Customizing LyX: Features for the Advanced User

by the LyX Team\footnote{Principal maintainer of this file is Mike Ressler. If you have comments or error corrections, please send them to the LyX Documentation mailing list, \texttt{lyx-docs@lists.lyx.org}.}

July 17, 2006
## CONTENTS

4.3.1 Defining Own Keymaps: Keymap File Format ........ 26
4.4 International Keymap Stuff ....................... 28
  4.4.1 The .kmap File .................................. 28
  4.4.2 The .cdef File .................................. 29
  4.4.3 Dead Keys ........................................ 30
  4.4.4 Saving your Language Configuration ................. 30

5 Installing New Document Classes .................. 31
  5.1 Installing a new \TeX package .................. 31
  5.2 Layouts ........................................... 32
    5.2.1 Supporting new document classes ............. 33
    5.2.2 A layout for a sty file ..................... 33
    5.2.3 Layout for a cls file ...................... 34
  5.3 Declaring a new text class ...................... 34
    5.3.1 File format .................................. 35
    5.3.2 General text class parameters ............... 35
    5.3.3 ClassOptions section ....................... 37
    5.3.4 Specific Paragraph Layouts .................. 37
    5.3.5 Floats ........................................ 41
    5.3.6 Character styles ................................ 42
    5.3.7 Counters ..................................... 43
    5.3.8 Font description ................................ 43
    5.3.9 Upgrading old layout files .................. 44
  5.4 Creating Templates ................................ 44

6 Including External Material ....................... 45
  6.1 Background ....................................... 45
  6.2 How does it work? ................................ 46
  6.3 The external material dialog .................... 47
  6.4 Examples .......................................... 48
  6.5 The external template configuration file .......... 48
    6.5.1 The template header ......................... 50
    6.5.2 The Format section .......................... 50
    6.5.3 Preamble definitions ....................... 52
  6.6 The substitution mechanism ...................... 52
  6.7 Security discussion ................................ 54

7 The LyX Server ........................................ 57
  7.1 Introduction ...................................... 57
  7.2 Starting the LyX Server ......................... 57
  7.3 Normal communication ............................ 58
  7.4 Notification ....................................... 59
  7.5 The simple LyX Server Protocol .................. 59
A Bindings 61
A.1 Toolbar 61
A.2 Menu 62
A.2.1 File 62
A.2.2 Edit 63
A.2.3 Insert 64
A.2.4 Layout 65
A.2.4.1 Layout △ Character 66
A.2.5 View 67
A.2.6 Navigate 67
A.2.7 Help 67
A.2.8 Paragraph Style 67
A.3 Keyboard 69
A.3.1 Specific to emacs.bind 69
A.3.2 Specific to cua.bind 72
A.3.3 Specific to sciword.bind 75
A.3.4 Standard math bindings 80
A.3.5 Other Accelerators 81
Chapter 1

Introduction

This manual covers the customization features present in LYX. In it, we discuss issues like keyboard shortcuts, screen previewing options, printer options, sending commands to LYX via the LYX Server, internationalization, installing new \LaTeX{} classes and LYX layouts, etc. We can’t possibly hope to touch on everything you can change—our developers add new features faster than we can document them—but we will explain the most common customizations and hopefully point you in the right direction for some of the more obscure ones.
Chapter 2

LyX configuration files

This chapter aims to help you to find your way through the LyX configuration files. Before continuing to read this chapter, you should find out where your LyX library directory is by using Help ⊿ About LyX. This directory is the place where LyX places its system-wide configuration files, and we will simply name it LyXDir in the remainder of this document.

2.1 What’s in LyXDir?

LyXDir and its sub-directories contain a number of files and that can be used to customise LyX’s behaviour. You can change many of these files from within LyX itself through the Tools ⊿ Preferences dialog. Most customization that you might want to do to LyX is possible through this dialog. However, many other inner aspects of LyX can be customized by modifying the files in LyXDir. They fall in different categories, described in the following subsections.

2.1.1 Automatically generated files

These files are generated when you configure LyX. They contain various default values that are guessed by inspection. In general, it is not a good idea to modify them, since they might be overwritten at any time.

lyxrc.defaults contains defaults for various commands.

packages.lst contains the list of packages that have been recognized by LyX. It is currently unused by the LyX program itself, but the information extracted, and more, is made available with Help ⊿ LaTeX Configuration.

textclass.lst is the list of text classes that have been found in your layout/ directory, along with the associated LATEX document class and their description.
CHAPTER 2. \LaTeX\ CONFIGURATION FILES

doc/LaTeXConfig.lyx is automatically generated during configuration from the file \LaTeX\Config.lyx.in.

2.1.2 Directories

bind/ this directory contains files with the extension .bind that define the keybindings used in \LyX\ (see section 3.3). If there exists an internationalized version of the bind file named $\$\$LANG_xxx.bind, that will be used first. See Chapter 4, and section 3.3 for details.

clipart/ contains graphics files that can be included in documents.

doc/ contains \LyX\ documentation files (including the one you are currently reading). The file LaTeXConfig.lyx deserves special attention, as noted above. If there exists an internationalized version of the help-document with $\$\$LANG_ prepended to the name, that will be used first. See Chapter 4 for details.

examples/ contains example files that explain how to use some features. In the file browser, press the Examples button to get there.

images/ contains image files that are used by the Document dialog. In addition, it also contains the individual icons used in the toolbar and the banners that can be shown when \LyX\ is launched.

kbd/ contains keyboard keymapping files. See Chapter 4.4 for details.

layouts/ contains the text class files described in Chapter 5.

reLyX/ contains lots of files that together make up re\LyX, the translator of “well behaved” \LaTeX into \LyX.

scripts/ contains some files that demonstrate the capabilities of the External Template feature.

templates/ contains the standard \LyX\ template files described in Chapter 5.4.

tex/ contains some \LaTeX\ cls files distributed with \LyX.

ui/ contains files with the extension .ui that define the user interface to \LyX. That is, the files define which items appear in which menus and the items appearing on the toolbar. See Chapter 3.4 for details.

2.1.3 Files you don’t want to modify

These files are used internally by \LyX\ and you generally do not need to modify them unless you are a developer.

CREDITS this file contains the list of \LyX\ developers. The contents are displayed with the menu entry Help $\$\$Help $\$\$\$About \LyX.
2.2. YOUR LOCAL CONFIGURATION DIRECTORY

chkconfig.ltx this is a \LaTeX\ script used during the configuration process. Do not run directly.

configure this is the script that is used to re-configure LyX. It creates configuration files in the directory it was run from.

2.1.4 Other files needing a line or two...

encodings this contains tables describing how different character encodings can be mapped to unicode

external_templates this file contains the templates available to the new External Template feature.

languages this file contains a list of all the languages currently supported by LyX.

lyxrc.example Deprecated and definitely obfuscated. This is the old style preferences file. It will probably disappear in the near future.

2.2 Your local configuration directory

Even if you are using LyX as an unprivileged user, you might want to change LyX configuration for your own use. The UserDir directory contains all your personal configuration files. This is the directory described as “user directory” in Help → About LyX. This directory is used as a mirror of LyXDir, which means that every file in UserDir is a replacement for the corresponding file in LyXDir. Any configuration file described in the above sections can be placed either in the system-wide directory, in which case it will affect all users, or in your local directory for your own use.

To make things clearer, let’s provide a few examples:

- The preferences set in the Tools → Preferences dialog are saved to a file preferences in UserDir.

- When you reconfigure using Tools → Reconfigure, LyX runs configure and the resulting files are written in your local configuration directory (see section 3.9 to have a list of the preferences settings affected by this section). This means that any additional text class file that you might have added in UserDir/layouts will be added to the list of classes in the Layout → Document dialog.

- Similarly, if you have installed some \LaTeX\ document classes in your home directory, that \LaTeX\ can find with your TEXINPUTS path, they will show up in your list of text classes.\footnote{as long as LyX or yourself have a .layout file for it, of course.}
2.3 Running LYX with multiple configurations

The configuration freedom of the local configuration directory may not suffice if you want to have more than one configuration at your disposal. For example, you may want to use different key bindings or printer settings at different times. You can achieve this by having several such directories. You then specify which directory to use at run-time.

Invoking LYX with the command line switch `-userdir <some directory>` instructs the program to read the configuration from that directory, and not from the default directory (you can determine the default directory by running LYX without this switch as described above). If this directory does not exist, LYX offers to create it for you, just like it does for the default directory on the first time you run the program. You can modify the configuration options in this additional `Userdir` exactly as you would for the default directory. These directories are completely independent (but read on). Note that setting the environment variable `LYX_USERDIR_13x` to some value has exactly the same effect.

Having several configurations also requires more maintenance: if you want to add a new layout to `Userdir/layouts` which you want available from all your configurations, you must add it to each directory separately. You can avoid this with the following trick: after LYX creates the additional directory, most of the subdirectories (see above) are empty. If you want the new configuration to mirror an existing one, replace the empty subdirectory with a symbolic link to the matching subdirectory in the existing configuration. Take care with the `doc/` subdirectory, however, since it contains a file written by the configuration script (also accessible through `Tools>Reconfigure 3.9`) which is configuration-specific.
Chapter 3

The Preferences dialog

3.1 Using the dialog for the first time

The UserDir/preferences file will contain only changes that you have made to the default behaviour, some of which is hard-coded into LyX and some of which is contained in the system file LyXDir/lyxrc.defaults. Note that in both files lines beginning with a “#” are just comments and not interpreted. However, only system administrators should edit LyXDir/lyxrc. Users should use the Tools>Preferences dialog to create and modify their own UserDir/preferences file.

We hope that the Tools>Preferences dialog will be largely self-explanatory. Almost all the commands have an associated comment, so you shouldn’t have too much trouble modifying it to taste. Before we highlight a few of the more important commands below, however, a word of warning: Applying some of your changes (e.g., screen fonts) will have an instant effect. Others (e.g. changing the bind file) will not. If nothing appears to have changed, just Save the changes and restart LyX.

3.2 On-screen fonts

The font used to display your documents on the LyX screen is very important, since you’ll be reading all your documents with this font. Therefore it is important that the font is as readable and good-looking as possible. The LyX team tried to provide the best possible default font for you, but since practically all X11 systems are different, it’s likely that the default fonts will be sub-optimal on your system. Fortunately, you can do something about this. Before we explain how to do this, you should learn a bit more about fonts so that you are better prepared for choosing your fonts, because it is a trade-off that is specific to your preferences and the capabilities of your system.

Notice that this section only deals with the fonts on the screen inside the LyX window. The fonts that appear on the paper output are independent from
these fonts, and are determined by the document class. Read the User's Guide to learn how to change the font of the printed version of your document.

Basically, screen fonts come in two different kinds: scalable outline fonts and non-scalable bitmap fonts. This distinction seems a bit arbitrary, since non-scalable fonts are actually scalable in most modern font renderers. The difference lies in the quality of the scaling, and the speed of display. The most important decision is thus whether you should use non-scalable bitmap fonts or scalable outline fonts.

The scalable fonts are built from outlines of the single glyphs (i.e. characters) in the font. This means that each glyph is defined using mathematical curves that are well suited for scaling to any requested size. This mathematical definition is interpreted by the font renderer and turned into a small picture composed of pixels according to which size and glyph, the programmer requests. This means that scalable fonts will look pretty good in all sizes. Well, almost all sizes. Since scalable fonts are defined in an abstract way, it can be hard to provide a good rendering at small sizes, where each pixel has to be very carefully computed to provide a good image. Technically it is possible to do this from the mathematical definition, but in order to keep the rendering reasonably fast, tradeoffs have to be made, and the result is that scalable fonts can be difficult to read at small sizes.

Bitmap fonts on the other hand, are defined by bitmap graphics from the start, so they will look good at all the sizes they are meant for. However, they don’t scale well, because in order to scale a glyph, each pixel is enlarged into several pixels. It is the same effect that happens if you try to enlarge a picture in xv or any other picture manipulation program. In order to relieve this effect, bitmap fonts are typically provided in several fixed sizes typically from around 8 pixels high up to 34 pixels or so high in steps according to what is believed to be useful. The advantage of bitmap fonts is that no complicated computations are necessary to display each glyph, so bitmap fonts are thus faster displayed than scalable fonts. The disadvantage is that sizes that don’t exists as fixed versions have to be scaled by doubling pixels, and thus look bad.

The net result of all this, is that bitmap fonts are generally best for the small sizes, where they are available, while scalable fonts are generally best for large sizes. The logical conclusion would thus be to use bitmap fonts for the small sizes, and scalable fonts for the large sizes. Unfortunately, this is not a good idea, since bitmap fonts and scalable fonts are not designed to be used together, so the overall look of such a scheme would be bad. The best you can do is thus to try both schemes and decide for yourself what suits you.

By default, \LaTeX uses non-scalable bitmap fonts (when using the XForms frontend). For serif fonts, \textit{times} is used, for sans serif fonts, \textit{helvetica} is used, while \textit{courier} is used as the monospaced/typewriter font.

In the following, we will describe what to do if the text does not look good in \LaTeX. We'll start with the most important parameters: DPI and font zoom.
3.2. ON-SCREEN FONTS

3.2.1 DPI setting and Font Zoom

\LaTeX{} automatically tries to scale the fonts to look as close as the paper output size as possible, except for the so-called font zoom factor.

In order for this to work on all systems, it relies on the screen DPI (dots per inch) setting to be correct. The DPI setting for your system is autodetected by \LaTeX{} using the information the X server can provide. You can check what \LaTeX{} autodetects the DPI setting to, by running \LaTeX{} as
\begin{verbatim}
lyx -dbg 2
\end{verbatim}

On many systems, X is not set up correctly, so you should check that it is correct by hand. Run “\texttt{xdpyinfo | more}” and write down what the DPI is for the resolution you use (this will be close to the value \LaTeX{} detects). It is the number mentioned as “resolution”. Also write down the number of pixels you have in the width (the first number under “dimensions”).

Then get the good old ruler out of the closet, and measure the width of the visible screen-image on your monitor. Convert this measurement to inches if you used a centimeter ruler by dividing by 2.54. Now you can determine the correct DPI setting for your screen by dividing the number of pixels in the width by the width of the screen-image on the monitor. If this number is more than, say, 5 DPI from the detected value, you should either fix the X setup, or at least tell \LaTeX{} that the DPI is different than the detected value.

If you can’t fix the X setup (which of course is best since other programs than \LaTeX{} will benefit from this as well), you can tell \LaTeX{} the correct DPI using the Preferences dialog.

If the text is too small or too big for your taste, you should fiddle with the font zoom setting. This setting is used to scale the point size of the text. If your DPI setting is correct, and the font zoom setting is set to 100, this means that \LaTeX{} will try to display the text exactly the same size as it will appear on the paper-output. If you set the zoom factor to 200, the text will try to be 2 times as big as on paper. Of course, this will only happen if \LaTeX{} can find a font that has the appropriate size, which you can’t count on. Since \LaTeX{} is a WYSIWYM system anyways, this limitation isn’t much of an issue.

The default font zoom setting is 150, since a monitor is typically wider than a piece of paper, but you should try to fiddle with it through the Font Zoom setting in the Preferences dialog to find a size that you like. When you’ve found a setting that seems to work nicely for you (tip: use the Apply button to keep the dialog open while you experiment), you can make this setting the default by using the Save button.

While it is often possible to find a suitable size for the text on the screen, this doesn’t necessarily mean that the fonts are the best ones available on your system. In order to help you get the most out of your system, you can use the font definition commands to fine-tune the look of the text in greater detail than merely size.
3.2.2 Font definition commands

As mentioned, \textsc{lyx} uses non-scalable bitmap fonts by default with the XForms frontend. For serif fonts, \textit{times} is used, for sans serif fonts, \textit{helvetica} is used, while \textit{courier} is used as the monospaced/typewriter font.

You can change all of these from within the Preferences dialog. The number of fonts that are available on different systems vary, but the program \texttt{xfontsel} should be available everywhere. Use that program to find candidate fonts. When you’ve found a font that you like, try to insert the first two elements of the name (called “fndry” and “fmly” in \texttt{xfontsel}) in the appropriate field in the Preferences dialog and press Apply. \textsc{lyx} will then reformat your document using the new font, and if you like the font, you should Save it. One place to start for a new font is to see if the scalable font “utopia” is available. Tip: You can see whether a font is a bitmap font or a scalable font by checking the “resx” or “resy” fields in \texttt{xfontsel}. If the value 0 is available, the font is scalable. If the value 0 isn’t available, the font is a bitmap font.

Before you go about scrapping a bitmap font because the larger sizes look “blocky”, you should toggle the “Use scalable fonts” button. This is only useful if you use bitmap fonts, because only these don’t scale well. If you define this flag, \textsc{lyx} will only use the fixed font sizes that are available, and this guarantees that all bitmap fonts look well. (You can see which individual font sizes are available with the \texttt{xlsfonts} command. Try \texttt{man xlsfonts}.) However, the prize is that the difference between the size of the fonts on screen and the size of fonts on paper will be larger because \textsc{lyx} will have to be satisfied with the closest available size, and not try to scale a size to fit. Also, you can risk that some logically different sizes, such as Large and Larger, will be mapped to the same screen font, making it hard for you to see the difference on screen. We’ve decided not to use scalable fonts by default because of these artifacts, but since \textsc{lyx} is a WYSIWYM system, many people like to use the flag anyways, well-knowing that the font size on the screen can’t be trusted. But remember that this flag only makes a difference when you use bitmap fonts. Scalable fonts won’t be affected for reasons you should understand by now.

One final note regarding this flag: you should know that there is nothing wrong with using bitmap and scalable fonts at the same time for different purposes. For instance, it’s common to use the scalable “Utopia” for the serif text together with a bitmap version of “Helvetica”. And you can safely select the “Use scalable fonts” button without worries: It will only apply to the Helvetica font.

Sometimes the artifacts introduced by use of the flag can be relieved by using the fine-detail screen font sizes which defines which point sizes the different logical font sizes correspond to. Run \textsc{lyx} as \texttt{lyx -dbg 513} to see exactly what concrete fonts the logical sizes map to, and try adjusting the corresponding entries in the Preferences dialog until you’ve managed to hit the nail and get the fonts you want. This can be hard to do, because \textsc{lyx} uses the DPI setting and the font zoom settings to calculate which exact screen font size to ask the X server for, thus obfuscating the mapping. If you can’t make it by trial-and-error,
you can make the process more transparent if you set both the DPI setting and font zoom settings to 100—even when this is known to be wrong. This will of course make your scalable fonts look weird, so use with care.

3.2.3 Font encoding

By default, LyX will use fonts meant to write Western European text, including all kinds of English. This is defined through the so-called font encoding. If you want to use LyX to write for instance Eastern European text, Cyrillic or any other language not covered by the ISO-8859-1 font encoding, you can define a different one with the encoding setting. This requires you to have special fonts installed. You can use xfontsel to see whether this is the case: check the “rgstry” and “encdng” fields for ISO-8859-X values different from ISO-8859-1, and search for one that contains the national characters of your language. If you find any, enter this encoding in the dialog. If not, go searching the Web for appropriate fonts. For the Qt frontend, it’s recommended you use an iso646 font set.

When you’ve set LyX up to use a different font encoding, you should also consider changing the font used by dialog windows in LyX. For instance, the Table of Contents dialog will not be understandable unless you tell LyX to use a different font for this. By default the menu font is set to -*-helvetica-medium-r, but often Helvetica is not available in the font encoding you need, so the dialog allows this to be changed.

As you can see, there are quite a few options that can be used to fine tune the look of your fonts. This should not scare you from fiddling with the settings, because after all, you will hopefully be using LyX for many hours in the future. And contrary to real WYSIWYG word processors where you are tied to using fonts that have to look good both on paper and on screen, LyX gives you the possibility of using fonts that are designed to look good on the screen while using a different set of fonts to look good on paper.

3.3 Bindings

Bindings are used to, well, bind a function to a key. Several prepackaged binding files are available: a CUA set of bindings (familiar as the typical set of PC and CDE set of keyboard shortcuts), an Emacs set of bindings, for those of us who follow the One True Way and refuse to lower our standards, as well as specialty bindings (broadway and hollywood) and other languages (French, German, etc.).

If, however, you’d like to customise the keybindings to your own exacting tastes, then copy the best-fit file in LyXDir/bind/ to your own UserDir/bind/ and modify that. Don’t forget to load this new file into LyX using the Preferences dialog. (For the moment you’ll have to restart LyX for these changes to take effect.)

1I’m kidding here, of course!
\lyxsupportsinternationalizationoftheuserinterface(seeChapter4).If
your\emph{locale}isset,withtheenvironmentvariable\$\textsc{lang},\lyxwilltrytouse
bindfilesby prepending\$\textsc{lang}_\_to their name. Forexample, you can put athanslated copy of some standard bind file in your personal\texttt{bind/}
directory, and\lyxwill use it automatically.

Thesyntax of the \texttt{.bind} files is straightforward:
\begin{verbatim}
\bind <key combination> <lyx-function>
\end{verbatim}
Both key combination and lyx-function (including any arguments) must be
enclosed in "double quotes". All the\lyxfunctions are listed in the \emph{Reference
Guide}.

### 3.4 User Interface

The appearance of both the menu and toolbar may both be changed using the
\textbf{Preferences}dialog. Simply change the \texttt{.ui} file in\texttt{LyXDir/ui/}. For the moment, 
only one file exists, \texttt{default.ui}, but feel free to experiment. Just copy the file
to the\texttt{UserDir/ui/} directory and play! Note that, for the moment, you'll have
to restart\lyxfor these changes to take effect.

Thesyntax of the \texttt{.ui} files is straightforward: have a look at\texttt{default.ui}.
The\texttt{Menubar},\texttt{Menu} and\texttt{Toolbar} entries must be ended with an explicit\texttt{End}. 
They may contain\texttt{Submenus},\texttt{Items},\texttt{OptItems},\texttt{Separators},\texttt{Icons} and in the
case of the “file” menus, a\texttt{Lastfiles} entry. One small word of warning.
\texttt{Submenus} may be inserted in a\texttt{Menubar} or\texttt{Menu}, but they are defined as\texttt{Menus}, 
not as\texttt{Submenus}.

### 3.5 Converters, Formats, Viewers, Editors and 
Copiers

\lyxhas a powerful mechanism to convert to and from any file format using
external programs. Define a pair of formats, e.g. \texttt{LaTeX} and\texttt{PDF}. Now define
a converter from one format to the other. In our example, two possible
mechanisms exist.

1. A direct conversion, from\texttt{\LaTeX} to PDF using\texttt{pdflatex}

2. A more convoluted route using intermediate formats and converters: \texttt{\LaTeX}
to DVI (using latex) to PostScript® (using dvips) to PDF (using ps2pdf).

\lyxwill always choose the shortest possible route, so you must specify two
different Format names for \texttt{.pdf} files to be able to use either. Both are included
by default in the\textbf{Preferences} dialog. Have a look and then invent your own!

Moreover, each Format can have a Viewer associated with it. For example, 
you might want to use\texttt{ghostview} to examine PostScript® files, or \texttt{xdvi} to
preview the\texttt{\LaTeX} output. You can alter the viewer to use (and what options to
pass to it) via the\textbf{Tools} > \textbf{Preferences} > \textbf{Conversion} dialog. For example, to change
3.6 BibTeX and makeindex

Both the bibliography generating command (default `bibtex`) and the index generating command (default `makeindex` with options `-c` and `-q`) can be changed. As an alternative for `makeindex`, `xindy` can be recommended.

The command to enter is

```
makeindex.sh -m $$lang
```

where the placeholder `$lang` will be replaced by the chosen document (babel) language. For this, you must have installed the packages `xindy` and `make-rules (xindy-make-rules)`. Type `makeindex.sh` at a shell prompt for a help page.

3.7 ASCII export options

There are a couple of commands that can be used to “clean up” exported ASCII text files. Note that LyX automatically detects and uses the best settings for your system at installation time, but you can modify them if you disagree with its interpretation.

**ASCII roff** This option defines the command used to produce better ASCII tables with the `groff/troff/nroff` UNIX-commands (refer to their manpages for more information about them). Setting this as empty tells LyX to use the internal (inferior) formatter.

---

2 This can be an included `.tex` file, a verbatim included text file, external material or an included graphics file.

3 For example, the file may reference other files with relative filenames, which will become invalid in the temporary directory.
ASCII line length  With this command you can set the default line length of the ASCII output file. Setting it to 0 means endless lines.

3.8 Printer

There are a bunch of configuration options that are used for interaction with the external print command from LyX. Normally the defaults are fine: if, however, your print command takes different option names, you can modify them here.

3.8.1 Changing Colors

You can change the colors used by LyX on-screen using the new Preferences dialog. Alternatively, if you're feeling particularly perverse you could use the set-color bindable function (see the Reference Guide). Input would have the format:

```
set-color LyXName X11Color
```

Here is a (partial) list of the functions and default colors:

<table>
<thead>
<tr>
<th>LyX Name</th>
<th>Purpose</th>
<th>Default Color (X11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>text background</td>
<td>black</td>
</tr>
<tr>
<td>foreground</td>
<td>text foreground</td>
<td>linen</td>
</tr>
<tr>
<td>latex</td>
<td>LaTeX code</td>
<td>DarkRed</td>
</tr>
<tr>
<td>math</td>
<td>Mathed formulae</td>
<td>DarkBlue</td>
</tr>
<tr>
<td>mathline</td>
<td>fraction Lines, brackets, etc.</td>
<td>Blue</td>
</tr>
<tr>
<td>mathbg</td>
<td></td>
<td>AntiqueWhite</td>
</tr>
<tr>
<td>mathframe</td>
<td></td>
<td>Magenta</td>
</tr>
<tr>
<td>mathcursor</td>
<td></td>
<td>black</td>
</tr>
<tr>
<td>selection</td>
<td>selection background</td>
<td>LightBlue</td>
</tr>
</tbody>
</table>

3.9 The autodetected settings

There are several items that are detected for you when you run Tools > Reconfigure. In this section, we list those which pertain to the user preferences.

\ascii_roff_command uses either groff or nroff+tbl, depending on what is available.

\chktex_command is set to chktex plus a bunch of options.

\print_spool_command is set to lp on systems (so-called System V) who have this command, and lpr otherwise (BSD systems).

\print_spool_printerprefix is set to -d or -P, depending on whether lp or lpr was found.

\font_encoding is set to T1 if the ec fonts are found and E\TeX has support for these fonts built-in. You can set it manually if you only have the so-called dc fonts.
3.10 The rest

There are many other configuration options that can be used to customize \LaTeX\ behavior. We still need to document them here, but again, most should be fairly obvious. Please ask on the mailing lists if you need some more information; it may even prompt us to expand this section.
CHAPTER 3. THE PREFERENCES DIALOG
Chapter 4

Internationalizing LyX

LyX supports using a translated interface. Last time we checked, LyX provided text in 14 languages together with the default English text. The language of choice is called your locale. (For further reading on locale settings, see also the documentation for locale that comes with your operating system. For Linux, the manual page for locale(5) could be a good place to start).

Notice that these translations will work, but do contain a few flaws. In particular, all dialogs have been designed with the English text in mind, which means that some of the translated text will be too large to fit within the space allocated. This is only a display problem and will not cause any harm. Also, you will find that some of the translations do not define short-cut keys for everything. Sometimes, there are simply not enough free letters to do it. Other times, the translator just hasn’t got around to doing it yet. Our localization team – which you may wish to join – will try to fix these shortcomings in future versions of LyX.

4.1 Selecting an alternative language for the user interface

This feature is disabled by default, meaning that system default language will be used. To enable an alternative language, you have to set an appropriate environment variable. Use "setenv LANG xx" for csh class shells or "export LANG=xx" for sh class shells. Substitute the xx with the two letter code (or four letter code, like en_GB for British English) for the language you want. For instance, no is Norwegian. Besides the user interface texts being translated, also the appropriate manuals will be presented under the Help menu – if available.

On some systems, you may have to redefine LC_ALL or LC_MESSAGES instead of LANG, to override the system settings; their preference is in this order\(^1\), which

\(^1\)The shell variable LANGUAGE has been disabled in LyX for technical reasons. Don’t use it.
corresponds to the way GNU gettext does it. Consult your system documentation. Normally, you’ll want to put the appropriate line in a shell script run on start-up, so that the translation is on by default. Remember that this affects all localized packages, not only LyX!

If LyX is configured and compiled with “--disable-nls”, this mechanism will not work.

4.2 Translating LyX

4.2.1 Translating the graphical user interface (text messages).

LyX uses the GNU gettext library to handle the internationalization of the interface. To have LyX speak your favorite language in all menus and dialogs, you need a po-file for that language. When this is available, you’ll have to generate a mo-file from it and install the mo-file. The process of doing all of this is explained in the documentation for GNU gettext, but in short, this is what you do (xx denotes the language code):

- Copy LYX-SOURCE-DIR/po/lyx.pot to xx.po (if lyx.pot doesn’t exist, it can be remade with make lyx.pot in that directory, or you can use an existing po-file for some other language as a template).

- Edit xx.po. For some menu- and widget-labels, there are also shortcut keys that should be translated. Those keys are marked after a ‘|’, and should be translated according to the words and phrases of the xx-language. There is a tool named scgen.pl written in Prolog in LYX-SOURCE-DIR/development/tools/ that may be useful to help determine short-cut keys. Note that XForms (version 0.86 at least) can’t handle anything but 7-bit characters as shortcut keys. You should also fill also out the information at the beginning of the new po-file with your email-address, etc., so people know where to reach you with suggestions and entertaining flames.

- Generate xx.mo. This can be done with msgfmt -o xx.mo < xx.po

- Copy the mo-file to your locale-tree, at the correct directory for application messages for the language xx, and under the name lyx.mo (e.g. /usr/local/share/locale/xx/LC_MESSAGES/lyx.mo)

Adding a new po-file to the distribution of LyX involves altering the configure scripts and more, but the way gettext works, you don’t actually need the source-code of LyX to translate it—having lyx.pot (or an existing po-file) and the gettext tools suffices.

2We recommend that you use Emacs to do this, since the gettext distribution includes a nice mode that supports you in doing this.
If you’ve written a translation file for a language that LyX does not currently support, feel free to submit it for inclusion by sending a patch. In this case, we recommend that you read the README provided in the LYX-SOURCE-DIR/po/ directory for more instructions.

### 4.2.1.1 Ambiguous messages

Sometimes it turns out that one english message needs to be translated into different messages in the target language. One example is the message To which has the german translation Nach or Bis. gettext does not handle such ambiguous translations. Therefore you have to add some context information to the message: Instead of To it becomes To[[as in ‘From format x to format y’]] and To[[as in ‘From page x to page y’]]. Now the two occurrences of To are different for gettext and can be translated correctly to Nach and Bis, respectively.

Of course the context information needs to be stripped off the original message when no translation is used. Therefore you have to put it in double square brackets at the end of the message (see the example above). The translation mechanism of LyX ensures that everything in double square brackets at the end of messages is removed before displaying the message.

### 4.2.2 Translating the documentation.

The online documentation (in the Help-menu) can (and should!) be translated. If there are translated versions of the documentation available, and the locale is set accordingly, these will be used automagically by LyX. LyX looks for translated versions as LyXDir/doc/xx_DocName.lyx, where xx denotes the language as set by the environmental variable $LANG. If there are none, the default English versions will be displayed. Note that the translated versions must have the same filenames (DocName above) as the original. If you feel up to translating the documentation (an excellent way to proof-read the original documentation BTW!), there are a few things you should do right away:

- Read DocStyle.lyx, the guide to writing LyX documentation. Pay special attention to the translator’s section.
- Check out the documentation translation web page at The LyX Developer’s Web Site http://www.devel.lyx.org. That way, you can find out which (if any) documents have already been translated into your language. You can also find out who (if anyone) is organizing the effort to translate the documentation into your language. If no one is organizing the effort, please let us know that you’re interested.

Once you get to actually translating, here’s a few hints for you that may save you trouble:

---

As of February 2003, almost all of the docs have been translated into German and French. The Tutorial has been translated into at least 12 other languages, with other translations in progress. The library of translated documents is growing rapidly.
• Join the documentation team! There is information on how to do that in Intro.lyx (Help→Introduction), which by the way is the first document you should translate.

• Learn the typographic conventions for the language you are translating to. Typography is an ancient art and over the centuries, a great variety of conventions have developed throughout different parts of the world. Also study the professional terminology amongst typographers in your country. Inventing your own terminology will only confuse the users. (Warning! Typography is addictive!)

• Make a copy of the document. This will be your working copy. You can use this as your personal translated help-file by placing it in your ~/.lyx/doc/-directory.

• Sometimes the original document (from the LyX-team) will be updated. Use the ViewCVS tool available at http://www.lyx.org/viewcvs.cgi/lyxdoc/ to see what has been changed\textsuperscript{4}. That way you can easily see which parts of the translated document need to be updated.

• If you ever find an error in the original document, fix it and notify the rest of the documentation team of the changes! (You didn’t forget to join the documentation team did you?)

4.3 International Keyboard Support

[Editor’s Note: The following section is by Ivan Schreter. It needs to be fixed to conform to the new Documentation Style sheet and to make use of the new v1.0 features. The whole thing also needs to be merged with the section following it.-jw]

4.3.1 Defining Own Keymaps: Keymap File Format

Let’s look at a keyboard definition file a little closer. It is a plain ASCII file defining

• key-to-key or key-to-string translations

• dead keys

• dead keys exceptions

To define key-to-key or key-to-string translation, use this command:

\kmap key outstring

\textsuperscript{4}Alternatively, you can keep a copy of the latest version of the English document which you’ve translated.
where **key** is the key to be translated and **outstring** is the string to be inserted into the document. To define dead keys, use:

\texttt{\kmod key deadkey}

where **key** is keyboard key and **deadkey** is dead key name. The following dead keys are supported (shortcut name is in parentheses):

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>acute (acu)</td>
<td>áéíóú</td>
</tr>
<tr>
<td>grave (gra)</td>
<td>åëîöú</td>
</tr>
<tr>
<td>macron (mac)</td>
<td>ò</td>
</tr>
<tr>
<td>tilde (til)</td>
<td>ñN</td>
</tr>
<tr>
<td>underbar (underb)</td>
<td>0</td>
</tr>
<tr>
<td>cedilla (ced)</td>
<td>çÇ</td>
</tr>
<tr>
<td>underdot (underd)</td>
<td>0</td>
</tr>
<tr>
<td>circumflex (circu)</td>
<td>âëîöú</td>
</tr>
<tr>
<td>circle (circl)</td>
<td>ÂÙÛ</td>
</tr>
<tr>
<td>tie (tie)</td>
<td>ö</td>
</tr>
<tr>
<td>breve (bre)</td>
<td>åö</td>
</tr>
<tr>
<td>caron (car)</td>
<td>čšž</td>
</tr>
<tr>
<td>hungarian umlaut (hug)</td>
<td>õû</td>
</tr>
<tr>
<td>umlaut (uml)</td>
<td>äöü</td>
</tr>
<tr>
<td>dot (dot)</td>
<td>žš</td>
</tr>
</tbody>
</table>

Since in many international keyboards there are exceptions to what some dead keys should do, you can define them using

\texttt{\kxmod deadkey key outstring}

For example, on Slovak keyboard, if you enter caron-o, it generates circumflex-o, so you put in

\texttt{\kxmod caron o "\^o"}

to make it work correctly. Also, you have to define as exceptions dead keys over i and j, to remove the dot from them before inserting an accent mark. I will change this when the time comes, but so far I haven’t had time.

Oh, and about characters: backslash is escaped, so to enter it, you’ll need double backslash. Also, quotes and \# have different meaning. \# marks comments, quotes start and end \LaTeX-style commands. To enter quote, you’ll need to use \texttt{"}, to enter \#, use \texttt{\#}.

If you make a keyboard description file that works for your language, please mail it to me, so I can include it in the next keymap distribution.

More keywords will be supported in keymap configuration file in future, like

- \texttt{\include filename} include another file
- \texttt{\kprog program} define an external keymap translation program

Also, it should look into \texttt{lyxrc} file for defaults, too (for example, a \texttt{\include} option to include default keyboard).
4.4 International Keymap Stuff

The next two sections describe the .kmap and .cdef file syntax in detail. These sections should help you design your own key map if the ones provided do not meet your needs.

4.4.1 The .kmap File

A .kmap file maps keystrokes to characters or strings. As the name suggests it sets a keyboard mapping. The .kmap file keywords kmap, kmod, ksmod, and kcomb are described in this section.

**kmap** Map a character to a string

\kmap char string

This will map char to string. Note that in string, the double-quote (") and the backslash (\) must be escaped with a preceding backslash (\).

An example of a kmap statement to cause the symbol / to be output for the keystroke & is:

\kmap & /

**kmod** Specify an accent character

\kmod char accent allowed

This will make the character char be an accent on the allowed character(s). This is the dead key mechanism.

If you hit char and then another key not in allowed, you will get a char followed by the other, unallowed key, as output. Note that a Backspace cancels a dead key, so if you hit char Backspace, the cursor will not go one position backwards but will instead cancel the effect that char might have had on the next keystroke.

The following example specifies that the character ’ is to be an acute accent, allowed on the characters a, e, i, o, u, A, E, I, O, and U:

\kmod ’ acute aeiouAEIOU

**ksmod** Specify an exception to the accent character

\ksmod accent char result

---

5The term dead key refers to a key that does not produce a character by itself, but when followed with another key, produces the desired accent character. For example, a German character with an umlaut like ä can be produced in this manner.
This defines an exception for accent on char. The accent must have been assigned a keystroke with a previous \kmod declaration and char must not belong in the allowed set of accent. When you enter the accent char sequence, result is produced. If such a declaration does not exist in the .kmap file and you enter accent char, you get accent_key char where accent_key is the first argument of the \kmod declaration.

The following command produces causes äi to be produced when you enter acute-i (‘i):

```
\kxmod acute i "\\'{i}\n```

\kcomb Combine two accent characters

```
\kcomb accent1 accent2 allowed
```

This one is getting pretty esoteric. It allows you to combine the effect of accent1 and accent2 (in that order!) on allowed chars. The keystrokes for accent1 and accent2 must have been set with a \kmod command at a previous point in the file.

Consider this example from the greek.kmap file:

```
\kmod ; acute aeioyvhAEIOYVH \kmod : umlaut iyIY \kcomb acute umlaut iyIY
```

This allows you to press ;:i and get the effect of \'{i}. A backspace in this case cancels the last dead key, so if you press ;: Backspace i you get \'{i}.

### 4.4.2 The .cdef File

After the .kmap mapping is performed, a .cdef file maps the strings that the symbols generate to characters in the current font. The LyX distribution currently includes at least the iso8859-1.cdef and iso8859-2.cdef files.

In general the .cdef file is a sequence of declarations of the form

```
char_index_in_set string
```

For example, in order to map \'{e} to the corresponding character in the iso8859-1 set (233), the following declaration is used

```
233 "\\'{e}\n```

with \ and " being escaped in string. Note that the same character can apply to more than one string. In the iso-8859-7.cdef file you have

```
192 "\\'{\"{i}\n
192 "\\\"{\"{i}\n```

If LyX cannot find a mapping for the string produced by the keystroke or a deadkey sequence, it will check if it looks like an accented char and try to draw an accent over the character on screen.
4.4.3 Dead Keys

There is a second way to add support for international characters through so-called dead-keys. A dead-key works in combination with a letter to produce an accented character. Here, we’ll explain how to create a really simple dead-key to illustrate how they work.

Suppose you happen to need the circumflex character, “ˆ”. You could bind the ^-key [a.k.a. Shift-6] to the LyX command `accent-circumflex` in your lyxrc file. Now, whenever you type the ^-key followed by a letter, that letter will have a circumflex accent on it. For example, the sequence “ˆe” produces the letter: “ê”. If you tried to type “ˆt”, however, LyX will complain with a beep, since a “t” never takes a circumflex accent. Hitting Space after a dead-key produces the bare-accent. Please note this last point! If you bind a key to a dead-key, you’ll need to rebind the character on that key to yet another key. Binding the ,key to a cedilla is a bad idea, since you’ll only get cedillas instead of commas.

One common way to bind dead-keys is to use Meta-, Ctrl-, and Shift- in combination with an accent, like “˘” or “˚” or “ˆ”. Another way involves using `xmodmap` and `xkeycaps` [remember them from section?] to set up the special `Mode_Switch` key. The `Mode_Switch` acts in some ways just like Shift and permits you to bind keys to accented characters. You can also turn keys into dead-keys by binding them to something like `usldead_cedilla` and then binding this symbolic key to the corresponding LyX command. You can make just about anything into the `Mode_Switch` key: One of the Ctrl- keys, a spare function key, etc. As for the LyX commands that produce accents, check the entry for `accent-acute` in the `Reference Manual`. You’ll find the complete list there.

4.4.4 Saving your Language Configuration

You can edit your preferences so that your desired language environment is automatically configured when LyX starts up, via the Edit ⊃ Preferences dialog.

---

Note from John Weiss: This is exactly what I do in my ~/.lyx/lyxrc and my ~/.xmodmap files. I have my Scroll Lock key set up as `Mode_Shift` and a bunch of these “usldead_*” symbolic keys bound such things as Scroll Lock-˘ and Scroll Lock-˚. This is how I produce my accented characters.
Chapter 5

Installing New Document Classes, Layouts, and Templates

In this chapter, we describe the procedures for creating and installing new \LaTeX{} layout and template files, as well as offer a refresher on correctly installing new \LaTeX{} document classes. Some definitions: a document class is a \LaTeX{} file (usually ending in .cls or .sty) which describes the format of a document such as an article, report, journal preprint, etc. and all the commands needed to realize that format. A layout file is a \LaTeX{} file which corresponds to a \LaTeX{} document class and which tells \LaTeX{} how to “draw” things on the screen to make the display look something like the final printed page. More precisely, a layout file describes a “text class” which is the internal construct \LaTeX{} uses to render the screen display. “Layout” and “text class” can be used somewhat interchangeably, but it is better to refer to the file as the layout, and the thing living in \LaTeX{}’s memory as the text class. A template file is simply a \LaTeX{} document which contains a set of predefined entries for a given document class which are generally required for that class. Templates are especially useful for things like journal manuscripts which are to be submitted electronically.

5.1 Installing a new \LaTeX{} package

Some installations may not include a \LaTeX{} package that you would like to use within \LaTeX{}. For example, you might need Foil\LaTeX{}, a common (and very powerful) package for preparing slides or viewgraphs for overhead projectors. Here are the formal steps involved in getting the package up and running if you are using \LaTeX{} or some other web2c based distribution.

1. Get the package from CTAN or wherever.\footnote{See the Inventory of your \LaTeX{} configuration manual for details of what CTAN is and}
2. Read the file `texmf.cnf` (this usually lives in the directory `$TEXMF/web2c`, though you can run `kpsewhich texmf.cnf` to locate it). It describes how to add a local `texmf` directory; follow the instructions. You need to insert the name of your local `texmf` directory in `texmf.cnf`. Under Linux, `/usr/local` is a logical place to install software that did not come with your distribution, so you might use `/usr/local/texmf`. Usually, you will have to modify only two things:

(a) Set `TEXMFLOCAL` to the directory you chose; e.g.

```
TEXMFLOCAL = /usr/local/texmf
```

(b) Make sure `TEXMF` includes the `TEXMFLOCAL` variable; e.g.

```
TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN}
```

3. Create your local `texmf` directory (e.g. `/usr/local/texmf`). You must follow the directory structure of your existing `texmf` directory (for example, latex packages should go under `/usr/local/texmf/tex/latex/`).

4. Install the package. For example, you would unpack the FoilT\TeX{} tarball and create `/usr/local/texmf/tex/latex/foiltex`. The `foiltex` directory contains various files.

5. Run: `texhash`. This should create `/usr/local/texmf/ls-R` amongst others.

6. From within \LaTeX{}, do: `Tools> Reconfigure`. Restart \LaTeX{}.

Now you should see your new package—for example slides (FoilT\TeX{})—under \LaTeX{}\texttt{Document}, field \texttt{Class}. Note that there are simpler ways of installing packages: you can add a link to the new package directory in the system \LaTeX{} directory (`$TEXMF/tex/latex`, don’t forget to then run `texconfig`), or sometimes simply set the `TEXINPUTS` environment variable to include the new package. However, the formal procedure described in `texmf.cnf` is guaranteed to work, so you should follow it unless circumstances absolutely prevent it: such as, when you don’t have superuser access.

5.2 Layouts

This section describes how to write and install your own \LaTeX{} layout files (also known as text classes) and walks through the \texttt{article} text class format as an example. The `.layout` files describe what paragraph styles are available for a given document class and how \LaTeX{} should display them. We try to provide a thorough description of the process here; however, there are so many different types of documents supported by \LaTeX{} classes we can’t hope to cover every different possibility or problem you might encounter.

where supported document classes can be found.
When you plan to write a new layout, it is extremely helpful to look at the example layouts distributed with LyX. If you use a nice \LaTeX document class that might be of interest for others, too, and have a nice corresponding LyX layout, feel free to contribute the stuff to us, so we may put it into the distribution.

All the tags described in this chapter are case-insensitive; this means that Style, style and StYlE are really the same command. The possible values are printed in brackets after the feature’s name. The default value if a feature isn’t specified inside a text class-description is typeset emphasized. If the argument has a datatype like “string” or “float”, the default is shown like this: float=default.

5.2.1 Supporting new document classes

There are two situations you are likely to encounter when wanting to support a new \LaTeX document class, involving \LaTeX 2ε class (.cls) and style (.sty) files.

5.2.2 A layout for a sty file

If your new document class is provided as a style file that is used in conjunction with an existing, supported document class, start by copying the existing class’s layout file into your local directory. For the sake of example we’ll assume that the style file is called myclass.sty and it is meant to be used with report.cls which is a standard class.

    cp report.layout ~/.lyx/layouts/myclass.layout

Then edit myclass.layout and change the line:

    \DeclareLaTeXClass{report}

to read

    \DeclareLaTeXClass{report, myclass.sty}{report (myclass)}

Then add:

    \usepackage{myclass}

near the top of the file.

Start LyX and select Tools->Reconfigure. Restart LyX and try creating a new document. You should see "report (myclass)" as a document class option in the Document->Settings dialog. It is likely that some of the sectioning commands
and such will differ from how the base class\(^2\) works, so you can fiddle around with the settings for the different sections if you wish. See below for more discussion on this.

### 5.2.3 Layout for a cls file

In this case, you will probably have to “roll your own” layout. We strongly suggest copying an existing layout file which uses a similar \LaTeX\ class and modifying it if at all possible. At least use an existing file as a starting point so you can find out what items you need to worry about. Again, the specifics are covered below.

### 5.3 Declaring a new text class

When it’s finally time to get your hands dirty and create or edit your own layout file, the following sections describe what you’re up against. Our advice is to go slowly, save and test often, listen to soothing music, and enjoy one or two of your favorite adult beverages; more if you are getting particularly stuck. It’s really not that hard, except that the multitude of options can become overwhelming if you try to do to much in one sitting. Go have another adult beverage, just for good measure.

Here we go!

Lines in a layout file which begin with a \# are comments. There is one exception to this rule: all layouts should begin with lines like:

```latex
#% Do not delete the line below; configure depends on this
# \DeclareLaTeXClass{article}
```

The second line is used when you configure L\LaTeX\X. The layout file is read by the \LaTeX\X script `chkconfig.ltx`, in a special mode where \# is ignored. The first line is just a \LaTeX\ comment, and the second one contains the declaration of the text class. If these lines appear in a file named `article.layout`, then they define a text class of name `article` (the name of the layout file) which uses the \LaTeX\ document class `article.cls` (the default is to use the same name as the layout). The string “article” that appears above is used as a description of the text class in the Document Settings dialog.

Let’s assume that you wrote your own text class that uses the `article.cls` documentclass, but where you changed the appearance of the section headings. If you put it in a file `myarticle.layout`, the header of this file should be:

```latex
#% Do not delete the line below; configure depends on this
# \DeclareLaTeXClass[article]{article (with my own headings)}
```

This declares a text class `myarticle`, associated with the \LaTeX\X document class `article.cls` and described as “article (with my own headings)\(^2\)”. If your text class depends on several packages, you can declare it as:

\(^2\)report in this example
5.3. DECLARING A NEW TEXT CLASS

## Do not delete the line below; configure depends on this
\DeclareLaTeXClass[article,foo.sty]{article (with my own headings)}

This indicates that your text class uses the foo.sty package. Finally, it is also possible to declare classes for SGML and DocBook code. Typical declarations will look like

## Do not delete the line below; configure depends on this
\DeclareSGMLClass{SGML (LinuxDoc)}
or

## Do not delete the line below; configure depends on this
\DeclareDocBookClass[article]{SGML (DocBook article)}

Note that these declarations can also be given an optional parameter declaring the name of the document class (but not a list).

When the text class has been modified to your taste, all you have to do is to copy it either in $\texttt{LyXDir/layouts/}$ or in $\texttt{UserDir/layouts}$ and run $\texttt{Tools\rightarrow Reconfigure}$. Exit $\texttt{LyX}$ and restart it; then your new text class should be available along with the others.

### 5.3.1 File format

The first non-comment line must contain the file format number:

**Format** [int] This tag was introduced with LyX 1.4.0 (layout files of LyX 1.3.x and earlier don’t have an explicit file format). The file format that is documented here is 2.

### 5.3.2 General text class parameters

These are the general parameters which describe the form of the entire document:

**Columns** [1, 2] Whether the class-default should have one or two columns. Can be changed in the Document > Settings dialog. This setting (same goes for Sides, too) is important: if your text class has two columns by default but you forget to set it correctly, the twocolumn \LaTeX option will not be output when you select Two columns in Document > Settings.

**Sides** [1, 2] Whether the class-default should be printing on one or both sides of the paper. Can be changed in the Document > Settings dialog.

**PageStyle** [plain, empty, headings] The class default pagestyle. Can be changed in the Document > Settings dialog.
CHAPTER 5. INSTALLING NEW DOCUMENT CLASSES

ClassOptions...End This section describes various global options supported by the document class. See Section 5.3.3 for a description.

ProvidesAmsmath [0, 1] Whether the class already loads the amsmath package. This is the case of the amsart and amsbook document classes.

ProvidesMakeidx [0, 1] Whether the class already provides the functionality of the makeidx package. This is the case of the amsart and amsbook document classes.

ProvidesUrl [0, 1] Whether the class already provides the functionality of the url package. This is the case of the AASTeX document class.

DefaultFont This is used to describe the default font of the document. See Section 5.3.8 for a description.

DefaultStyle [string] This is the style that will be assigned to new paragraphs, usually Standard. This will default to the first defined style if not given, but you are highly encouraged to use this directive.

TitleLatexType [CommandAfter, Environment] Indicates what kind of markup is used to define the title of a document. CommandAfter means that the macro with name TitleLaTeXCommand will be inserted after the last layout which has “InTitle 1”. Environment corresponds to the case where the block of paragraphs which have “InTitle 1” should be enclosed into the TitleLaTeXCommand environment.

TitleLaTeXCommand [string="maketitle"] The name of the command/environment mentioned above.

Preamble...EndPreamble A set of macro definitions that will be output at the beginning of the \LaTeX files. Use this for global definitions.

Input As its name implies, this command allows you to include another layout definition file within yours to avoid duplicating commands. Common examples are the standard layout files, for example, stdclass.inc, which contains most of the basic layouts.

Style...End This sequence defines a new style. If the style already exists, it will redefine some of its parameters instead. See Section 5.3.4 for details.

NoStyle This command deletes an existing style. This is particularly useful when you want to suppress a style that has been defined in an input file.

Float...End This sequence defines a new float. See Section 5.3.5 for details.

NoFloat This command deletes an existing float. This is particularly useful when you want to suppress a float that has been defined in an input file.

CharStyle...End This section defines a new character style. See Section 5.3.6 for a description.
5.3. DECLARING A NEW TEXT CLASS

Counter...End This sequence defines a new counter. See Section 5.3.7 for details.

5.3.3 ClassOptions section

The ClassOptions section can contain the following entries:

FontSize [string="10|11|12"] The list of available font sizes for the document's main font, separated by “|”.

PageStyle [strings="empty|plain|headings|fancy"] The list of available page styles, separated by “|”.

Other [string=""] Some document class options, separated by a comma, that will be added to the optional part of the \documentclass command.

5.3.4 Specific Paragraph Layouts

A paragraph layout description looks like this\textsuperscript{3}:

Style name

... 

End

where the following commands are allowed:

CopyStyle [string] This is used to copy all the features of an existing layout into the current one.

LatexType [Paragraph, Command, Environment, Item_Environment, List_Environment] How the layout should be translated into \LaTeX. Paragraph means nothing special. Command means \texttt{\LaTeXName\{...\}} and Environment means \texttt{\begin{\LaTeXName}...\end{\LaTeXName}}. Item_Environment is the same as Environment, except that a \texttt{\item} is generated for each paragraph of this environment. List_Environment is the same as Item_Environment, except that \texttt{LabelWidthString} is passed as an argument to the environment. \texttt{LabelWidthString} can be defined in the \texttt{Layout\ Paragraph} dialog. \texttt{LatexType} is perhaps a bit misleading, since these rules apply to SGML classes, too. Visit the SGML class files for specific examples.

InTitle [1, 0] If 1, marks the layout as being part of a title block (see also the \texttt{TitleLatexType} and \texttt{TitleLatexCommand} global entries)

LatexName The name of the corresponding \LaTeX stuff. Either the environment or command name.

\textsuperscript{3}Note that this will either define a new layout or modify an existing one.
**CHAPTER 5. INSTALLING NEW DOCUMENT CLASSES**

**LatexParam** The optional parameter for the corresponding LatexName stuff. This parameter cannot be changed from within \LaTeX{}.

**OptionalArgs** [int=0] The number of optional arguments that can be used with this layout. This is useful for things like section headings, and only makes sense with \LaTeX{}.

**Margin** [Static, Manual, Dynamic, First_Dynamic, Right_Address_Box] The kind of margin that the layout has on the left side. Static just means a fixed margin. Manual means that the left margin depends on the string entered in the \texttt{Edit} \texttt{Paragraph Settings} dialog. This is used to typeset nice lists without tabulators. Dynamic means that the margin depends on the size of the label. This is used for automatic enumerated headlines. It is obvious that the headline “5.4.3.2.1 Very long headline” must have a wider left margin (as wide as “5.4.3.2.1” plus the space) than “3.2 Very long headline”, even if other word processors are not able to do this. First_Dynamic is similar, but only the very first row of the paragraph is dynamic, while the others are static; this is used, for example, for descriptions. Right_Address_Box means the margin is chosen in a way that the longest row of this paragraph fits to the right margin. This is used to typeset an address on the right edge of the page.

**NextNoIndent** [1, 0] Whether the following Paragraph is allowed to indent its very first row. 1 means that it is not allowed to do so, 0 means it could do so if it wants to.

**ParIndent** [string=“”] The indent of the very first line of a paragraph. The argument is passed as a string. For example “MM” means that the paragraph is indented with the width of “MM” in the normal font. You can get a negative width by prefixing the string with “-”. This way was chosen so that the look is the same with each used screen font. The Parindent will be fixed for a certain layout. The exception is Standard layout, since the indentation of a Standard layout paragraph can be prohibited with NextNoIndent. Also, Standard layout paragraphs inside environments use the Parindent of the environment, not their native one. For example, Standard paragraphs inside an enumeration are not indented.

**Parskip** [float=0] \LaTeX{} allows to choose either “indent” or “skip” to typeset a document. When “indent” is chosen, this value is completely ignored. When “skip” is chosen, the parindent of a \LaTeX{} type “Paragraph” layout is ignored and all paragraphs are additionally separated by this parskip argument. The vertical space is calculated with \texttt{value*DefaultHeight()} where \texttt{DefaultHeight()} is the height of a row with the normal font. This way, the look stays the same with different screen fonts.

**TopSep** [float=0] The vertical space with which the very first of a chain of paragraphs with this layout is separated from the previous paragraph. If
5.3. DECLARING A NEW TEXT CLASS

the previous paragraph has another layout, the separations are not simply added, but the maximum is taken.

**BottomSep** [float=0] The same as **TopSep** for the very last paragraph.

**Parsep** [float=0] The vertical space between two paragraphs of this layout.

**Itemsep** [float=0] This is an extra space between the paragraphs of an environment layout. If you put other layouts into an environment, each is separated with the environment’s **Parsep**. But the whole items of the environment are additionally separated with this **Itemsep**.

**LeftMargin** [string=""] If you put layouts into environments, the leftmargins are not simply added, but added with a factor \( \frac{4}{depth+4} \). Note that this parameter is also used when the border is defined as **Manual** or **Dynamic**. Then it is added to the manual or dynamic border. This string has the same meaning as for **ParIndent**.

**RightMargin** [string=""] Similar to **LeftMargin**.

**Labeltype** [No_Label, Manual, Static, Top_Environment, Centered_Top_Environment, Sensitive, Counter]

Manual means the label is the very first word (up to the first real blank). Static means it is defined in the layout (see **LabelString**). **Top_Environment** and **Centered_Top_Environment** are special cases of Static. The label will be printed above the paragraph, but only at the top of an environment or the top of a chain of paragraphs with this layout.

Usage is for example the **Abstract** layout or the **Bibliography** layout. This is also the case for **Manual** labels with latex type **Environment**, in order to make layouts for theorems work correctly. **Sensitive** is a special case for the caption-labels “Figure” and “Table”. **Sensitive** means the (hard-coded) label string depends on the kind of float. The **Counter** label type defines automatically numbered labels.

**LabelCounter** [Chapter, Section, Subsection, Subsubsection, Paragraph, Subparagraph, EnumI, EnumII, EnumIII, EnumIV]

The name of the counter for automatic numbering. This must be given if **Labeltype** is **Counter**.

**Labelsep** [string=""] The horizontal space between the label and the text body. Only used for labels that are not above the text body.

**LabelBottomsep** [float=0] The vertical space between the label and the text body. Only used for labels that are above the text body (**Top_Environment**, **Centered_Top_Environment**).

**LabelString** [string=""] The string used for a label with a **Static** labeltype. When the border is **Manual** this string is also used as a suggestion for the **LabelWidthString** that can be set in the **Edit Paragraph Settings** dialog.
When \texttt{LabelCounter} is set, this string can be contain special formatting commands as explained in Section 5.3.7.

\texttt{LabelStringAppendix [string=""]} If non-empty, this is used inside the appendix instead of \texttt{LabelString}.

\texttt{TocLevel [int]} The level of the style in the table of contents. This is used for automatic numbering of section headings.

\texttt{EndLabeltype [\texttt{No_Label, Box, Filled_Box, Static}]} The type of label that stands at the end of the paragraph (or sequence of paragraphs if \texttt{LatexType} is \texttt{Environment}, \texttt{Item_Environment} or \texttt{List_Environment}). \texttt{No_Label} means “nothing”; \texttt{Box} (resp. \texttt{Filled_Box}) is a white (resp. black) square suitable for end of proof markers, \texttt{Static} is an explicit text string.

\texttt{EndLabelString [string=""]} The string used for a label with a \texttt{Static} \texttt{EndLabelType}.

\texttt{Align [\texttt{block, left, right, center}]} Paragraph alignment.

\texttt{AlignPossible [\texttt{block, left, right, center}]} A comma separated list of possible aligns. Some \LaTeX{} styles prohibit certain alignments, since those wouldn’t make sense. For example a right-aligned or centered enumeration isn’t possible.

\texttt{Fill_Top [0,1]} With this parameter the Fill value of the “Vertical space above” list of the \texttt{Edit>Paragraph Settings} dialog can be set when initializing a paragraph with this layout\footnote{Note from Jean-Marc: I’m not sure that this setting has much use, and it should probably be removed in later versions}.

\texttt{Fill_Bottom [0,1]} Similar to \texttt{Fill_Top}.

\texttt{NeedProtect [0,1]} Whether fragile commands in this layout should be \texttt{\protect}’ed.

\texttt{Newline [0, 1]} Whether newlines are translated into \LaTeX{} newlines (\textbackslash\textbackslash) or not. The translation can be switched off to allow more comfortable \LaTeX{} editing inside \lyx{}.

\texttt{PassThru [0, 1]} Whether the contents of this paragraph should be output in raw form, meaning without special translations that \LaTeX{} would require. This somehow replaces the older \texttt{Latex} font property.

\texttt{FreeSpacing [0, 1]} Usually \lyx{} doesn’t allow you to insert more than one space between words, since a space is considered as the separation between two words, not a character or symbol of its own. This is a very fine thing but sometimes annoying, for example when typing program code or plain \LaTeX{} code. For this reason, \texttt{FreeSpacing} can be enabled. Note that \lyx{}
5.3. DECLARING A NEW TEXT CLASS

will create protected blanks for the additional blanks when in another mode than \LaTeX-mode.

KeepEmpty [0, 1] Usually \LaTeX{} does not allow you to leave a paragraph empty, since it would lead to empty \LaTeX{} output. There are some cases where this could be desirable however: in a letter template, the required fields can be provided as empty fields, so that people do not forget them; in some special classes, a layout can be used as some kind of break, which does not contain actual text.

Spacing [single, onehalf, double, other value] This defines what the default spacing should be in the layout. The arguments single, onehalf and double correspond respectively to a multiplier value of 1, 1.25 and 1.667. If you specify the argument other, then you should also provide a numerical argument which will be the actual multiplier value. Note that, contrary to other parameters, Spacing implies the generation of specific \LaTeX{} code, using the package \texttt{setspace.sty}.

Font The font used for both the text body \textit{and} the label. See section 5.3.8. Note that defining this font automatically defines the LabelFont to the same value.

TextFont The font used for the text body. See section 5.3.8.

LabelFont The font used for the label. See section 5.3.8.

Preamble...EndPreamble A set of macro definitions that will be output at the beginning of the \LaTeX{} files when the layout is used. Use this to define the macros needed by this particular layout.

DependsOn the name of a style which preamble should be output before the one mentioned above. This allows to ensure some ordering of the preamble snippets when macros definitions depend on one another.

5.3.5 Floats

Since version 1.3.0 of \LaTeX{}, it is necessary to define the floats (figure, table, ...) in the text class itself. If you are looking here to learn how to upgrade an existing text class, it will probably turn out that all you have to do is to add

\begin{verbatim}
\input stdfloats.inc
\end{verbatim}

at a reasonable location of the text class. If you want to implement a text class that proposes some other float types (like the AGU class bundled with \LaTeX{}), the information below will hopefully help you:

\textsuperscript{5}Note that, besides that functionality, there is no way to ensure any ordering of preambles. The ordering that you see in a given version of \LaTeX{} may change without warning in later versions.

\textsuperscript{6}Don’t forget to also have a look at counters in next section.
CHAPTER 5. INSTALLING NEW DOCUMENT CLASSES

Type [string=””] The “type” of the new class of floats, like program or algorithm. After the appropriate \newfloat, commands such as \begin{program} or \end{algorithm*} will be available.

GuiName [string=””] The string that will be used in the menus and also for the caption.

LaTeXBuiltin [0 , 1] Set to 1 if the float is already defined by the document class. If this is set to 0, the float will be defined using the \LaTeX package float.

NumberWithin [string=””] This (optional) argument determines whether floats of this class will be numbered within some sectional unit of the document. For example, if within is equal to chapter, the floats will be numbered within chapters.

Style [string=””] The style used when defining the float using \newfloat.

Placement [string=””] The default placement for the given class of floats. They are like in standard \LaTeX: t, b, p and h for top, bottom, page, and here, respectively. On top of that there is a new type, H, which does not really correspond to a float, since it means: put it “here” and nowhere else. Note, however that the H specifier is special and, because of implementation details cannot be used in non-builtin float types. If you do not understand what this means, just use "tbp".

Extension [string=””] The file name extension of an auxiliary file for the list of figures (or whatever). \LaTeX writes the captions to this file.

ListName [string=””] The heading used for the list of floats.

5.3.6 Character styles

You can define character styles since version 1.4.0 of \LaTeXX. The CharStyle section can contain the following entries:

Font The font used for both the text body and the label. See section 5.3.8. Note that defining this font automatically defines the LabelFont to the same value.

LabelFont The font used for the label. See section 5.3.8.

LatexName The name of the corresponding \LaTeX stuff. Either the environment or command name.

LatexParam The optional parameter for the corresponding LatexName stuff. This parameter cannot be changed from within \LaTeXX.

LatexType See section 5.3.4.

Preamble...EndPreamble See section 5.3.4

Note that the order of these letters in the string is irrelevant, like in \LaTeX.
5.3. DECLARING A NEW TEXT CLASS

5.3.7 Counters

Since version 1.3.0 of LyX, it is necessary to define the counters (chapter, figure, ...) in the text class itself. If you are looking here to learn how to upgrade an existing text class, it will probably turn out that all you have to do is to add

Input stdcounters.inc

The definition of counters is presently a bit primitive in LyX, since many things are still hardcoded. The following two parameters can be used:

Name [string="] The name of the counter

Within [string="] If this is set to the name of another counter, the present counter will be reset everytime the other one is increased (is that unclear enough?).

When a counter has been associated to a style, it is possible to use some special constructs in LabelString and LabelStringAppendix:

- @style-name@ will be replaced the expanded LabelString of style style-name. This is used for example to define the label of a subsection in terms of the label of a section.
- counter values can be expressed using LyX-like macros \numbertype{counter}, where numbertype can be:
  - arabic to translate counter to arabic numerals, like 1, 2, 3...\footnote{Actually, the situation is a bit more complicated than that: any numbertype other than those described below will produce arabic numerals. It would not be surprising to see this change in the future.}
  - alph for lower-case letters: a, b, c, ...
  - Alph for upper-case letters: A, B, C, ...
  - roman for lower-case roman numerals: i, ii, iii, ...
  - Roman for upper-case roman numerals: I, II, III...
  - hebrew for hebrew numerals.

5.3.8 Font description

A font description looks like that:

Font or LabelFont
...
EndFont

and the following commands are available:
CHAPTER 5. INSTALLING NEW DOCUMENT CLASSES

Family [Roman, Sans, Typewriter]
Series [Medium, Bold]
Shape [Up, Italic, SmallCaps, Slanted]
Size [tiny, small, normal, large, larger, largest, huge, giant]
Color [none, black, white, red, green, blue, cyan, magenta, yellow]

5.3.9 Upgrading old layout files

The file format of layout files changes from time to time, so old layout files need to be converted. This process has been automated in LyX 1.4.0: If LyX reads an old format layout file it will call the conversion tool $LyXDir/scripts/layout2layout.py and convert it to a temporary file in current format. The original file is left untouched, so that you can still use it with LyX 1.3.x. If you want to convert the layout file permanently, just call the converter by hand:

```
python $LyXDir/scripts/layout2layout.py myclass.layout myclassnew.layout
```

Then copy myclassnew.layout to $UserDir/layouts/.

The automatic conversion does only handle syntax changes. It cannot handle the case where the contents of included files was changed. For example, layout files based on book.layout need to include numreport.inc in addition to stdclass.inc. If you get error messages about undefined counters, try to convert your file with layout2layout.py and then add one of numarticle.inc, numreport.inc and numrevtex.inc.

5.4 Creating Templates

Templates are created just like usual documents. The only difference is that usual documents contain all possible settings, including the fontscheme and the papersize. Usually a user doesn’t want a template to overwrite his defaults in these cases. For that reason, the designer of a template should remove the corresponding commands like \fontscheme or \papersize from the template LyX file. This can be done with any simple text-editor, for example vi or xedit.

Put the edited template files you create in $UserDir/templates/, copy the ones you use from the global template directory in $LyXDir/templates/ to the same place, and redefine the template path in the Edit > Preferences dialog (tabs Input, Path).

Note that there is a template which has a particular meaning: defaults.lyx. This template is loaded everytime you create a new document with File > New in order to provide useful defaults. To create this template from inside LyX, all you have to do is to open a document with the correct settings, and use the Save as Document Defaults button.
Chapter 6

Including External Material

6.1 Background

One often requested feature from LyX users is to be able to interface LyX with XFig, Dia, or other similar applications that specialize in producing a certain kind of diagram, figure, schematic or whatever material might be relevant to include in your document. Previously, it was only possible to include boring, static, fixed images in LyX documents with the graphics feature, but there are several limitations attached to this approach:

- If you want to change the figure, you have to invoke an external program by hand
- LyX does not notice that the referenced files change, so the on-screen display can fast become obsolete, and this is aggravated by the lack of a means of updating the display
- The graphics stuff does not provide any mechanisms for coping with different exported formats such as DocBook, HTML or raw Ascii

The external material facility attempts to solve all of these problems\(^1\). It does this by offering a general method to interface LyX to external applications. Instead of introducing a long list of different constructs tailored for each specific application, we chose to sacrifice the in-line displaying of the included material in order to provide a general construct to cover a wide range of applications. The result is the external material construct. External material presents itself in the document simply as a button, but don’t let this fool you. When you click on it, a dialog will appear that allows you to chose exactly what material to include, and in the following sections you will learn that this is indeed a powerful mechanism that can solve all of the above problems, and more.

\(^1\)Even if the graphics facility can’t solve all problems, it is still valuable because it does provide in-line preview of the graphics, and supports advanced geometric transformations with a comfortable user interface.
6.2 How does it work?

The external material feature is based on the concept of a template. A template is a specification of how LYX should interface with a certain kind of material. As bundled, LYX comes with predefined templates for XFig figures, Dia diagrams, various raster format images, gnuplot, and more. You can check the actual list by using the Insert > External Material command. Furthermore, it is possible to roll your own template to support a specific kind of material. Later we’ll describe in more detail what is involved, and hopefully you will submit all the templates you create so we can include them in a later LYX version.

Another basic idea of the external material feature is to distinguish between the original file that serves as a base for final material and the produced file that is included in your exported or printed document. For example, consider the case of a figure produced with XFig. The XFig application itself works on an original file with the .fig extension. Within XFig, you create and change your figure, and when you are done, you save the fig-file. When you want to include the figure in your document, you invoke transfig in order to create a PostScript file that can readily be included in your LATEX file. In this case, the .fig file is the original file, and the PostScript file is the produced file.

This distinction is important in order to allow updating of the material while you are in the process of writing the document. Furthermore, it provides us with the flexibility that is needed to support multiple export formats. For instance, in the case of an Ascii resulting file, it is not exactly an award-winning idea to include the figure as raw PostScript. Instead, you'd either prefer to just include a reference to the figure, or try to invoke some graphics to Ascii converter to make the final result look similar to the real graphics. The external material management allows you to do this, because it is parameterized on the different export formats that LYX supports.

Besides supporting the production of different products according to the exported format, it supports tight integration with editing and viewing applications. In the case of an XFig figure, you are able to invoke xfig on the original file with a single click from within the external material dialog in LYX, and also preview the produced PostScript file with ghostview with another click. No more fiddling around with the command line and/or file browsers to locate and manipulate the original or produced files. In this way, you are finally able to take full advantage of the many different applications that are relevant to use when you write your documents, and ultimately be more productive.

So, all in all, LYX has information about a number of different programs to use behind the scenes in order to realize all of this machinery. This information, in fact, is exactly what is contained in the templates. To each template, there is a list of command lines that are used to invoke applications, convert the original file to the produced file, and more. This mechanism allows the advanced user to extend the capabilities of LYX without fiddling with the source code. It requires some footwork to define all the different commands and flags, but luckily, the LYX team did all the hard work and specified these for you.

But before the trees grow into the skies, we have to admit that we did take
one tiny short-cut. Since you can produce many different kinds of files to go with each exported format, one could also expect that it would be possible to preview each product. The \LaTeX{} team decided against this in order to keep the user interface simple. Instead of providing a button for each exported file format, we decided to introduce the concept of the primary file format and just have one button. When you press View result in the external material dialog, you will get a view of the produced file in the primary file format. And the primary file format is specified by your document class. For most document classes, the primary file format is \LaTeX{}, but for the DocBook document classes, the primary file format is DocBook. So, when you view the produced file, keep in mind that it will only be a preview of what the main result will be. If you want to see how other exported formats turn out, you have to export them and preview them by hand.

6.3 The external material dialog

You insert external material from the Insert menu. When you do this, a button is inserted into your document, and the external material dialog is shown. This dialog allows you to describe exactly what material should be included, and also how it should be included. Furthermore, it provides access to the external applications to either view, edit or produce the material that is used in the resulting file.

At the top of this dialog, there is a drop-down list where you can chose which template should be used. Just below the template drop-down, there’s a text area with a short blurb about the chosen template that should help you use it. Most often, it will provide a short description of the template, and a few hints on how to parameterize the use of it. Further down, you’ll find a filename input field along with a “Browse” button that allows you to chose which file should be included with the standard file browser. Thus this field specifies the original file. Since the produced file is automatically generated when needed, there is no need to give access to it in the user interface.

At the bottom of the dialog, you’ll find a general input box called Parameters. This box is generally used to parameterize the specific template. The specific use should be covered in the help blurb associated with the template, but it typically allows you to define variations on how the produced file should be generated.

At the right side of the dialog, you’ll find three buttons: Edit file, View result, and Update result. These in turn allow you to edit your original file with the appropriate editing application, view the produced file as included in the primary format document, and finally force an update of the resulting material in the primary format. Normally, the Update result button will be disabled, because most templates are configured to automatically update the produced file when needed. In those cases, there is no need to force the production of a new produced file. However, some templates are configured to not be automatically producing the residual product, because the cost of producing the produced file
might be so large that it would be a pain to do it all the time. Those types are known as manual external material. In those cases, you can use the button to force the production of the produced file exactly when you need it, and thus control the amount of work that is done. In fact, it is your responsibility to do this to keep the produced files current at all times: before printing, before exporting, before viewing, etc. At some time in the future, it might be possible that \LaTeX will help you with this task. Any changes in the template, filename or parameters are actually applied whenever you press Edit file, View result or Update result buttons.

6.4 Examples

In this section, we should include some examples of use of the external material. Those examples could include:

- External raster images
- External XFig figures
- Chess diagrams
- Sound samples
- The use of makefiles
- Recursive external \LaTeX templates

6.5 The external template configuration file

It is relatively easy to add custom external template definitions to \LaTeX. However, be aware this doing this in an careless manner most probably will introduce an easily exploitable security hole. So before you do this, please read the discussion about security which will follow later.

Having said that, we encourage you to submit any interesting templates that you create.

The external templates are defined in the lib/external_templates file. You can place your own version in .lyx/external_templates.

A typical template looks like this:

```latex
Template XFig
GuiName "XFig: $$AbsOrRelPathParent$$Basename"
HelpText
An XFig figure.
HelpTextEnd
InputFormat fig
FileFilter "*.fig"
AutomaticProduction true
```
As you can see, the template is enclosed in Template ... TemplateEnd. It con-
CHAPTER 6. INCLUDING EXTERNAL MATERIAL

tains a header specifying some general settings, and for each supported primary
document file format a section Format ... FormatEnd.

6.5.1 The template header

Template <id> A unique name for the template. It must not contain substitu-
tion macros (see below).

GuiName <guiname> The text that is displayed on the button. This command
must occur exactly once.

HelpText <text> HelpTextEnd The help text that is used in the External di-
alog. Provide enough information to explain to the user just what the
template can provide him with. This command must occur exactly once.

InputFormat <format> The file format of the original file. This must be the
name of a format that is known to LyX (see the Tools>Preferences:Conversion
dialog). Use "*" if the template can handle original files of more than one
format. LyX will attempt to interrogate the file itself in order to deduce
its format in this case. This command must occur exactly once.

FileFilter <pattern> A glob pattern that is used in the file dialog to filter
out the desired files. If there is more than one possible file extension
(e.g. tgif has .obj and .tgo), use something like "*.{obj,tgo}". This
command must occur exactly once.

AutomaticProduction true|false Wether the file represented by the tem-
plate must be generated by LyX. This command must occur exactly once.

Transform Rotate|Resize|Clip|Extra This command specifies which trans-
formations are supported by this template. It may occur zero or more
times. This command enables the corresponding tabs in the external
dialog. Each Transform command must have either a corresponding
TransformCommand or a TransformOption command in the Format sec-
tion. Otherwise the transformation will not be supported by that format.

6.5.2 The Format section

Format LaTeX|PDFLaTeX|Ascii|DocBook|LinuxDoc The primary document file
format that this format definition is for. Not every template has a sensible
representation in all document file formats. Please define nevertheless a
Format section for all formats. Use a dummy text when no representa-
tion is available (see the LinuxDoc format in the example above). Then
you can at least see a reference to the external material in the exported
document.

TransformCommand Rotate RotationLatexCommand This command specifies that
the built in \LaTeX command should be used for rotation. This command
may occur once or not at all.
6.5. THE EXTERNAL TEMPLATE CONFIGURATION FILE

TransformCommand Resize ResizeLatexCommand This command specifies that the built in \LaTeX command should be used for resizing. This command may occur once or not at all.

TransformOption Rotate RotationLatexOption This command specifies that rotation is done via an optional argument. This command may occur once or not at all.

TransformOption Resize ResizeLatexOption This command specifies that resizing is done via an optional argument. This command may occur once or not at all.

TransformOption Clip ClipLatexOption This command specifies that clipping is done via an optional argument. This command may occur once or not at all.

TransformOption Extra ExtraLatexOption This command specifies that an extra optional argument is used. This command may occur once or not at all.

Product <text> The text that is inserted in the exported document. This is actually the most important command and can be quite complex. This command must occur exactly once.

UpdateFormat <format> The file format of the converted file. This must be the name of a format that is known to \LaTeX{} (see the Tools\ Preferences: Conversion dialog). This command must occur exactly once.

UpdateResult <filename> The file name of the converted file. The file name must be absolute. This command must occur exactly once.

ReferencedFile <format> <filename> This command denotes files that are created by the conversion process and are needed for a particular export format. If the filename is relative, it is interpreted relative to the master document. This command may be given zero or more times.

Requirement <package> The name of a required \LaTeX{} package. The package is included via \usepackage{} in the \LaTeX{} preamble. This command may occur zero or more times.

Preamble <name> This command specifies a preamble snippet that will be included in the \LaTeX{} preamble. It has to be defined using PreambleDef ... PreambleDefEnd. This command may occur zero or more times.

Option <name> <value> This command defines an additional macro $$<name> for substitution in Product. <value> itself may contain substitution macros. The advantage over using <value> directly in Product is that the substituted value of $$<name> is sanitized so that it is a valid optional argument in the document format. This command may occur zero or more times.
6.5.3 Preamble definitions

The external template configuration file may contain additional preamble definitions enclosed by \texttt{PreambleDef ... PreambleDefEnd}. They can be used by the templates in the Format section.

6.6 The substitution mechanism

When the external material facility invokes an external program, it is done on the basis of a command defined in the template configuration file. These commands can contain various macros that are expanded before execution. Execution always take place in the directory of the containing document.

Also, whenever external material is to be displayed, the name will be produced by the substitution mechanism, and most other commands in the template definition support substitution as well.

The available macros are the following:

\texttt{$$FName$$} The filename of the file specified in the external material dialog. This is either an absolute name, or it is relative to the \LaTeX{} document.

\texttt{$$Basename$$} The filename without path and without the extension.

\texttt{$$Extension$$} The file extension (including the dot).

\texttt{$$FPath$$} The path part of \texttt{$$FName$$} (absolute name or relative to the \LaTeX{} document).

\texttt{$$AbsPath$$} The absolute file path.

\texttt{$$RelPathMaster$$} The file path, relative to the master \LaTeX{} document.

\texttt{$$RelPathParent$$} The file path, relative to the \LaTeX{} document.

\texttt{$$AbsOrRelPathMaster$$} The file path, absolute or relative to the master \LaTeX{} document.

\texttt{$$AbsOrRelPathParent$$} The file path, absolute or relative to the \LaTeX{} document.

\texttt{$$Tempname$$} A name and full path to a temporary file which will be automatically deleted whenever the containing document is closed, or the external material insertion deleted.

\texttt{$$Contents(\texttt{filename.ext})$$} This macro will expand to the contents of the file with the name \texttt{filename.ext}.

\texttt{$$Sysdir$$} This macro will expand to the absolute path of the system directory. This is typically used to point to the various helper scripts that are bundled with \LaTeX{}.
All path macros contain a trailing directory separator, so you can construct e.g. the absolute filename with \texttt{\$\$AbsPath\$\$Basename\$\$Extension}.

The macros above are substituted in all commands unless otherwise noted. The command \texttt{Product} supports additionally the following substitutions if they are enabled by the \texttt{Transform} and \texttt{TransformCommand} commands:

\texttt{\$\$ResizeFront} The front part of the resize command.
\texttt{\$\$ResizeBack} The back part of the resize command.
\texttt{\$\$RotateFront} The front part of the rotation command.
\texttt{\$\$RotateBack} The back part of the rotation command.

The value string of the \texttt{Option} command supports additionally the following substitutions if they are enabled by the \texttt{Transform} and \texttt{TransformOption} commands:

\texttt{\$\$Clip} The clip option.
\texttt{\$\$Extra} The extra option.
\texttt{\$\$Resize} The resize option.
\texttt{\$\$Rotate} The rotation option.

You may ask why there are so many path macros. There are mainly two reasons:

First, relative and absolute file names should remain relative or absolute, respectively. Users may have reasons to prefer either form. Relative names are useful for portable documents that should work on different machines, for example. Absolute names may be required by some programs.

Second, \LaTeX{} treats relative file names differently than \LaTeX{} and other programs in nested included files. For \LaTeX{}, a relative file name is always relative to the document that contains the file name. For \LaTeX{}, it is always relative to the master document. These two definitions are identical if you have only one document, but differ if you have a master document that includes part documents. That means that relative filenames must be transformed when presented to \LaTeX{}. Fortunately \LaTeX{} does this automatically for you if you choose the right macros.

So which path macro should be used in new template definitions? The rule is not difficult:

- Use \texttt{\$\$AbsPath} if an absolute path is required.
- Use \texttt{\$\$AbsOrRelPathMaster} if the substituted string is some kind of \LaTeX{} input.
- Else use \texttt{\$\$AbsOrRelPathParent} in order to preserve the user’s choice.
Chapter 6. Including External Material

There are special cases where this rule does not work and e.g. relative names are needed, but normally it will work just fine. One example for such a case is the command \texttt{ReferencedFile latex "\$\$AbsOrRelPathMaster\$\$Basename.pstex_t"} in the XFig template above: We can’t use the absolute name because the copier for \texttt{.pstex_t} files needs the relative name in order to rewrite the file content.

6.7 Security discussion

The external material feature interfaces with a lot of external programs and does so automatically, so we have to consider the security implications of this. In particular, since you have the option of including your own filenames and/or parameter strings and those are expanded into a command, it seems that it would be possible to create a malicious document which executes arbitrary commands when a user views or prints the document. This is something we definitely want to avoid.

However, since the external program commands are specified in the template configuration file only, there are no security issues if I\LaTeX{} is properly configured with safe templates only. This is so because the external programs are invoked with the \texttt{execvp-system} call rather than the \texttt{system system-call}, so it’s not possible to execute arbitrary commands from the filename or parameter section via the shell.

This also implies that you are restricted in what command strings you can use in the external material templates. In particular, pipes and redirection are not readily available. This has to be so if I\LaTeX{} should remain safe. If you want to use some of the shell features, you should write a safe script to do this in a controlled manner, and then invoke the script from the command string. In the \texttt{lib/scripts} directory of the I\LaTeX{} installation, you can find a safe wrapper script \texttt{general_command_wrapper.py} that supports redirection of input and output. That can serve as an example for how to write safe template scripts. For a more advanced example that uses \texttt{fork} and friends, take a look at the \texttt{pic2ascii.py} converter script.

It is possible to design a template that interacts directly with the shell, but since this would allow a malicious user to execute arbitrary commands by writing clever filenames and/or parameters, we generally recommend that you only use safe scripts that work with the \texttt{execvp system} call in a controlled manner. Of course, for use in a controlled environment, it can be tempting to just fall back to use ordinary shell scripts. If you do so, be aware that you will provide an easily exploitable security hole in your system. Of course it stands to reason that such unsafe templates will never be included in the standard I\LaTeX{} distribution, although we do encourage people to submit new templates in the open source tradition. But I\LaTeX{} as shipped from the official distribution channels will never have unsafe templates.

Including external material provides a lot of power, and you have to be careful not to introduce security hazards with this power. A subtle error in a single line in an innocent looking script can open the door to huge security
problems. So if you do not fully understand the issues, we recommend that you consult a knowledgable security professional or the LyX development team if you have any questions about whether a given template is safe or not. And do this before you use it in an uncontrolled environment.
Chapter 7

The \LaTeX{} Server

7.1 Introduction

The \LaTeX{} server is a method implemented in \LaTeX{} that will enable other programs to talk to \LaTeX{}, invoke \LaTeX{} commands, and retrieve information about the \LaTeX{} internal state. This is only intended for advanced users, but they should find it useful.

7.2 Starting the \LaTeX{} Server

The \LaTeX{} server works through the use of a pair of named pipes. These are usually located in your home directory and have the names “.lyxpipe.in” and “.lyxpipe.out”. External programs write into .lyxpipe.in and read back data from .lyxpipe.out. The stem of the pipe names can be defined in the \texttt{Tools}$\rightarrow$\texttt{Preferences} dialog, for example “/home/myhome/.lyxpipe”.

\LaTeX{} will add the ’.in’ and ’.out’ to create the pipes. The above setting also has the effect of activating the \LaTeX{} server. If one of the pipes already exists, \LaTeX{} will assume that another \LaTeX{} process is already running and will not start the server. To have several \LaTeX{} processes with servers at the same time, you have to change the configuration between the start of the programs.

If you are developing a client program, you might find it useful to enable debugging information from the \LaTeX{} server. Do this by starting \LaTeX{} as \texttt{lyx -dbg lyxserver}.

Warning: if \LaTeX{} crashes, it may not manage to remove the pipes; in this case you must remove them manually. If \LaTeX{} starts and the pipes exist already, it will not start any server.

Other than this, there are a few points to consider:

- Both server and clients must run on UNIX or OS/2 machines. Communications between \LaTeX{} on UNIX and clients on OS/2 or vice versa is not possible right now.
• On OS/2, only one client can connect to L\textsc{y}XServer at a time.

• On OS/2, clients must open inpipe with 0\textunderscore{}\texttt{WRONLY} mode.

You can find a complete example client written in C in the source distribution as \texttt{development/server_monitor.c}.

### 7.3 Normal communication

To issue a L\textsc{y}X call, the client writes a line of ASCII text into the input pipe. This line has the following format:

```
LYXCMD:clientname:function:argument
```

Here \textit{clientname} is a name that the client can choose arbitrarily. Its only use is that \textsc{ly}X will echo it if it sends an answer - so a client can dispatch results from different requesters.

\textit{function} is the function you want \textsc{ly}X to perform. It is the same as the commands you'd use in the minibuffer.

\textit{argument} is an optional argument which is meaningful only to some functions (for instance “self-insert” which will insert the argument as text at the cursor position.)

The answer from \textsc{ly}X will arrive in the output pipe and be of the form

```
INFO:clientname:function:data
```

where \textit{clientname} and \textit{function} are just echoed from the command request, while \textit{data} is more or less useful information filled according to how the command execution worked out. Some commands will return information about the internal state of \textsc{ly}X, such as “font-state”, while other will return an empty data-response. This means that the command execution went fine.

In case of errors, the response from \textsc{ly}X will have this form

```
ERROR:clientname:function:error message
```

where the \textit{error message} should contain an explanation of why the command failed.

Examples:

```
echo "LYXCMD:test:beginning-of-buffer:" > ~/.lyxpipe.in
echo "LYXCMD:test:get-xy:" > ~/.lyxpipe.in
read a < ~/.lyxpipe.out
```

```
```
7.4 Notification

LYX can notify clients of events going on asynchronously. Currently it will only do this if the user binds a key sequence with the function “notify”. The format of the string LYX sends is as follows:

\[ \text{NOTIFY: key-sequence} \]

where \textit{key-sequence} is the printed representation of the key sequence that was actually typed by the user.

This mechanism can be used to extend LYX’s command set and implement macros: bind some key sequence to “notify”, start a client that listens on the output pipe, dispatches the command according to the sequence and starts a function that may use LYX calls and LYX requests to issue a command or a series of commands to LYX.

7.5 The simple LYX Server Protocol

LYX implements a simple protocol that can be used for session management. All messages are of the form

\[ \text{LYXSRV: clientname: protocol message} \]

where \textit{protocol message} can be “hello” or “bye”. If “hello” is received from a client, LYX will report back to inform the client that it’s listening to it’s messages, while “bye” sent from LYX will inform clients that LYX is closing.
Appendix A

Bindings

This appendix is a huge cross-reference to all the English language keybindings. Originally, we simply wanted to list all of the key bindings followed by the function it’s bound to. That way, a user can look up a key to find out what it does. We then decided, what the hey, why not include the default toolbar and menu bindings, too. Please note this section is likely to be very out of date.

The form is really self-explanatory, but here are a few tips: all entries are arranged roughly alphabetically for a given modifier (C-a, C-b, etc.). For the general keyboard layout, simpler prefixes precede the more complex (C-s before C-S-c). All entries were gleaned from the default user interface and binding files located in the directories .../share/lyx/ui and .../share/lyx/bind; they should be treated as the final word on the bindings.

As a final note, be aware that some window managers (such as FVWM) take control of some of the function keys or motion keys. C-right is listed here as generating word-forward, but FVWM grabs it and uses it to change virtual desktops instead. Very annoying unless you instruct your window manager to stop intercepting such sequences.

A.1 Toolbar

Toolbar
Layouts
Icon "buffer-open"
Icon "buffer-write"
Icon "buffer-print"
Separator
Icon "cut"
Icon "copy"
Icon "paste"
Separator
Icon "font-emph"
APPENDIX A. BINDINGS

Icon "font-noun"
Icon "font-free"
Separator
Icon "tex-mode"
Icon "math-mode"
Separator
Icon "footnote-insert"
Icon "marginpar-insert"
Icon "depth-next"
Separator
Icon "figure-insert"
Icon "dialog-tabular-insert"

End

A.2 Menu

A.2.1 File

M-fa  buffer-write-as
M-fc  buffer-close
M-fd  buffer-reload
M-fe  file_export submenu
M-ff  buffer-export fax
M-fi  file_import submenu
M-fn  buffer-new
M-fo  buffer-open
M-fp  buffer-print
M-fs  buffer-write
M-ft  buffer-new-template
M-fv  file_vc submenu
       h   vc-history
       i   vc-check-in
       l   vc-revert
       o   vc-check-out
       r   vc-register
       u   vc-undo-last
M-fx  lyx-quit
A.2.2 Edit

M-e a paste
M-e c cut
M-e d redo
M-e e error-remove-all
M-e f find-replace
M-e h buffer-chktex
M-e i edit_floats submenu
   a tabular-feature append-row
   b tabular-feature toggle-line-bottom
   c tabular-feature align-center
   d tabular-feature delete-column
   e tabular-feature align-left
   i tabular-feature align-right
   l tabular-feature toggle-line-left
   m tabular-feature multicolumn
   n tabular-feature valign-center
   o tabular-feature valign-top
   r tabular-feature toggle-line-right
   t tabular-feature toggle-line-top
   u tabular-feature append-column
   v tabular-feature valign-bottom
   w tabular-feature delete-row
M-e l math-panel
   a floats-operate openfoot
   c floats-operate closefoot
   f floats-operate openfig
   m melt
   o open-stuff
   t floats-operate closefig
M-e o copy
M-e p dialog-preferences
APPENDIX A. BINDINGS

M-er reconfigure
M-es spellchecker
M-et edit_tabular submenu
M-eu undo
M-ex edit_paste submenu

| l | primary-selection-paste paragraph |

A.2.3 Insert

M-ia insert_floats submenu

| a | buffer-float-insert algorithm |
| d | buffer-float-insert wide-tab |
| f | buffer-float-insert figure |
| t | buffer-float-insert table |
| w | buffer-float-insert wide-fig |

M-ib dialog-tabular-insert
M-ic citation-insert
M-id math-display
M-ie buffer-child-insert
M-if footnote-insert
M-ig figure-insert
M-ih math-mode
M-ii index-insert
M-il label-insert
M-im marginpar-insert
M-in note-insert
M-io insert_toc submenu

| a | loa-insert |
| b | bibtex-insert |
| c | toc-insert |
A.2. MENU

\[ \text{f} \quad \text{lof-insert} \]
\[ \text{i} \quad \text{index-print} \]
\[ \text{t} \quad \text{lot-insert} \]
\[ \text{M-i r} \quad \text{reference-insert} \]
\[ \text{M-i s} \quad \text{insert_special submenu} \]
\[ \text{b} \quad \text{protected-space-insert} \]
\[ \text{e} \quad \text{end-of-sentence-period-insert} \]
\[ \text{h} \quad \text{hfll-insert} \]
\[ \text{i} \quad \text{dots-insert} \]
\[ \text{l} \quad \text{break-line} \]
\[ \text{m} \quad \text{menu-separator-insert} \]
\[ \text{p} \quad \text{hyphenation-point-insert} \]
\[ \text{q} \quad \text{quote-insert} \]
\[ \text{s} \quad \text{command-sequence math-insert ^;math-mode;} \]
\[ \text{u} \quad \text{command-sequence math-insert _;math-mode;} \]
\[ \text{M-i t} \quad \text{insert_file submenu} \]
\[ \text{l} \quad \text{file-insert-ascii lines} \]
\[ \text{p} \quad \text{file-insert-ascii paragraphs} \]
\[ \text{x} \quad \text{file-insert} \]
\[ \text{M-i u} \quad \text{url-insert} \]
\[ \text{M-i w} \quad \text{index-insert-last} \]
\[ \text{M-i x} \quad \text{external-insert} \]

A.2.4 Layout

\[ \text{M-l a} \quad \text{appendix} \]
\[ \text{M-l b} \quad \text{font-bold} \]
\[ \text{M-l c} \quad \text{layout-character} \]
\[ \text{M-l d} \quad \text{layout-document} \]
\[ \text{M-l e} \quad \text{font-emph} \]
\[ \text{M-l l} \quad \text{layout-preamble} \]
\[ \text{M-l n} \quad \text{font-noun} \]
APPENDIX A. BINDINGS

M-l p  layout-paragraph
M-l s  layout-save-default
M-l t  layout-tabular
M-l v  depth-increment
M-l x  tex-mode

A.2.4.1 Layout > Character
M-c b  font-bold
M-c c  font-noun
M-c e  font-emph
M-c m  math-mode
M-c p  font-code
M-c r  font-roman
M-c s  font-sans
M-c u  font-underline
M-c space font-default
M-c Down word-lowcase
M-c Up  word-upcase
M-c Right word-capitalize
M-s h  font-size huge
M-s l  font-size large
M-s n  font-size normal
M-s s  font-size small
M-s t  font-size tiny
M-s 0  font-size huger
M-s l  font-size tiny
M-s 2  font-size smallest
M-s 3  font-size smaller
M-s 4  font-size small
A.2. **MENU**

M-s 5  font-size normal
M-s 6  font-size large
M-s 7  font-size larger
M-s 8  font-size largest
M-s 9  font-size huge
M-s S-H  font-size huger
M-s S-L  font-size larger
M-s S-S  font-size smaller
M-s plus  font-size increase
M-s minus  font-size decrease

A.2.5  View
A.2.6  Navigate
A.2.7  Help

A.2.8  **Paragraph Style**
M-p a  layout Abstract
M-p b  layout Itemize
M-p c  layout LyX-Code
M-p d  layout Description
M-p e  layout Enumerate
M-p f  layout ShortFoilhead
M-p i  layout Itemize
M-p l  layout List
M-p n  layout Enumerate
M-p q  layout Quote
M-p r  layout ShortRotatefoilhead
M-p s  layout Standard
M-p t  layout Title
M-p v layout Verse
M-p space drop-layouts-choice
M-p 0 layout Part
M-p 1 layout Chapter
M-p 2 layout Section
M-p 3 layout Subsection
M-p 4 layout Subsubsection
M-p 5 layout Paragraph
M-p 6 layout Subparagraph
M-p asterisk 0 layout Part*
M-p asterisk 1 layout Chapter*
M-p asterisk 2 layout Section*
M-p asterisk 3 layout Subsection*
M-p asterisk 4 layout Subsubsection*
M-p asterisk 5 layout Paragraph*
M-p asterisk 6 layout Subparagraph*
M-p S-A layout Author
M-p S-B layout Bibliography
M-p S-C layout Comment
M-p S-D layout Date
M-p S-F layout Foilhead
M-p S-L layout LaTeX
M-p S-Q layout Quotation
M-p S-R layout Rotatefoilhead
M-p C-a layout RightAddress
M-p M-a layout Address
M-p M-c layout Caption
M-p Left depth-decrement
A.3. KEYBOARD

M-p Right depth-increment

These ones are kept for backwards compatibility, but only make sense on a
qwerty keyboard:

M-p S-at layout Section*
M-p S-dollar layout Subsubsection*
M-p S-numbersign layout Subsection*

A.3 Keyboard

A.3.1 Specific to emacs.bind

C-a       line-begin
C-b       char-backward
C-d       delete-forward
C-e       line-end
C-f       char-forward
C-g       cancel
C-h       hyphenation-point-insert
C-i       hfill-insert
C-k       line-delete-forward
C-l       screen-recenter
C-m       mark-toggle
C-n       down
C-o       open-stuff
C-p       up
C-q       quote-insert
C-s       find-replace
C-u       font-underline
C-v       screen-down
C-w       cut
C-y paste
C-S-Y layout-paste
C-x a buffer-auto-save
C-x b buffer-previous
C-x c lyx-quit
C-x d buffer-new
? C-x f buffer-open
C-x g buffer-view-ps
C-x k buffer-close
C-x p buffer-view
C-x r buffer-typeset
? C-x s buffer-write
C-x t buffer-typeset
C-x u undo
C-x v c vc-undo-last
C-x v h vc-history
C-x v i vc-register
C-x v u vc-revert
C-x v v vc-check-in
? C-x w buffer-write-as
C-x bracketleft screen-up
C-x bracketright screen-down
C-x C-a buffer-auto-save
C-x C-b menu-open Documents
C-x C-c lyx-quit
C-x C-d buffer-new
C-x C-f buffer-open
C-x C-g buffer-view-ps
A.3. KEYBOARD

C-x C-p    buffer-view
C-x C-q    buffer-toggle-read-only
C-x C-s    buffer-write
C-x C-t    buffer-typeset
C-x C-w    buffer-write-as
Home       buffer-begin
End         buffer-end
S-Home     line-begin-select
S-End       line-end-select
S-Up        up-select
S-Down      down-select
S-Next      screen-down-select
S-Prior     screen-up-select
S-Left      backward-select
S-Right     forward-select
C-Up        paragraph-up
C-Down      paragraph-down
C-Left      word-backward
C-Right     word-forward
C-Delete    word-delete-forward
C-BackSpace word-delete-backward
C-Return    break-line
C-period    end-of-sentence-period-insert
C-space     protected-space-insert
C-S-at      mark-on
C-S-greater label-goto
C-S-less    reference-back
C-S-slash  undo
APPENDIX A. BINDINGS

C-S-underscore undo
C-S-quotedbl quote-insert
C-S-Home buffer-begin-select
C-S-End buffer-end-select
C-S-Up paragraph-up-select
C-S-Down paragraph-down-select
C-S-Left word-backward-select
C-S-Right word-forward-select
Escape meta-prefix
M-d word-delete-forward
M-w copy
M-x command-execute
M-S-W layout-copy
M-period dots-insert
M-Return break-paragraph-keep-layout
M-S-percent find-replace

A.3.2 Specific to cua.bind
C-b font-bold
C-c copy
C-d buffer-view
C-e font-emph
C-f find-replace
C-g error-next
C-i open-stuff
C-k font-noun
C-l tex-mode
C-m math-mode
C-n buffer-new
A.3. KEYBOARD

C-o  buffer-open
C-p  buffer-print
C-q  lyx-quit
C-r  buffer-reload
C-s  buffer-write
C-t  buffer-view-ps
C-u  font-underline
C-v  paste
C-w  buffer-close
C-x  cut
C-z  undo
C-space  protected-space-insert
C-S-C  layout-copy
C-S-D  buffer-typeset
C-S-M  math-display
C-S-N  buffer-new-template
C-S-P  font-code
C-S-S  buffer-write-as
C-S-T  buffer-typeset-ps
C-S-V  layout-paste
C-S-Z  redo
S-Insert  paste
S-Delete  cut
C-period  end-of-sentence-period-insert
C-S-greater  label-goto
C-S-less  reference-back
C-minus  hyphenation-point-insert
C-S-quotedbl  quote-insert
APPENDIX A. BINDINGS

C-S-space  protected-space-insert
M-x  command-execute
S-Home  line-begin-select
S-End  line-end-select
C-S-Home  buffer-begin-select
C-Home  buffer-begin
S-Up  up-select
S-Right  forward-select
S-Delete  cut
S-Insert  paste
C-Insert  copy
S-Next  screen-down-select
S-Prior  screen-up-select
C-S-Next  paragraph-down-select
C-Next  paragraph-down
S-Delete  word-delete-forward
C-Delete  word-delete-forward-backward
C-S-Delete  paragraph-down-select
C-Delete  buffer-end-select
C-S-Delete  buffer-begin-select
C-Delete  word-backward-select
C-S-Delete  word-backward-forward
C-Delete  word-forward-select
C-Delete  paragraph-up-select
A.3. KEYBOARD

C-S-quotedbl quote-insert
M-Return break-paragraph-keep-layout
M-period dots-insert
M-S-Right depth-increment
M-S-Left depth-decrement
Escape cancel
F2 buffer-write
F3 buffer-open
F5 screen-recenter
F7 spellchecker
F9 meta-prefix
C-F4 buffer-close
M-F4 lyx-quit

A.3.3 Specific to sciword.bind

These are $\LaTeX$X keyboard definitions for mathematics, similar to those of Scientific Word.

The bindings file and the present documentation were prepared by Serge Winitzki with assistance from Jean-Marc Lasgouttes. Version 1.3, for $\LaTeX$X 1.2.x and 1.3.x.

These definitions make it a lot easier to type equations without using the mouse, especially for people familiar with Scientific Word. The standard $\LaTeX$X bindings such as M-m or M-o are unmodified.

Tip: to find the "$\LaTeX$X bind name" for a key, look at the status bar after typing some non-existent key combination. E.g. to find how "Ctrl-\&" is referenced, press Ctrl-S and then Ctrl-\&: the status bar shows "C-s S-C-ampersand." (This does not work in $\LaTeX$X 1.3.0!)

C-c copy – Copy, cut, paste is as in Sciword, C-c, C-x, and C-v.

C-d math-display – Display equation toggle: type C-d to insert a displayed formula (d for “displayed”). You can also type C-d in a displayed formula to convert it back to an inline formula.

C-f math-insert \frac – Fractions: type C-f to insert a fraction (f for “fraction”). You can also select an expression and type C-f to convert it to the numerator of a fraction. Note: pressing Backspace at the left end of the denominator will delete the numerator and convert the denominator to a non-fraction.
APPENDIX A. BINDINGS

C-i \math-insert \int – Inserts \( f \) (i for “integral”)

C-k \textbf{line-delete-forward} – Emacs-like binding: delete forward of cursor to end of line.

C-m \textbf{math-mode} – A text/math toggle (m for “math”): switches to math in text mode, and also inserts roman text in math mode. Also bound to C-t (t for “text”).

C-n \textbf{math-number} – Add/remove numbering in a single equation.

S-C-N \textbf{math-nonumber} – Add/remove numbering at a line in equation arrays.

The above commands are toggles that control the numbering of equations (N for “number”). Note: when deleting a number in a labeled eqnarray, the label is not really removed (the \LaTeX{} code becomes “\label{} \nonumber”) and this generates a (harmless) \LaTeX{} warning.

C-o \textbf{file-open} – Open a new document. (W*ndows)

C-q \textbf{quote-insert} – Insert a quote character “ (q for “quote”). This is not the “smart” double quote character that you get by default.

C-r \textbf{math-insert} \sqrt{} – Square root sign \( \sqrt{x} \) (r for “root”).

S-C-R \textbf{math-insert} \root{} – Root sign \( \sqrt[n]{x} \).

C-t \textbf{math-mode} – Another binding for a switch between the text and the math mode (t is for “text”). Note that the roman text inserted in math mode is special.

C-u \textbf{font-underline} – Underline the selected text (text mode only, use things like \overline{} or \underbar{} for math).

C-v \textbf{paste} – W*ndows heritage.

C-w \textbf{buffer-close} – Close the current document (again, a W*ndows heritage).

C-x \textbf{cut}

C-z \textbf{undo}

S-C-Z \textbf{redo} – the “Redo” operation, or “undo the undo”.

Bracket delimiters: press Ctrl-<bracket key> to insert a matching pair of delimiters. For example, Ctrl-[ inserts a pair of parentheses [ ]. (Note: Ctrl-] does the same thing.) It will switch to math mode if needed. The supported characters are ( ( ] ) { } < |. It is the same to press the right or the left bracket. The corresponding delimiters are ( ) [ ] { } |. The delimiters are “smart” and resize with their contents. Use Math Panel to get other or non-matching delimiters. Press backspace on the left delimiter to remove both “smart” delimiters without removing their contents.
A.3. KEYBOARD

C-9 \texttt{math-delim ( )} – for convenience, pressing Ctrl-9 is the same as Ctrl- ( or Ctrl-)

S-C-parenleft \texttt{math-delim ( )}

C-0 \texttt{math-delim ( )}

S-C-parenright \texttt{math-delim ( )}

C-bracketleft \texttt{math-delim [ ]}

C-bracketright \texttt{math-delim [ ]}

C-S-less \texttt{math-delim langle rangle} – angular delimiters (), not to confuse with ordinary < > signs.

C-S-greater \texttt{math-delim langle rangle}

The bar bracket: on some keyboards (e.g. some British ones), the bar character is bound to an Alt-something and on some wayward Unices to "brokenbar". So we define all of these keys as well.

C-S-bar \texttt{math-delim | |}

C-S-brokenbar \texttt{math-delim | |}

C-M-bar \texttt{math-delim | |}

S-C-braceleft \texttt{math-delim { }}

S-C-braceright \texttt{math-delim { }}

Accents are in most cases Ctrl-<accent key>, e.g. Ctrl-. for overdot, Ctrl-’ for acute accent, Ctrl-~ for tilde (you also need to press Shift here) etc. For example, “Ctrl-’ a” inserts á. Some accents work only in math mode and others only in text mode.

C-period \texttt{accent-dot} – overdot accent, á (text mode only).

C-comma \texttt{math-insert \dot} – overdot accent, á (math mode only – in physics this denotes a first derivative).

C-equals \texttt{math-insert \overrightarrow} – Vector accent over math \rightarrow.

S-C-quotedbl \texttt{accent-umlaut} – umlaut accent, ä (text mode only)

S-C-colon \texttt{math-insert \ddot} – double dot accent, á (math mode only – in physics this denotes a second derivative). To get a triple dot in math mode, use \dddot.

C-quoteleft \texttt{accent-grave} – grave accent à (text mode only, use \grave for math)
APPENDIX A. BINDINGS

S-C-asciitilde accent-tilde – tilde accent à (text mode only, use \tilde for math)

C-apostrophe accent-acute – acute accent á (text mode only, use \acute for math)

S-C-asciicircum accent-circumflex – circumflex (caret) accent â (text mode only, use \hat for math).

Function keys. The new key S-F2 for creating a \LaTeX file seems handy.

F2 \texttt{buffer-write} – Save current document.

S-F2 \texttt{buffer-export latex} – Write a \LaTeX file for the current document.

F3 \texttt{find-replace} – Find and replace dialog.

C-F4 \texttt{buffer-close} – same as C-w.

M-F4 \texttt{lyx-quit} – Alt-F4 to quit LYX is the W*=ndows w*=ndow manager’s mnemonic.

F4 to F8 are used to switch fonts. Use F4 to switch back to the normal font. The non-default font switches F5 - F8 all work as toggles. They also work on the whole word if you put the cursor in the middle of the word, or if you select some text.

F4 \texttt{font-default} – stop using any special font

F5 \texttt{font-bold} – make \textbf{bold} text.

F6 \texttt{font-emph} – make \textit{emphasized} text.

F7 \texttt{font-code} – make typewritten text.

F8 \texttt{font-noun} – make \textsc{Caps/small caps} text. (Used sometimes for people’s names.)

F9 is bound as “meta-prefix”, same as the Alt key (useful e.g. if the keyboard has no working Alt).

Here are some Sciword-inspired mnemonics for frequently used math symbols. Many symbols start with a C-s sequence. Therefore C-s cannot be itself bound to anything.

M-apostrophe math-insert \prime – The “prime” symbol ′ in math mode. This is frequently unnecessary: in most cases the normal apostrophe works just fine, e.g. \texttt{x′ + 2x = 0}, but in some cases this would generate a double superscript error in \LaTeX. For example: \texttt{x′2} (x prime squared) must be entered with the prime character.
C-s apostrophe command-sequence math-superscript; math-insert \prime;
- Insert a prime as a superscript (see example above).

C-Up          math-insert ^ - Insert an upper index. Also, ^
C-Down        math-insert _ - Insert a lower index. Also, _
C-s d          math-insert \partial - Partial derivative symbol ∂.
C-s e          math-insert \sum - Summation symbol ∑ (not the same as the
Greek letter uppercase Sigma Σ because it can resize and allows
smart upper/lower limits).
C-s p          math-insert \prod - Product symbol Π (not the same as the
Greek letter uppercase Pi, Π).
C-s i          math-insert \infty - Infinity ∞.
C-s x          math-insert \times - Cross product ×.
C-s v          math-matrix 1 2 - Insert a stacked array . (Frequently useful in
formulae.)
C-s m          math-matrix 3 3 - Insert a 3×3 matrix . (Then you can modify
its size using the Edit>Math menu.)
C-s S-plus     math-insert \dagger - The “dagger” †.
C-s equal      math-insert \equiv - “Identical equality” ≡.
M-equal       math-insert \approx - “Approximate equality” ≈.
M-minus       math-insert \sim - The “of order” sign ∼.
C-minus       math-insert \rightarrow - The arrow → as in \lim_{x \to 0}.
S-M-less      math-insert \leq - Less-or-equal ≤.
S-M-greater   math-insert \geq - Greater-or-equal ≥.
C-s S-less    math-insert \ll - “Much less than” ≪ (useful in physics)
C-s S-greater math-insert \gg - “Much greater than” ≫.
A.3.4 Standard math bindings

M-m b  math-insert \overline
M-m d  math-display
M-m e  math-insert ^
M-m f  math-insert \frac
M-m g  math-greek
M-m h  accent-circumflex
M-m i  math-insert \int
M-m l  math-limits
M-m m  math-mode
M-m n  math-number
M-m o  math-insert \oint
M-m p  math-insert \partial
M-m r  math-insert \sqrt
M-m s  math-insert \sqrt
M-m u  math-insert \sum
M-m v  math-insert \vec
M-m x  math-insert _
M-m 8  math-insert \infty
M-m S-G math-greek-toggle
M-m S-N math-nonumber
M-m S-period accent-dot
M-m S-asciitilde accent-tilde
M-m S-apostrophe math-insert \prime
M-m S-parenleft math-delim ( )
M-m S-bracketleft math-delim [ ]
M-m S-braceleft math-delim { }
M-m S-less math-delim langle rangle
A.3. KEYBOARD

M-m S-greater math-delim rangle langle
M-m S-bar math-delim ||
M-m S-plus math-insert \pm
M-m S-equal math-insert \neq

A.3.5 Other Accelerators
M-k o keymap-off
M-k t keymap-toggle
M-k x keymap-off
M-k 1 keymap-primary
M-k 2 keymap-secondary