at `$TEXMF /web2c/texmf.cnf`, and make sure that the line mentioning `TTFONTS` is uncommented and points to the folder where you saved `cyberbit.ttf`.
- To make it possible for PDFL^AT_EX to use your CJK fonts, it is necessary that you add a line in the configuration file `$TEXMF /pdftex/config/pdftex.cfg`. Add `map +cyberbit.map` in the file to complete the configuration for PDFL^AT_EX.
- To configure L^AT_EX so that you can produce DVI files with CJK characters, you have to add a line in file `ttfonts.map`. The file might be in a folder named `$TEXMF /ttf2pk/`, but you will probably have to look for it. Append the line `cyberbit@Unicode@ cyberbit.ttf` to it.
- Now, you only have to run `texhash` and the system should be ready.

To test whether your configuration is correct, you can try to compile [this test file](#).

13.3.1 CJK Troubleshooting

There are many things that can go wrong when adding CJK support manually. If something seems not to work, the following checklist might help you.

- Obviously, since you run L^AT_EX as a user and not as root, you must *allow* ordinary users to access the new files. Make sure all folders and files are accessible using the `chmod` command.
- If L^AT_EX writes a DVI without problems, but you cannot view it, it is almost certainly because of some problems in the automatic generation of *.pk fonts. They are supposed to be generated on the fly when viewing a DVI file, but this might fail for a number of reasons: double-check `ttfonts.map` for your custom line first. However, it might happen that your `ttf2pk` command, which is usually invoked by the DVI viewer, has been compiled *without* support for the `kpathsea` libraries. If this is the case, `ttf2pk --version` will make no mention of `kpathsea`. As support for these libraries is necessary, you might have to find a new package, or recompile FreeType 1 yourself.

13.3.2 How do I input CJK in Unicode?

There are a number of different input engines, and the choice can depend also on personal preference. The author uses [Skim](#), a port to KDE of the [Scim](#) engine. Refer to your distribution's documentation to learn how to install these programs. Configuration of such programs can be tricky too, in the case of Skim you will have to define an environment variable `XMODIFIERS=@im=SCIM` before starting X.

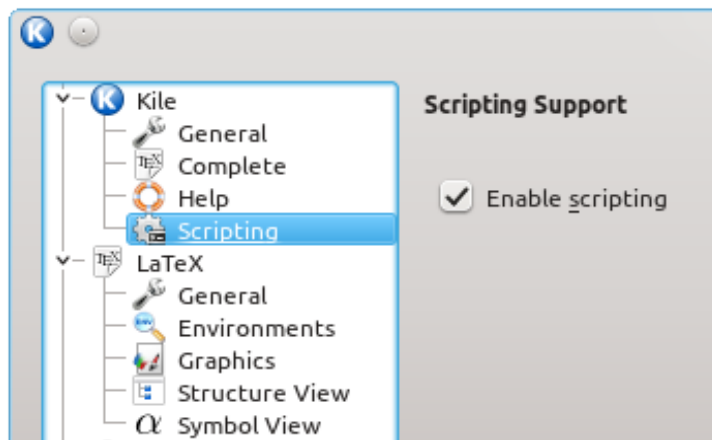
Chapter 14

Scripting

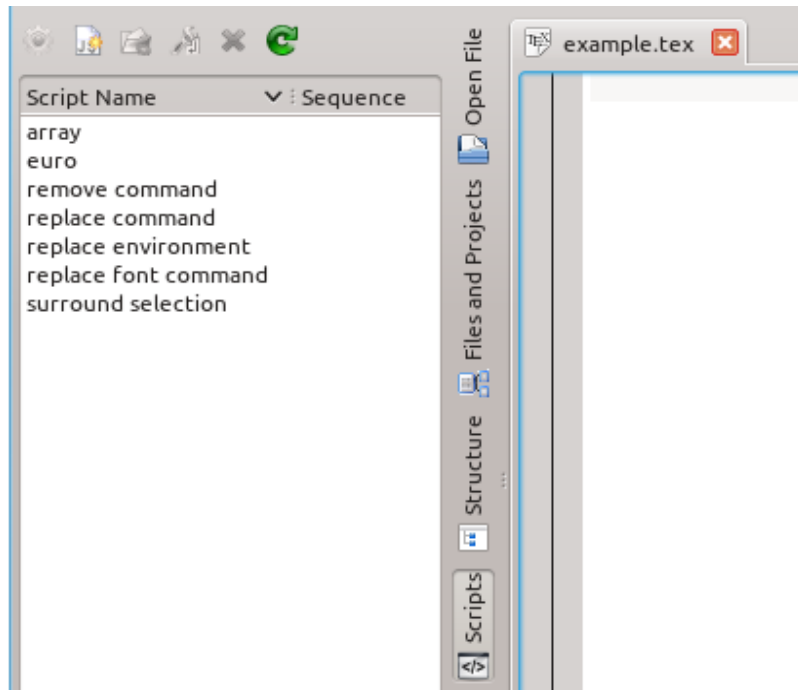
14.1 Scripting in Kile

Kile's scripting feature allows the execution of [ECMAScript](#) code, widely known as JavaScript. You will find a lot of tutorials, which provide information about objects (variables), functions and properties supported by JavaScript.

Scripting support can be enabled in the configuration dialog of Kile: **Settings** → **Configure Kile...** → **Kile+Scripting**.



If scripting is enabled, an additional scripting panel is visible in the sidebar, where scripts can be managed:



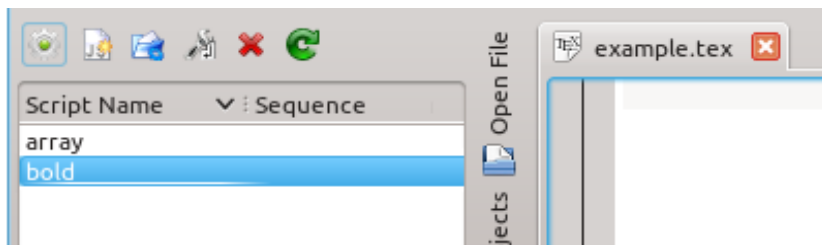
This widget contains six icons, which offer different tasks:

- Run the selected script.
- Create a new script.
- Open the selected script in the editor.
- Configure a key sequence for the selected script.
- Remove an assigned key sequence.
- Refresh the list of available scripts, which are all found in `$KDEDIR /share/apps/kile/scripts/`.

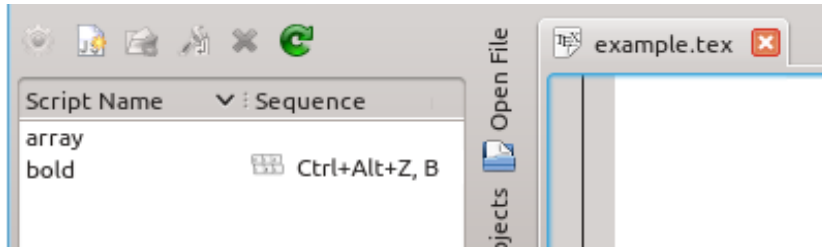
14.2 Executing a Script

You can execute a script in three different ways:

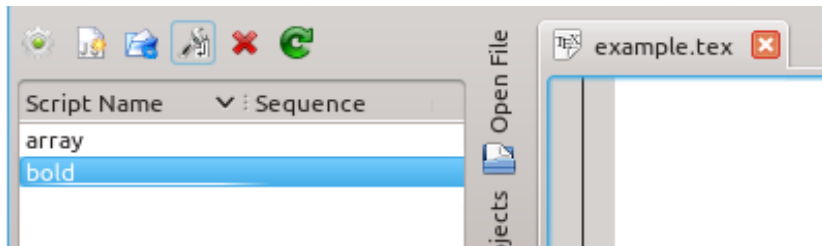
1. Select the desired script and click on the **Execute** button on the left side of the script management widget.



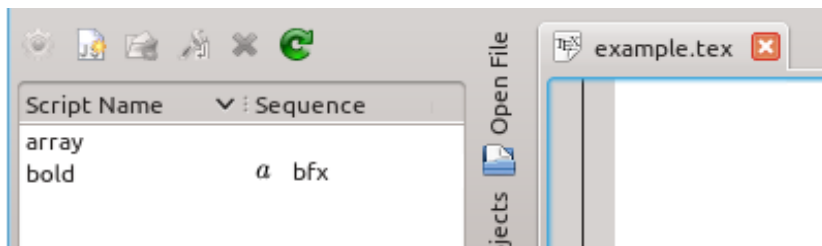
2. Use a keyboard shortcut.



You can assign a keyboard shortcut to a script using the **Configure** button in the script management widget.



3. Use an editor key sequence. The script will be executed, if you type the assigned key sequence in the editor.



This method can be extended to a rather sophisticated kind of code completion. Let us assume that you have written a script, which simply inserts the L^AT_EX command `\textbf{ }` into the current document.

```
document.insertText ("\\textbf{%C}");
```

If you now type the assigned key sequence `bfx` in your text document, this key sequence will be removed and the script will be executed. It will insert `\textbf{ }` and the cursor is placed between the braces.



What a comfortable and powerful method of code completion.

14.3 API Reference

The scripting API presented here is available in all scripts. Before the contents of a script is loaded, Kile first adds several prototypes and functions to the scripting context. This convenience

The Kile Handbook

API contains prototypes like text cursors and text ranges and is located in the folder `KILE_APP_DIR/script-plugins/`.

Kile scripts differ slightly from *Kate* scripts, which use another design, as they also can be started from the command line. But all functions of the *Kate* scripting API are also available in Kile's scripting API, so porting JavaScript code from *Kate* to Kile should be very easy. But as Kile is a very rich featured L^AT_EX editor, its own scripting API offers many more possibilities than *Kate*'s one.

Remark: Description of API calls, which are also available in *Kate* scripting, have been taken from *Kate*'s documentation.

14.3.1 Global Functions

This section lists global functions.

`void debug(String text);`

Prints *text* to `stdout` in the console. The printed text is colored to distinguish it from other debug output.

14.3.2 The Cursor Prototype

As Kile is a text editor, all the scripting API is based on cursors and ranges whenever possible. A `Cursor` is a simple `(line, column)` tuple representing a text position in the document.

`Cursor();`

Constructor: Returns a `Cursor` at position `(0, 0)`.

Example: `var cursor = new Cursor();`

`Cursor(int line, int column);`

Constructor: Returns a `Cursor` at position `(line, column)`.

Example: `var cursor = new Cursor(3, 42);`

`Cursor(Cursor other);`

Copy constructor: Returns a copy of the cursor *other*.

Example: `var copy = new Cursor(other);`

`Cursor Cursor.clone();`

Returns a clone of the cursor.

Example: `var clone = cursor.clone();`

`bool Cursor.isValid();`

Check whether the cursor is valid. The cursor is invalid if line and/or column are set to `-1`.

Example: `var valid = cursor.isValid();`

Cursor Cursor.invalid();

Returns a new invalid cursor located at (-1, -1).

Example: `var invalidCursor = cursor.invalid();`

int Cursor.compareTo(Cursor other);

Compares this cursor to the cursor *other*. Returns

- -1, if this cursor is located before the cursor *other*,
- 0, if both cursors equal and
- +1, if this cursor is located after the cursor *other*.

bool Cursor.equals(Cursor other);

Returns `true`, if this cursor and the cursor *other* are equal, otherwise `false`.

String Cursor.toString();

Returns the cursor as a string of the form `Cursor(line, column)`.

14.3.3 The Range Prototype

As Kile is a text editor, all the scripting API is based on cursors and ranges whenever possible. As `Cursor` is a simple `(line, column)` tuple representing a text position in the document, a `Range` spans text from a starting cursor position to an ending cursor position.

Range();

Constructor: Calling `new Range()` returns a `Range` at `(0,0) - (0,0)`.

Range(Cursor start, Cursor end);

Constructor: Calling `new Range(start, end)` returns the range from cursor *start* to cursor *end*.

Range(int startLine, int startColumn, int endLine, int endColumn);

Constructor: Calling `new Range(startLine, startColumn, endLine, endColumn)` returns the `Range` from `(startLine, startColumn)` to `(endLine, endColumn)`.

Range(Range other);

Copy constructor: Returns a copy of `Range other`.

Range Range.clone();

Returns a clone of the range.

Example: `var clone = range.clone();`

The Kile Handbook

bool Range.isValid();

Returns `true`, if both start and end cursor are valid, otherwise `false`.

Example: `var valid = range.isValid();`

bool Range.invalid();

Returns the Range from (-1,-1) to (-1,-1).

bool Range.contains(Cursor cursor);

Returns `true`, if this range contains the cursor position, otherwise `false`.

bool Range.contains(Range other);

Returns `true`, if this range contains the Range *other*, otherwise `false`.

bool Range.containsColumn(int column);

Returns `true`, if *column* is in the half open interval [`start.column`, `end.column`], otherwise `false`.

bool Range.containsLine(int line);

Returns `true`, if *line* is in the half open interval [`start.line`, `end.line`], otherwise `false`.

bool Range.overlaps(Range other);

Returns `true`, if this range and the range *other* share a common region, otherwise `false`.

bool Range.overlapsLine(int line);

Returns `true`, if *line* is in the interval [`start.line`, `end.line`], otherwise `false`.

bool Range.overlapsColumn(int column);

Returns `true`, if *column* is in the interval [`start.column`, `end.column`], otherwise `false`.

bool Range.equals(Range other);

Returns `true`, if this range and the Range *other* are equal, otherwise `false`.

String Range.toString();

Returns the range as a string of the form `Range(Cursor(line, column) - Cursor(line, column))`.

14.3.4 The View API

Whenever a script is being executed, there is a global object (variable) **view** representing the current active editor view. All functions of `view` work with cursor positions or selected text. The following is a list of all available **view** functions.

void view.backspace();

Programmatically performs the equivalent of pressing the backspace key.

Cursor view.cursorPosition();

Returns the current cursor position in the view.

**void view.setCursorPosition(int line, int column); void
view.setCursorPosition(Cursor cursor);**

Set the current cursor position to either `line, column` or to the given `cursor`.

void view.cursorLeft();

Moves the cursor one position backward in the text.

void view.cursorRight();

Moves the cursor one position forward in the text.

void view.cursorUp();

Moves the cursor one line up in the document.

void view.cursorDown();

Moves the cursor one line down in the document.

int view.cursorLine();

Returns the line which the cursor is currently located at.

int view.cursorColumn();

Returns the column which the cursor is currently located at.

void view.setCursorLine(int line);

Set the cursor line to the given `line`.

void view.setCursorColumn(int column);

Set the cursor column to the given `column`.

The Kile Handbook

Cursor view.virtualCursorPosition();

Get the current *virtual* cursor position. *Virtual* means the tabulator character (TAB) counts *multiple* characters, as configured by the user (e.g. one TAB is 8 spaces). The virtual cursor position provides access to the user visible values of the current cursor position.

bool view.hasSelection();

Returns `true`, if the view has selected text, otherwise `false`.

String view.selectedText();

Returns the selected text. If no text is selected, the returned string is empty.

Range view.selectionRange();

Returns the selected text range. The returned range is invalid if there is no selected text.

void view.setSelection(Range range);

Set the selected text to the given *range*.

void view.selectAll();

Selects the entire text in the document.

void view.clearSelection();

Clears the text selection without removing the text.

void view.removeSelectedText();

Remove the selected text. If the view does not have any selected text, this does nothing.

void view.selectLine();

Selects the text in the current line.

void view.selectLine(int line);

Selects the text in the given *line*.

void view.selectLines(int from, int to);

Selects the entire text from line *from* to line *to*.

void view.selectWord();

Selects the current word. If no word is found at the current cursor position, nothing is done.

```
void view.selectLatexCommand();
```

Selects the current L^AT_EX command. If no command is found at the current cursor position, nothing is done.

```
void view.selectEnvironment(bool inside = false);
```

Selects the entire text of the current L^AT_EX environment. If *inside* is false, the environment text including the surrounding L^AT_EX tags `\begin{...}\end{...}` will be selected, else without these tags. If no parameter is given, *inside* is set to false.

```
void view.selectTexgroup(bool inside = true);
```

Selects the text of the current L^AT_EX group. If *inside* is true, only the texgroup without the surrounding braces will be selected. If no parameter is given, *inside* is set to true.

```
void view.selectMathgroup();
```

Selects the text of the current math group.

```
void view.selectParagraph(bool wholeLines = true);
```

Selects the entire text of the current L^AT_EX paragraph. If *wholeLines* is true, the first and the last lines of the paragraph will be included in the selection entirely (including the end-of-line character); otherwise, the selection will only contain non-whitespace characters.

14.3.5 The Document API

Whenever a script is being executed, there is a global object (variable) **document** representing the current active document. The following is a list of all available **document** functions.

```
void document.insertText(String text);
```

Inserts the *text* at the current cursor position.

```
void document.insertText(int line, int column, String text); void  
document.insertText(Cursor cursor, String text);
```

Inserts the *text* at the given cursor position.

```
bool document.removeText(int fromLine, int fromColumn, int toLine, i-  
nt toColumn); bool document.removeText(Cursor from, Cursor to); bool  
document.removeText(Range range);
```

Removes the text in the given range. Returns true on success, or false, if the document is in read-only mode.

```
bool document.replaceText(Range range, String text);
```

Replace the text of the given range with the specified text.

The Kile Handbook

int document.lines();

Returns the number of lines in the document.

int document.length();

Returns the number of characters in the document.

Range document.documentRange();

Returns a range which encompasses the whole document.

Cursor document.documentEnd();

Returns the current cursor position of the document's end.

String document.text();

Returns the entire content of the document in a single text string. Newlines are marked with the newline character `\n`.

String document.text(int fromLine, int fromColumn, int toLine, int toColumn); **String document.text(Cursor from, Cursor to);** **String document.text(Range range);**

Returns the text in the given range. It is recommended to use the cursor and range based version for better readability of the source code.

bool document.setText(String text);

Sets the entire document text.

bool document.clear();

Removes the entire text in the document.

String document.line();

Returns the current text line as string.

String document.line(int line);

Returns the given text line as string. The string is empty if the requested line is out of range.

int document.lineLength();

Returns the length of the current line.

int document.lineLength(int line);

Returns the *line*'s length.

The Kile Handbook

bool document.insertLine(*String* s);

Inserts text in the current line. Returns `true` on success, or `false`, if the document is in read-only mode or the line is not in the document range.

bool document.insertLine(*int* line, *String* s);

Inserts text in the given line. Returns `true` on success, or `false`, if the document is in read-only mode or the line is not in the document range.

bool document.removeLine();

Removes the current text line. Returns `true` on success, or `false`, if the document is in read-only mode.

bool document.removeLine(*int* line);

Removes the given text line. Returns `true` on success, or `false`, if the document is in read-only mode or the line is not in the document range.

bool document.replaceLine(*String* text);

Replace the text of the current line with the specified text.

bool document.replaceLine(*int* line, *String* text);

Replace the text of the given line with the specified text.

bool document.truncateLine();

Truncate the current line at the given column or cursor position. Returns `true` on success, or `false` if the given line is not part of the document range.

bool document.truncate(*int* line, *int* column); bool document.truncate(*Cursor* cursor);

Truncate the given line at the given column or cursor position. Returns `true` on success, or `false` if the given line is not part of the document range.

String document.word();

Returns the word at the current cursor position. If no word is found at this cursor position, the returned string is empty.

String document.wordAt(*int* line, *int* column); String document.wordAt(*Cursor* cursor);

Returns the word at the given cursor position. If no word is found at this cursor position, the returned string is empty.

The Kile Handbook

Range document.wordRange ();

Returns the range of the word at the given cursor position. If no word is found, `Range.invalid()` is returned, which can be tested with `Range.isValid()`.

String document.latexCommand ();

Returns the L^AT_EX command at the current cursor position. If no command is found at this cursor position, the returned string is empty.

**String document.latexCommandAt (int line, int column); String
document.latexCommandAt (Cursor cursor);**

Returns the L^AT_EX command at the given cursor position. If no command is found at this cursor position, the returned string is empty.

Range document.latexCommandRange ();

Returns the range of the L^AT_EX command at the given cursor position. If no L^AT_EX command is found, `Range.invalid()` is returned, which can be tested with `Range.isValid()`.

**String document.charAt (int line, int column); String document.charAt (C-
ursor cursor);**

Returns the character at the given cursor position.

String document.firstChar (int line);

Returns the first character in the given *line* that is not a whitespace. The first character is at column 0. If the line is empty or only contains whitespace characters, the returned string is empty.

String document.lastChar (int line);

Returns the last character in the given *line* that is not a whitespace. If the line is empty or only contains whitespace characters, the returned string is empty.

**bool document.isSpace (int line, int column); bool document.isSpace (Curs-
or cursor);**

Returns `true`, if the character at the given cursor position is a whitespace, otherwise `false`.

void document.insertBullet ();

Insert a Kile *bullet*. Remember that you can easily jump to the next or previous bullet. This will also highlight this bullet so that it will be deleted automatically, when you enter your first letter.

void document.nextBullet ();

Jump to the next bullet in the text if there is one.

void document.previousBullet ();

Jump to the previous bullet in the text if there is one.

bool document.hasEnvironment ();

Returns `true` if a surrounding L^AT_EX environment is found, else `false`.

String document.environment (bool inside = false);

Returns the entire text of the surrounding L^AT_EX environment. If *inside* is `false`, the environment text including the surrounding L^AT_EX tags `\begin{...}\end{...}` will be returned, else without these tags. If no parameter is given, *inside* is set to `false`. If no environment is found, the returned string is empty.

Range document.environmentRange (bool inside = false);

Returns the range of the surrounding L^AT_EX environment. If *inside* is `false`, the range including the surrounding L^AT_EX tags `\begin{...}\end{...}` will be returned, else without these tags. If no parameter is given, *inside* is set to `false`. If no environment is found, `Range.invalid()` is returned, which can be tested with `Range.isValid()`.

String document.environmentName ();

Returns the name of the surrounding L^AT_EX environment or an empty string.

void document.removeEnvironment (bool inside = false);

Removes the text of the surrounding L^AT_EX environment. If *inside* is `false`, the environment text including the surrounding L^AT_EX tags `\begin{...}\end{...}` will be removed, else without these tags. If no parameter is given, *inside* is set to `false`.

void document.closeEnvironment ();

Insert a closing environment tag, if an opened L^AT_EX environment is found at the current cursor position.

void document.closeAllEnvironments ();

Insert closing environment tags for all opened L^AT_EX environments, which were found at the current cursor position.

bool document.hasTexgroup ();

Returns `true` if a surrounding L^AT_EX group is found at the current cursor position, else `false`.

String document.texgroup (bool inside = true);

Returns the text of the surrounding L^AT_EX group. If *inside* is `false`, the text of this L^AT_EX group including the surrounding braces `{...}` will be returned, else without them. If no parameter is given, *inside* is set to `false`. The returned string is empty, if no surrounding L^AT_EX group is found at the current cursor position.

Range document.texgroupRange(*bool* inside = true);

Returns the range of the surrounding L^AT_EX group. If *inside* is false, the range including the surrounding braces { . . . } will be returned, else without them. If no parameter is given, *inside* is set to false. If no group is found, Range.invalid() is returned, which can be tested with Range.isValid().

void document.removeTexgroup(*bool* inside = true);

Removes the text of the surrounding L^AT_EX group. If *inside* is false, the text of this L^AT_EX group including the surrounding braces { . . . } will be removed, else without them. If no parameter is given, *inside* is set to false.

bool document.hasMathgroup();

Returns true if a surrounding L^AT_EX mathgroup is found at the current cursor position, else false.

String document.mathgroup();

Returns the text of the surrounding L^AT_EX mathgroup. The returned string is empty, if no surrounding L^AT_EX mathgroup is found at the current cursor position.

Range document.mathgroupRange();

Returns the range of the surrounding L^AT_EX mathgroup. If there is no surrounding mathgroup, Range.invalid() is returned, which can be tested with Range.isValid().

void document.removeMathgroup();

Removes the text of the surrounding L^AT_EX mathgroup.

String document.paragraph();

Returns the text of the current L^AT_EX paragraph.

Range document.paragraphRange();

Returns the range of the surrounding L^AT_EX paragraph.

void document.removeParagraph();

Removes the text of the current L^AT_EX paragraph.

**bool document.matchesAt(*int* line, *int* column, *String* text); bool
document.matchesAt(*Cursor* cursor, *String* text);**

Returns true, if the given *text* matches at the corresponding cursor position, otherwise false.

The Kile Handbook

```
bool document.startsWith(int line, String pattern, bool skipWhiteSpaces = true);
```

Returns `true`, if the line starts with *pattern*, otherwise `false`. The argument *skipWhiteSpaces* controls whether leading whitespaces are ignored.

```
bool document.endsWith(int line, String pattern, bool skipWhiteSpaces = true);
```

Returns `true`, if the line ends with *pattern*, otherwise `false`. The argument *skipWhiteSpaces* controls whether trailing whitespaces are ignored.

```
int document.firstColumn(int line);
```

Returns the first non-whitespace column in the given *line*. If there are only whitespaces in the line, the return value is `-1`.

```
int document.lastColumn(int line);
```

Returns the last non-whitespace column in the given *line*. If there are only whitespaces in the line, the return value is `-1`.

```
int document.prevNonSpaceColumn(int line, int column); int document.prevNonSpaceColumn(Cursor cursor);
```

Returns the column with a non-whitespace character starting at the given cursor position and searching backwards.

```
int document.nextNonSpaceColumn(int line, int column); int document.nextNonSpaceColumn(Cursor cursor);
```

Returns the column with a non-whitespace character starting at the given cursor position and searching forwards.

```
int document.prevNonEmptyLine(int line);
```

Returns the next non-empty line containing non-whitespace characters searching backwards.

```
int document.nextNonEmptyLine(int line);
```

Returns the next non-empty line containing non-whitespace characters searching forwards.

```
void document.gotoBeginEnv();
```

Go to the start of a surrounding L^AT_EX environment.

```
void document.gotoEndEnv();
```

Go to the end of a surrounding L^AT_EX environment.

```
void document.gotoBeginTexgroup();
```

Go to the start of a surrounding L^AT_EX group.

```
void document.gotoEndTexgroup();
```

Go to the end of a surrounding L^AT_EX group.

```
void document.gotoNextParagraph();
```

Go to the next L^AT_EX paragraph.

```
void document.gotoPrevParagraph();
```

Go to the previous L^AT_EX paragraph.

```
void document.gotoNextSectioning();
```

Go to the next L^AT_EX section.

```
void document.gotoPrevSectioning();
```

Go to the previous L^AT_EX section.

```
void document.gotoLine(int line);
```

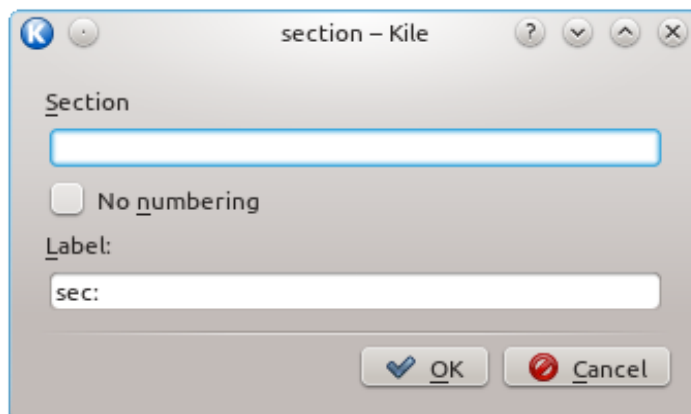
Go to the given line.

```
void document.insertChapter();
```

Insert a `\chapter` command (see also `document.insertSection()`).

```
void document.insertSection();
```

Insert a `\section` command. As with choosing the menu entry **LaTeX** → **Sectioning** → **section** a dialog will appear, where you can choose the title and an optional label for this sectioning command.



The Kile Handbook

void document.insertSubsection();

Insert a `\subsection` command (see also `document.insertSection()`).

void document.insertSubsubsection();

Insert a `\subsubsection` command (see also `document.insertSection()`).

void document.insertParagraph();

Insert a `\paragraph` command (see also `document.insertSection()`).

void document.insertSubparagraph();

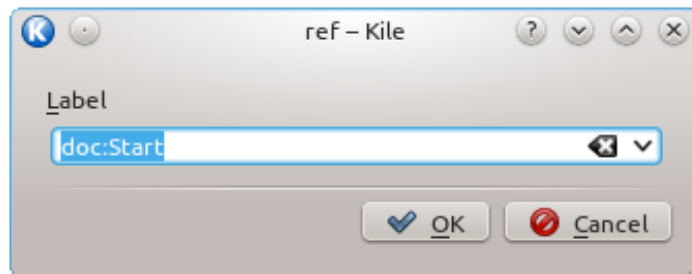
Insert a `\subparagraph` command (see also `document.insertSection()`).

void document.insertLabel();

Insert a `\label` command.

void document.insertReference();

Insert a `\ref` command. As with choosing the menu entry `LATEX → References → ref` a dialog will appear, where you can choose from already defined labels, which are listed in a combobox.



void document.insertPageref();

Insert a `\pageref` command (see also `document.insertReference()`).

void document.insertCitation();

Insert a `\cite` command.

void document.insertIndex();

Insert a `\index` command.

void document.insertFootnote();

Insert a `\footnote` command.

The Kile Handbook

void document.comment ();

Inserts comment markers to make the selection or current line a comment.

void document.uncomment ();

Removes comment markers from the selection or current line.

void document.uppercase ();

Put the selected text or the letter after the cursor in uppercase.

void document.lowercase ();

Put the selected text or the letter after the cursor in lowercase.

void document.capitalize ();

Capitalize the selected text or the current word.

void document.joinLines ();

Joins the lines of the current selection. Two succeeding text lines are always separated with a single space.

void document.insertIntelligentNewline ();

Insert a smart newline (see Section 5.11).

void document.insertIntelligentTabulator ();

Insert a smart tabulator (see Section 5.12).

void document.editBegin ();

Starts an edit group for undo/redo grouping. Make sure to always call `editEnd()` as often as you call `editBegin()`. Calling `editBegin()` internally uses a reference counter, i.e., this call can be nested.

void document.editEnd ();

Ends an edit group. The last call of `editEnd()` (i.e. the one for the first call of `editBegin()`) finishes the edit step.

StringList document.labelList ();

Returns all defined labels as a `StringList`, which can be used in JavaScript as an array of strings.

StringList document.bibitemList ();

Returns all defined bibitems as a `StringList`, which can be used in JavaScript as an array of strings.

void document.refreshStructure ();

Refresh the structure view (see chapter 11).

14.3.6 The Kile API

The global object (variable) **kile** is used to handle top level interactions with the outside world, input message and dialog interfaces. These API calls are divided into subobjects to structure this part of the scripting API. Conceptually **kile** is a bit like **window** in a browser API.

- `kile.alert`: message boxes
- `kile.input`: get user input
- `kile.wizard`: call one of Kile's wizards
- `kile.script`: get info about a running script
- `kile.file`: file operations like read and write.

14.3.6.1 Alert

```
void kile.alert.information(String text, String caption);
```

Display an *Information* dialog. *text* is the message string and *caption* the title of the message box. The default title is the script name.

```
void kile.alert.sorry(String text, String caption);
```

Display a *Sorry* dialog. *text* is the message string and *caption* the title of the message box. The default title is the script name.

```
void kile.alert.error(String text, String caption);
```

Display an *Error* dialog. *text* is the message string and *caption* the title of the message box. The default title is the script name.

```
String kile.alert.question(String text, String caption);
```

Display a simple *question* dialog. *text* is the message string and *caption* the title of the message box. The default title is the script name. The returned string is either *yes* or *no*.

```
String kile.alert.warning(String text, String caption);
```

Display a simple *warning* dialog. *text* is the message string and *caption* the title of the message box. The default title is the script name. The returned string is either *continue* or *cancel*.

14.3.6.2 Input

```
String kile.input.getListboxItem(String caption, String label, StringList list);
```

Function to let the user select an item from a list, which is shown as a listbox. *caption* is the text that is displayed in the title bar, *label* is the text that appears as the label for the list and *list* is the string list which is inserted into the list.

```
String kile.input.getComboboxItem(String caption, String label, StringList list);
```

Function to let the user select an item from a list, which is shown as a combobox. *caption* is the text that is displayed in the title bar, *label* is the text that appears as the label for the list and *list* is the string list which is inserted into the list.

```
String kile.input.getText(String caption, String label);
```

Function to get a string from the user. *caption* is the text that is displayed in the title bar and *label* the text that appears as a label for the line edit.

```
String kile.input.getLatexCommand(String caption, String label);
```

Function to get a L^AT_EX command from the user. This means that only lower- and uppercase letters are allowed. *caption* is the text that is displayed in the title bar and *label* the text that appears as a label for the line edit.

```
int kile.input.getInteger(String caption, String label, int min = INT_MIN, int max = INT_MAX);
```

Function to get an integer from the user. *caption* is the text that is displayed in the title bar. *label* is the text that appears as the label for the spin box. *min* and *max* are the minimum and maximum allowable values the user may choose. Default values are INT_MIN and INT_MAX.

```
int kile.input.getPosInteger(String caption, String label, int min = 1, int max = INT_MAX);
```

Function to get a positive integer from the user. *caption* is the text that is displayed in the title bar. *label* is the text that appears as the label for the spin box. *min* and *max* are the minimum and maximum allowable values the user may choose. Default values are 1 and INT_MAX.

14.3.6.3 Wizard

```
void kile.wizard.tabular();
```

Calls the *Tabular wizard*, which helps to write a tabular environment (see Section 7.3).

```
void kile.wizard.array();
```

Calls the *Array wizard*, which helps to write an array environment (see Section 7.3).

```
void kile.wizard.tabbing();
```

Calls the *Tabbing wizard*, which helps to write a tabbing environment (see Section 7.3).

```
void kile.wizard.floatEnvironment();
```

Calls the *Floats wizard*, which helps to insert floating elements (see Section 7.4).

```
void kile.wizard.mathEnvironment();
```

Calls the *Math wizard*, which helps to insert math environments (see Section 7.5).

```
void kile.wizard.postscript();
```

Calls the *Postscript Tools wizard*, which may help to manipulate or rearrange Postscript documents (see Section 7.6).

14.3.6.4 Script

String kile.script.name();

Returns the basename of a running script (without path and extension).

String kile.script.caption();

Returns a string, which can be used as a caption of alert boxes. It looks like **Script: scriptname.js**.

14.3.6.5 File

Object kile.file.read(String filename);

Read the contents of a text file. It is used like

Example: `var res = kile.file.read("~/path/to/file.txt");`

The return value **res** is an object (better: a map) with three properties:

- **status:** Gives the status code of the operation, which can be 0 (no error), 1 (access failed) or 2 (access denied). So, if no error occurred, the value of **res.status** or **res["status"]** will be 0.
- **result:** Contains the text of the given file.
- **message:** Contains an error message, if an error occurred.

Object kile.file.read();

The same as `read(filename)`, but no filename is given. A dialog will appear to select the file to read.

Object kile.file.write(String filename, String text);

Write the given text into a file. It is used like

Example: `var res = kile.file.write("~/path/to/file.txt", "Some text...");`

The return value **res** is an object (better: a map) with two properties: **status** and **message** (see `read()` for more information).

Object kile.file.write(String text);

The same as `write(filename, text)`, but no filename is given. A dialog will appear to choose a filename.

String kile.file.getOpenFileName(String startDir, String filter);

Creates a modal file dialog and return the selected filename or an empty string if none is chosen. Note that with this method the user must select an existing filename.

Parameters:

- **startDir:** Starting directory of the open dialog.
- **filter:** A shell glob or a mime-type-filter that specifies which files to display. Refer to the `KFileDialog` documentation for more information on this parameter.

Both parameters are optional. If you omit `filter`, all files will be displayed. If additionally `startDir` is omitted, the dialog will take the current document directory as starting point.

String kile.file.getSaveFileName(String startDir, String filter);

Creates a modal file dialog and returns the selected filename or an empty string if none is chosen. Note that with this method the user need not select an existing filename. See `getOpenFileName()` for an explanation of the parameters.

14.4 Examples

Some examples may help you to understand how to use the scripting API. These examples and some more are found in the scripting directory of Kile: `KILE_APP_DIR/scripts/`. Each script contains a short description.

14.4.1 Example 1: replace environment name

Replace a surrounding L^AT_EX environment with another, where the relative cursor position will not be changed. `\begin{abc}... \end{abc}` for example can be changed to `\begin{xyz}... \end{xyz}`.

```
var range = document.environmentRange(false);
if ( range.isValid() ) {
    var envname = kile.input.getLatexCommand("Enter Environment", "New ↔
        environment name:");
    if ( envname != '' ) {
        replaceEnvCommand(envname, range);
    }
}
else {
    kile.alert.sorry("No surrounding LaTeX environment found.");
}

function replaceEnvCommand(newEnv, r)
{
    var c = view.cursorPosition();

    var envname = document.environmentName();

    if ( envname != "" ) {
        var beginRange = new Range(r.start, new Cursor(r.start.line, r.start. ↔
            column+8+envname.length));
        var endRange = new Range(new Cursor(r.end.line, r.end.column-6-envname. ↔
            length), r.end);

        document.editBegin();
        document.replaceText(endRange, "\\end{"+newEnv+"}");
        document.replaceText(beginRange, "\\begin{"+newEnv+"}");
        document.editEnd();
    }
}
```

14.4.2 Example 2: replace a L^AT_EX font command

Replace a surrounding L^AT_EX font command with another font command, when the cursor is placed inside the texgroup. The relative cursor position will not be changed. `\textbf{abc}` for example can be changed to `\textit{abc}`.

```
var fontCommands = new Array("\\textbf", "\\textit", "\\textsl", "\\texttt",
    "\\textsc", "\\textrm", "\\textsf", "\\emph");

var range = document.texgroupRange(false);
if ( range.isValid() ) {
    replaceFontCommand(range);
}
```

The Kile Handbook

```
}
else {
  kile.alert.sorry("No surrounding TeX group found.");
}

function replaceFontCommand(r)
{
  var c = view.cursorPosition();

  document.editBegin();
  view.setCursorPosition(r.start);
  var cmd = document.latexCommand();
  var index = fontCommands.indexOf(cmd);
  if ( index >= 0 ) {
    var cmdRange = document.latexCommandRange();
    if ( cmdRange.isValid() ) {
      var newcommand = kile.input.getListboxItem("Choose",
                                                "Choose font command:", buildCmdList(cmd)) ←
      ;
      if ( newcommand != "" ) {
        document.replaceText(cmdRange, newcommand);
        c.column = c.column - (cmd.length - newcommand.length);
      }
    }
  }
  / view.setCursorPosition(c);
}
else {
  kile.alert.sorry("No surrounding font command found.");
}
document.editEnd();
}

function buildCmdList(current)
{
  var result = new Array();
  for ( i=0; i<fontCommands.length; ++i ) {
    if ( fontCommands[i] != current ) {
      result.push(fontCommands[i]);
    }
  }
  return result;
}
}
```

14.4.3 Example 3: surround selected text

Surround selected text with a TeX command, where the relative cursor position will not be changed. **abc** for example can be changed to **\texcommand{abc}**.

```
var range = view.selectionRange();

if ( range.isValid() ) {
  var cmd = kile.input.getLatexCommand("Choose", "Choose surrounding LaTeX ←
  command:");
  if ( cmd != "" ) {
    surroundTexCommand("\\\\"+cmd, range);
  }
}
}
```

The Kile Handbook

```
else {
    kile.alert.sorry("No selection found.");
}

function surroundTexCommand(cmd,r)
{
    var c = view.cursorPosition();

    document.editBegin();
    view.clearSelection();
    document.insertText(r.end,"}");
    document.insertText(r.start,cmd+"{");

    c.column = c.column + cmd.length + 2;
    view.setCursorPosition(c);
    document.editEnd();
}
```

Chapter 15

Help

15.1 Help Documents

L^AT_EX is a rather sophisticated system, where basic features can be expanded by a great variety of additional packages. Kile provides numerous different ways to aid the user.

The **Help** → **LaTeX Documentation** submenu includes links on documentation for all the included packages and an additional L^AT_EX reference.

Documentation Browser

A handy tool to browse all L^AT_EX topics. Please install L^AT_EX help packages for your distribution if you need the full-fledged documentation compendium.

LaTeX

A full unofficial reference for T_EX and friends. This is not only a description of all programs, but some important packages are also mentioned. It also includes a full reference manual of L^AT_EX commands—ideal for looking up a particular piece of formatting while writing a document. As this document is really extensive, it is referenced in Kile by three bookmarks.

LaTeX Commands

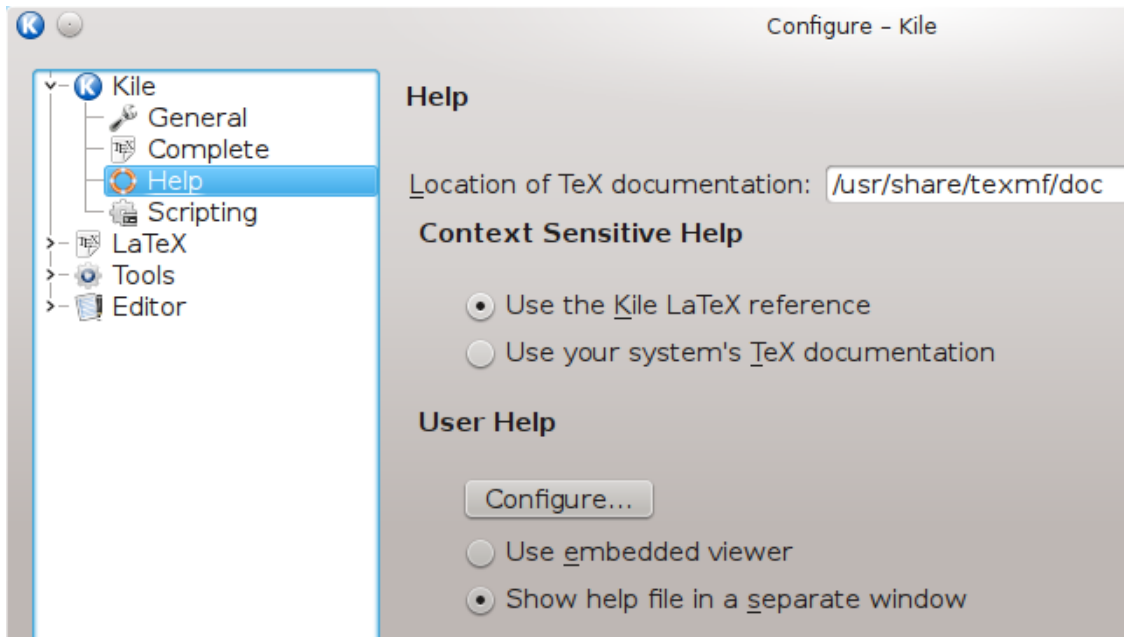
Another alphabetical index of the most common L^AT_EX commands.

LaTeX Environments

An alphabetical index of the most common L^AT_EX environments.

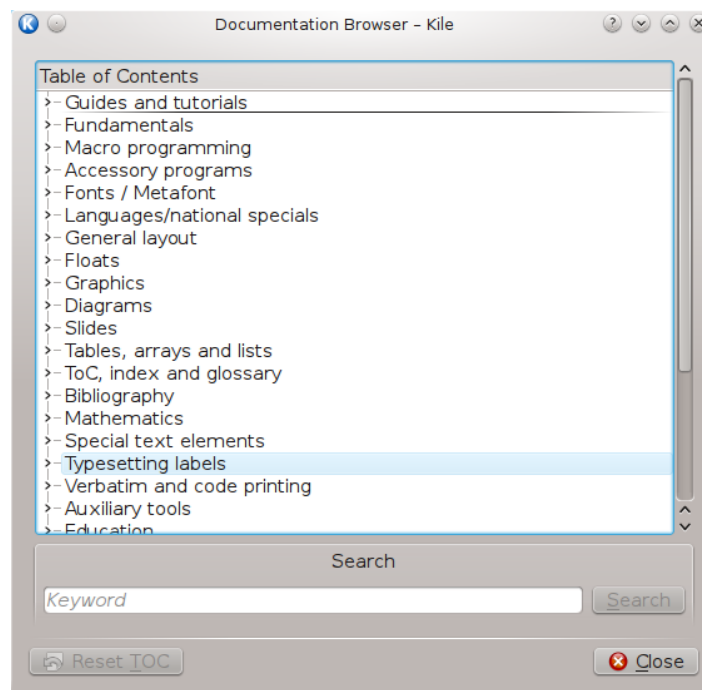
15.2 Context Sensitive Help

Kile also supports a context sensitive help, which is called with **Ctrl+Alt+H,K**. In **Settings** → **Configure Kile...** → **Kile+Help** you can choose whether you want to use Kile's L^AT_EX reference or the help system of t_EX/TeX Live, which is the default setting.

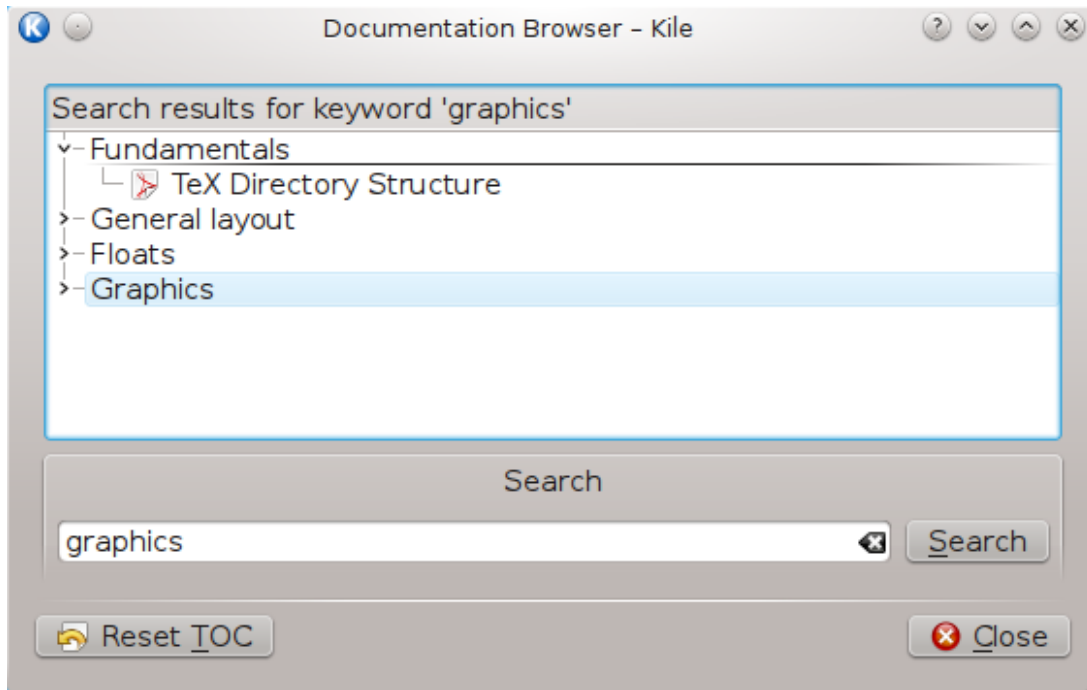


15.3 Searching for Keywords

It is not always easy to find the right document as teTeX / TeX Live ships with a huge number of help documents. In order to facilitate this process, teTeX / TeX Live provides a tiny program called `texdoctk`. It provides a database of all the help documents, for which Kile offers a user-friendly interface.



All the documents are grouped into categories. Additionally, one can search for package names or keywords. Kile will then show only the help documents matching the search string.



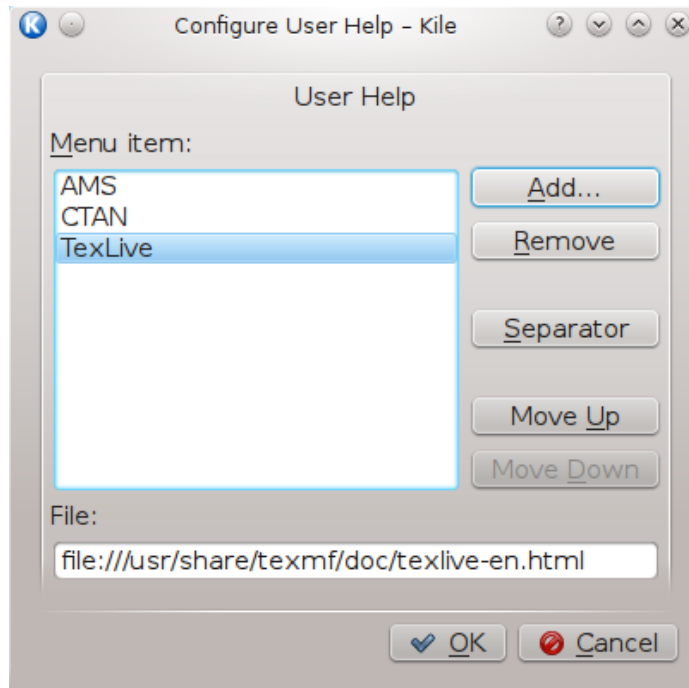
A mouse double-click or the **Space** key will start a viewer for the selected document. This can be an arbitrary document, not only a DVI, PS, PDF or HTML document. Kile takes the KDE settings into account in order to start an appropriate viewer.

15.4 User Defined Help

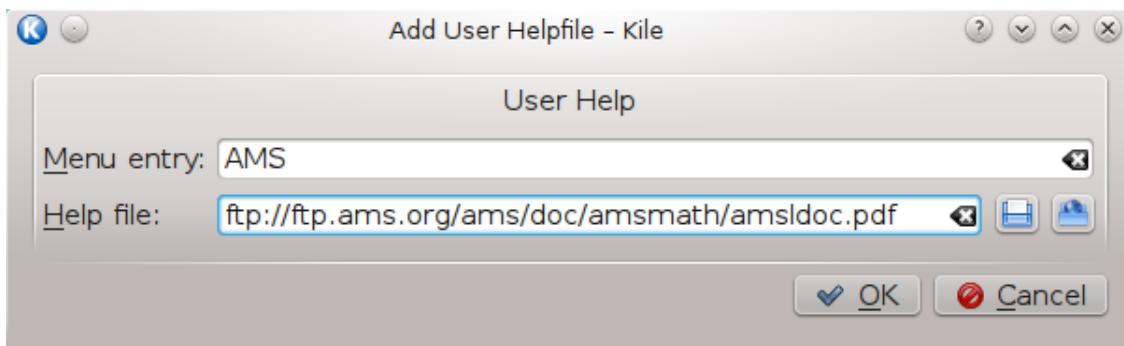
Besides this static $\text{tE}X/\text{TeX}$ Live documentation, Kile also supports a more flexible variable way for help documents. In the **Help** menu Kile has a special **User help** submenu, where the user can add documents of his or her own choice. These can be the most important documents of the $\text{tE}X/\text{TeX}$ Live documentation, or even self-written documents. It is also possible to enter URLs.

Go to **Settings** → **Configure Kile...** → **Kile+Help** and choose the **Configure** button to configure this **User help** menu. You can add, remove or move menu entries around, and insert separators to optimize the structure of the menu.

The Kile Handbook

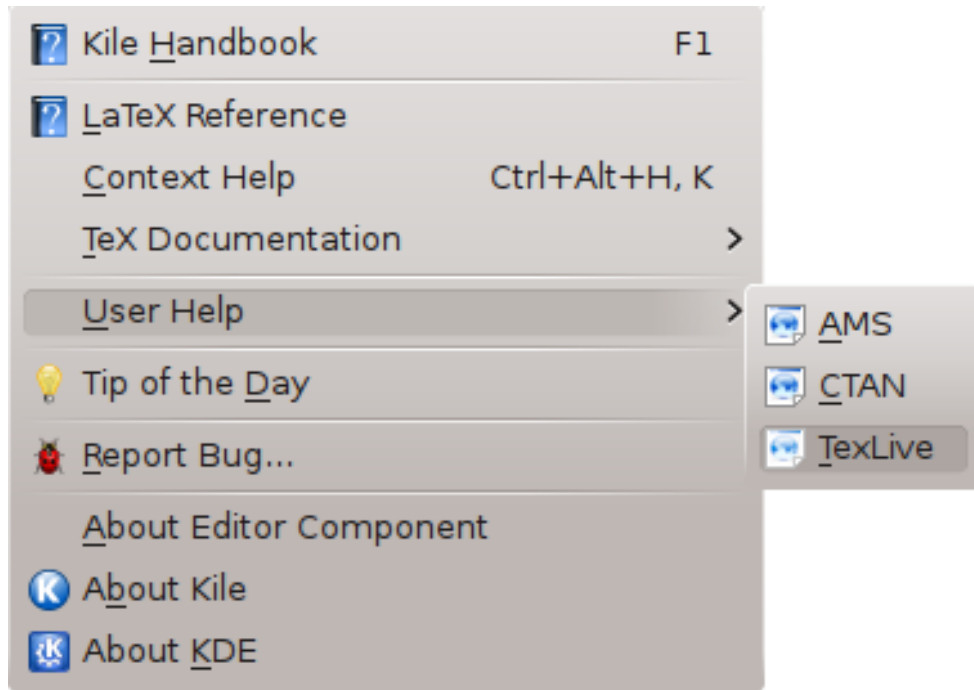


Pressing the **Add** button will open another dialog, where you can edit the name of the menu entry, and choose the corresponding file or URL. The second button to the right of the text field launches Konqueror, which can be used to determine the correct URL.



After finishing the configuration, all the entries will appear in the **Help** menu of Kile as a special menu entry **User help**.

The Kile Handbook



Chapter 16

Credits and License

Kile is an open-source user-friendly L^AT_EX / T_EX source code editor. It runs on systems that have the KDE Desktop Environment installed. KDE is available for several architectures including Linux[®] and other Unix-like systems.

Many thanks are owed to the people who strive to continue the Kile project and to those who sacrifice numerous hours of their time to develop tools we can all use under the GNU license. Up-to-date information about contributors can be found in the **About Kile** dialog from the **Help** menu.

Many thanks to all those involved!

This documentation is licensed under the terms of the [GNU Free Documentation License](#).

This program is licensed under the terms of the [GNU General Public License](#).