Extended LyX Features

by the LyX Team[1]

June 1, 2006

[1]Principal maintainer of this file is Mike Ressler. If you have comments or error corrections, please send them to the LyX Documentation mailing list, <lyx-docs@lists.lyx.org>.

Contents

Chapter 1

Introduction

The *Extended LyX Features* manual, which you are now reading, is essentially Part II of the *User's Guide.* The reason for splitting this document is simple: the *User's Guide* is already huge, and it contains all of the basic features one needs to know in order to prepare most documents. However, the LyX Team has a long-term goal of making LyX extensible through various configuration files and external packages. That means that if you want to support the Fizzwizzle LaTeX package, you can create a layout file for it without having to alter LyX itself. We've already had contributions of several new features this way. This is the place where all of that gets documented.

This manual also documents some special features, like fax support, version control, and SGML support, which require additional software to work properly. Lastly, there's a chapter of LaTeX tools and tips, things you can use to spruce up your documents by directly using the powerful features of LaTeX. After all, LyX *is* only WYSIWYM, and will only ever interface to certain LaTeX features.

Of course, with all of this extra documentation, *Extended LyX Features* may itself grow too big for its britches. In that case, you can just call it the "Overextended Manual" for fun!

If you haven't read the *Introduction* yet, you are definitely in the wrong manual. The *Introduction* is the first place to go, since it will direct you to the correct manual, and it also describes the notation and format of all of the manuals. You should also be thoroughly familiar with the *User's Guide* and all of the basic features of LyX.

In this document, many sections are independent articles contributed by an individual and are noted as such. This person is generally whoever wrote the layout file for the new document class or LaTeX package, or implemented the feature. If there is no mention of an author to a chapter [or chapter sections], that means it was written by the LyX Documentation Team.

Since all the topics in this manual depend heavily on LyX's interaction with LaTeX, this first chapter covers the inner workings of LyX and how to direct LyX to generate exactly the LaTeX code you want. It is obviously for more seasoned LyX users.

Chapter 2

LyX and LaTeX

2.1 How LyX Uses LaTeX

This chapter is for both TeX-nicians and the LaTeX-curious. In it, we'll explain how LyX and LaTeX work together to produce printable output. This is the only place in any of the manuals where we assume you know something about LaTeX.

At one time, we called LyX a "WYSIWYM frontend to LaTeX," but that's no longer true. There are frontends to LaTeX out there. They are basically editors with the ability to run LaTeX and mark any errors in the file you're editing. Although LyX *is* an editor, and it *does* run LaTeX, and it also marks errors in the file, it also does much, much more. Thanks to the WYSIWYM concept, you don't need LaTeX to use LyX effectively. LyX has also added a few extensions to LaTeX. Try the following sometime: select Export ▷ LaTeX from the File menu, then look at the preamble of the resulting `.tex` file. You'll notice a variety of new macros defined specifically by LyX. These macros are defined automatically, according to the features you use in the document.

There are several commands that automatically invoke LaTeX. They are:

- View ▷ View *Format*
- View ▷ Update ▷ *Format*
- File ▷ Print
- File ▷ Fax

They will only invoke LaTeX if the file has changed since the last time LaTeX was run.

When you run LaTeX on the file you're editing, LyX performs these steps:

1. Convert the document to LaTeX and save to a file with the extension `.tex` in place of `.lyx`.

2. Run LATEX on the `.tex` file (maybe several times).
3. If there are any errors, insert error boxes in the document to mark where they are. These boxes are transient and are not saved along with the document.

If you've run LATEX using View DVI, LYX then executes `xdvi` on the Dvi file. If you've used View PostScript or Print, LYX performs two more steps:

- Run `dvips` to convert the Dvi file to PostScript®:
  - For View PostScript, the output file has the extension `.ps_tmp`
  - For Print , the output file has the extension `.ps`, as expected.
- Execute `ghostview` or send the PostScript® file to the printer.

2.2 "Help! LYX generated an unreadable `.tex` file!"

Die-hard LATEX users will scream and howl this into the night, then declare LYX useless, simply because they didn't RTFM.

We're going to set the record straight. LYX produces two kinds of LATEX files. One is human readable. The other is LYX readable. Every time LYX executes LATEX, it produces a LATEX file that it can easily scan for errors. The resulting `.tex` file is not human readable. Don't even try to read it. If you want a `.tex` file that you can send to a colleague, select Export ▷ LaTEX from the File menu.

2.3 Translating LATEX files into LYX

You can import a LATEX file into LYX by using the File ▷ Import ▷ LaTEX command in LYX. This will call a Perl script named `reLyX`—which will create a file `foo.lyx` from the file `foo.tex`—and then open that file. If the translation doesn't work, you can try calling `reLyX` from the command line, possibly using fancier options.

`reLyX` will translate most legal LATEX, but not everything. It will leave things it doesn't understand in TEX mode, so after translating a file with `reLyX`, you can look for red text and hand-edit it to look right.

`reLyX` has its own section in the *Extended Features* manual (as well as a Unix manpage equivalent), which you should read to find out about what LATEX isn't supported, bugs (and how to get around them), and how to use the various options.

If you can't get `reLyX` to work, or you just want to put a piece of LATEX code into a LYX file, see Section2.4.

2.4 Inserting LaTeX Code into LyX Documents

This is a rather important point: You can always insert LaTeX code into any LyX document. LyX simply cannot, and will probably never be able to, display every possible LaTeX construct. If ever you need to insert LaTeX commands into your LyX document, you can use the ERT box, which you can insert into your document with Insert ▷ TeX. The ERT box comes in three forms: collapsed, open, and inlined. The first two are used just like any other collapsable (foldable) box (such as footnotes), and are useful for significant amounts of LaTeX commands. An "inlined" ERT box displays its content as part of the button, and is useful for very short sections of LaTeX commands.

You can switch between all three by right-clicking on the ERT. Note that if you want more than one line of LaTeX commands, you cannot use the inlined mode.

Here's an example of inserting LaTeX commands in a LyX document. The code looks like this:

```
\begin{tabular}{ll}
\begin{minipage}{5cm}
This is an example for a minipage environment. You
can put nearly everything in it, even (non-floating)
figures and tables.
\end{minipage}
&
\begin{minipage}{5cm}
\begin{verbatim}
\begin{minipage}{5cm}
This ...
\end{minipage}
\end{verbatim}
\end{minipage}
\end{tabular}
```

The ERT box containing this text is directly after this paragraph. Those of you reading the manual online will only see a bunch of funky text in red. Those reading a printed version of the manuals will see the actual results:

This is an example for a minipage environment. You can put nearly everything in it, even (non-floating) figures and tables.

```
\begin{minipage}{5cm}
This ...
\end{minipage}
```

In addition to these two methods, you can also create a separate file containing some complex LaTeX structure. You can then use Insert ▷ Child Document to include your file (you should select the type Input). We recommend that you only do this if you have a `.tex` file which you *know* works already. Otherwise, you'll have a big job tracking down LaTeX errors…

There are a few last notes to emphasize:

- Inside of LyX, LaTeX code appears *in red*
- LyX *does not* check if your LaTeX code is correct.
- Beware reinventing the wheel.

That last note refers to two things. First, LyX does have quite a few features tucked into it, and more are coming. Be sure to check the manuals to make sure that LyX doesn't have such-and-such feature before you go off merrily coding LaTeX. Second, there are numerous LaTeX packages out there to do all sorts of things, from labels to envelopes to fancy multipage tables. Check out a CTAN site for details (see Section "Requirements" of the *User's Guide*).[1]

If you do need to do some wild and fancy things within your document, be sure to check out a good LaTeX book for assistance. There are a number of them listed in the bibliography of the *User's Guide*.

There are a number of LaTeX commands which have to be placed before the beginning of the actual text. They go into the preamble, and this is explained in the next section.

2.5 LyX and the LaTeX Preamble

2.5.1 About the LaTeX Preamble

If you already know LaTeX, there is no need to explain here what the preamble is good for. If you don't, the following will give you some ideas — we recommend again that you consult a LaTeX book for further information. In any case, you should read the points below, because they explain what you can do and what you don't need to do in the LaTeX preamble of a LyX document.

The LaTeX preamble comes at the very beginning of a document, *before* the text. It serves to:

- declare the document class. LyX already does this for you.

  If you're a seasoned LaTeX-nician, and you have some custom document class you want to use, check out the *Customization Manual* for information on how to make LyX interface to it. Be sure to submit your efforts to the LyX Team for inclusion in future versions!

- declare the usage of packages. LaTeX packages provide special commands, which are only available within a document when the package has been declared in the preamble. For example, the package `indentfirst` forces all paragraphs to be indented. There are other packages for labels, envelopes, margins, etc.

[1]Note from John Weiss: I seem to do this an awful lot. Sat down and merrily began coding something to print out labels, only to learn that there were already 2 different LaTeX packages to do this. Worse yet — I had them already!

- set counters, variables, lengths and widths. There are several LaTeX counters and variables which *must* be set globally from within the preamble in order to have the desired effect. [There are other variables which you can set and reset inside the document, too.] Margins are a good example of something which must be set in the preamble. Another example is the label format for lists. You can actually set these just about anywhere, but it's best to do it just once, inside the preamble.

- declare user defined commands [with `\newcommand` or `\renewcommand`], mostly abbreviations for LaTeX commands which appear very often inside a document. Although the preamble is a good place to declare such commands, they *can* be declared anywhere else [but *before* they are used for the first time, of course...]. This can be useful if there is a lot of raw LaTeX code in your document, which normally should not be the case.

LyX adds its own set of definitions to the preamble of the `.tex` file it produces. This makes LaTeX files generated by LyX portable.

2.5.2 Changing the Preamble

The commands which LyX adds to the preamble of a LaTeX file are fixed; you can't change them without patching LyX itself. You can, however, add your own stuff to the preamble. There are two ways to do this:

1. Select LaTeX Preamble from the Document menu, or via the Document ▷ Settings dialog, depending on your frontend. Note that the LyX keybindings will not work in this dialog, alas.

2. Use the preamble contents you've added as your default template (see "Basic LyX Setup" in the *User's Guide*), so that it will be the default preamble for any file you create.

LyX adds anything in the Preamble dialog to its own built-in preamble. Before adding your own declarations in the preamble, you should make sure that LyX doesn't already support what you want to do (remember what we said about reinventing the wheel?). Also, *make sure your preamble code is correct.* LyX doesn't check it.

2.5.3 Examples

Here are some examples of what you can add to a preamble, and what they do:

2.5.3.1 Example #1: Offsets

There are two variables under LaTeX that control page position: `\hoffset` and `\voffset`. Their names should be self-explanatory. These variables are useful if you think for a moment about computer labels. Sometimes, the size of a print medium and the area of the medium that you can actually print on aren't the same. This is where `\hoffset` and `\voffset` come in.

The default values for `\hoffset` and `\voffset` are both 0 pt., i. e. the page isn't shifted.

Unfortunately, some DVI drivers always seem to shift the page. We have no idea why, or why the sysadmin hasn't fixed such behavior. If you're using LyX on a system that you don't personally maintain, and your sysadmin is a doofus, `\hoffset` and `\voffset` can save the day. Suppose you're left and top margins are always 0.5 inches too big. You can add this to the preamble:

```
\setlength{\hoffset}{-0.5 in}
\setlength{\voffset}{-0.5 in}
```

. . . and your margins should now be correct.

2.5.3.2 Example #2: Labels

Speaking of labels, suppose you wanted to print out a bunch of address labels. There's a rather nice package, available at your nearest CTAN archive, for printing sheets of labels, called `labels.sty`. Now, your system may not have this package installed by default. We leave that up to you to check. You'll also want to read the documentation for it; we're not going to do that for you. Since this is an example, however, we'll give you an example of how you use this package.

First, make sure you're using the article document class. Next, you need to put the following in your preamble:

```
\usepackage{labels}
\LabelCols=3
\LabelRows=7
\LeftBorder=8mm
\RightBorder=8mm
\TopBorder=9mm
\BottomBorder=2mm
```

This sets things up for Avery® label sheets, stock #5360. You're now ready to print labels, but you'll need to insert LaTeX code, placing the commands `\begin{labels}` and `\end{labels}` around each label text. This and other special features of `labels.sty` are explained in its documentation.

Someday, someone may write a LyX layout file to support this package directly. Maybe that someone is you.

2.5.3.3 Example #3: Paragraph Indentation

Americans are trained to indent the first line of *every* paragraph. As with all of their other weird quirks, most Americans will whine and moan until they can have their way and indent the first line of all paragraphs.[2]

Of course, this behavior isn't standard typography. In books, you typically only indent the first line of a paragraph *if* it follows another one. The idea behind indenting the first line of a paragraph is to distinguish neighboring paragraphs from one another. If there is no previous paragraph, for example, it follows a figure, or is the first paragraph in a section, then there is no special indentation.

If you're a typical American, though, you don't care about such esoteric things; you want your indentation! Add this to the preamble:

```
\usepackage{indentfirst}
```

If your TEX distribution isn't a braindead one, you'll have this package, and all of your paragraphs will get the indentation you think they deserve.

2.5.3.4 Example #4: This Document

You can also check out the preamble of this document to get an idea of some of the advanced things you can do. You'll probably need to make the Preamble... dialog full-screen to see most of it. Also, there are more examples and an assortment of LATEX "dirty tricks" given in Chapter 7.

2.6 LYX and LATEX Errors

When LYX calls LATEX, it tells LATEX to blithely ignore any errors and keep going. It then uses the log-file from the LATEX run to do a post-mortem. As we stated earlier in the chapter, LYX generates two kinds of `.tex` files, one of which it uses to locate errors in the document. If there was an error someplace, LYX will put a box with the word "Error" at the appropriate place in the document.[3] It will also display a message alerting you to the fact that there were errors.

You can navigate through the errors by using Error in the Navigate menu. You can "open" the error-boxes and view the error message LATEX produced by clicking on it.

Some folks also like to look at the log file directly, accessible from Document ▷ LaTEX Log File. There are some fairly common error messages and warnings. We'll cover those here. You should look at a good LATEX book for a complete listing.

- "`LaTeX Warning:`"

  Anything beginning with these word is a warning message for the purpose of "debugging" the LATEX code itself. You'll get messages like this if you

[2]Note from JOHN WEISS: This was written by an American — *me*! It's my perception of my fellow countrymen. Tough if you don't like it. Thpbpbpbpbpbpbpbp!

[3]LYX will occasionally misguess where the error was. This will typically happen with tables, figures, math, and the preamble.

added or changed cross-references or bibliography entries, in which case, LaTeX is trying to tell you that you need to make another run.

You can by-and-large ignore these.

- "`LaTeX Font Warning:`"

  Another warning message, this time about fonts which LaTeX couldn't find. The rest of the message will often say something about a replacement font that LaTeX used.

  You can safely ignore these.

- "`Overfull \hbox`"

  LaTeX absolutely *loves* to spew these out. They are warning you about lines that were too long and run past the right margin. Almost always, this is unnoticeable in the final output. Or, only one or two characters extend past the margin. LaTeX seems to generate at least one of these messages for just about any document you write.

  You can ignore these stupid messages. Your eyes will tell you if there's a problem with something that's too wide; just look at the output.

- "`Underfull \hbox`"

  Not quite as common as its cousin. LaTeX seems to like to print lines that are a bit too wide as opposed to ones that are a bit too narrow. We have no idea why.

  You can ignore these, too.

- "`Overfull \vbox`" and "`Underfull \vbox`"

  Warnings about troubles breaking the page. Once again, just look at the output. Your eyes will tell you where something has gone wrong.

- " `LaTeX Error: File 'Xxxx' not found`"

  The file "Xxxx" isn't installed on this system. This usually appears because some package your document needs isn't installed. If you didn't touch the preamble or didn't use the `\usepackage{}` command, then one of the packages LyX tried to load is missing. Use Help ▷ LaTeX Configuration, to get a list of packages that LyX knows about. This file is updated whenever you reconfigure LyX (using Tools ▷ Reconfigure) and tells you which packages have been detected and what they do.

  If you *did* use the `\usepackage{}` command, and the package in question isn't installed, you'll need to install it yourself.

- "`LaTeX Error: Unknown option`"

  Error messages beginning with this are trying to tell you that you specified a bad or undefined option to a package. Check the package's documentation.

- "`Undefined control sequence`"

  If you've inserted LaTeX code into your document, but made a typo, you'll get one of these. You may have forgotten to load a package. In any case, this error message usually means that you used an undefined command.

There are other error and warning messages. Some are self-explanatory. These are usually LaTeX messages. Others are downright cryptic. These are actually TeX error messages, and we really have *no clue* what they mean or how to decipher them.

There's a general sequence you should follow if you get error messages:

1. Look at the LaTeX code you inserted for typos.
2. If there are no typos, check and see that you used the command(s) correctly.
3. If you get a bunch of error boxes piled up at the very top of the document, it means that there are errors in the preamble. Start debugging your preamble.
4. If you didn't add anything to the preamble and didn't add any LaTeX code to the document, the first suspect is your LaTeX distribution itself. Check for missing packages and install them.
5. Okay, so there are no missing packages. Did you use any of the fine-tuning options in LyX? Specifically, did you *misuse* any of them, like trying to manually insert lots of Protected Blanks, Linebreaks, or Pagebreaks? Did you try to kludge something together with these instead of using the appropriate paragraph environment?
6. All right, you didn't use any of the fine-tuning options, you played by the rules. Did you try to pull a fancy maneuver? Did you do something funky inside a table or an equation, like inserting a graphic into a table cell?
7. Do you have long sections of text where LaTeX cannot find a place to break a line? By default, LaTeX is rather strict about how much extra inter-word spacing it will add in order to break a line. Preferrably, you should rework the paragraph to avoid the problem. If this isn't an option, you can wrap your text in `\sloppypar` to make LaTeX's line breaking more, well, sloppy.
8. Did you go overboard with the nesting? LyX (currently) doesn't check to make sure you're in the limits for nesting environments. If you nested a bunch of environments to the 17th level, that's the problem.
9. Okay, you didn't get any error messages, but your output looks whacked. If you have a table or figure that's too wide or long for the page, you need to:

   (a) rescale the figure so it fits.

   (b) trim down the table so it fits.

   If something else is wrong with the output, and you didn't try to pull anything fancy or kludge the fine-tuning options, we're not sure what's wrong.

If all this doesn't help — well, then *perhaps* you might have found a bug in LYX...

Chapter 3

Supplemental Tools

3.1 Preparing a Bibliography with BibTEX

by Mike Ressler and Jürgen Spitzmüller

STOP! If you don't know what BibTEX is, or have a reasonably good idea of how to use it (*e.g.* setting up your own bibliographic databases), *run*, do not walk, to your nearest copy of the 2nd edition of Lamport's *LATEX: A Document Preparation System*, particularly Appendix B. The rest of this discussion assumes you have created a correct bibliography file, that you have all relevant environment variables set correctly (esp. `BIBINPUTS`, `BSTINPUTS`, and `TEXINPUTS`), and that if sufficiently desperate, you could create and "TEX" a LATEX file with a BibTEX database.

For those who don't know what BibTEX is, it is a system for creating a large database of your most used journal references. For all future articles you write, you only need to include this standard database and reference the appropriate key to each reference. Even if you write only a few papers with handful of references each, it is well worth your time to examine BibTEX and decide whether it will be useful to you.

To use BibTEX with LYX, first read the *User Guide* where it describes how to insert citations. The basic mechanism for inserting BibTEX references is the same. Then, at the very end of your document, select Insert ▷ List / TOC ▷ BibTEX Reference. In the resulting dialog, fill out the dialog boxes as follows:

Database: enter the name of your `.bib` file *without* the `.bib` extension. For searching multiple `.bib` files, just enter them in the desired order, separated by commas.

Style: enter the name of your BibTEX style file *without* the `.bst` extension. The default style is `plain` (which should be included in your LATEX distribution, so you don't have to worry about creating it).

For each citation, assuming that the source is in the `.bib` file, just call Insert ▷ Citation Reference at the correct location in the text, and enter the appropriate reference key. Nothing else is required; when invoking View ▷ DVI, for example, you should see that BibTex and LaTeX are invoked as needed, including multiple invocations of LaTeX.

3.1.1 Alternative Citation Styles

Standard BibTeX uses numbers (e. g. "[12]") to refer to a cited work. However, in many scientific disciplines, other citation styles are in use. The most common one is the author-year style (e. g. "Knuth 1984a"). LyX supports two packages that provide this style, `natbib` and `jurabib`. Both packages have their own pros and cons, which cannot be listed in detail. If you only want to have simple author-year (or author-numerical) style or if you want to use one of the countless style files for natbib, than the established `natbib` package is probably your choice. If you need special features like short title references, ibidem etc., you might consider the fairly new `jurabib` package.

The handling of both packages in LyX is basically the same. Go to Document ▷ Settings and select the Bibliography pane (with the xforms frontend: the Extras tab). Then select Natbib or Jurabib. With both packages, you will get some extra features in the citation dialog and you can select the style of the reference ("Knuth 1984", "Knuth (1984)", "Knuth, 1984", "1984" etc.). Note that both packages need specifically designed style files (they both ship their own, while there are lots of additional style files and even an interactive style file builder[1] for `natbib` available).

3.1.2 Sectionated Bibliographies

Sometimes you might need to divide your bibliography into several sections. If you are, for instance, a historian, the possibility to separate sources and scientific works is most likely a "must have". Unfortunately, BibTeX itself does not allow you to do this. The good news is, though: With the help of some LaTeX packages, BibTeX can be extended to fit your historical needs.

As of version 1.4, LyX provides native support for one of these packages, Stefan Ulrich's `bibtopic`.[2] The advantage of this package (compared to other packages like `multibib`) is that you don't need to define new citation commands. Instead, you need to prepare different bibliographic databases which include the entries for the different sections of the bibliography. For example: If you want to divide your bibliography into the sections "Sources" and "Scientific works", you first need to create two bibliographic databases, e. g. `sources.bib` and `scientific.bib`.

In LyX, go to Document ▷ Settings and select the Bibliography pane (with the xforms frontend: the Extras tab). Check Sectionated bibliography. Now you can insert multiple BibTeX references (as described in section 3.1), one for

[1] See `ftp://ctan.tug.org/tex-archive/macros/latex/contrib/custom-bib/`
[2] Available from `ftp://ctan.tug.org/tex-archive/macros/latex/contrib/bibtopic/`

each section of your bibliography. Returning to our example: Insert a BibTEX reference for the database `sources.bib` and a second one for the database `scientific.bib`. You are free to use the same or different styles for each section. Additionally, you can chose if the bibliography section should contain "all cited references" of the specified database(s) (which is the default), "all uncited references" or even "all references". This might be useful if you would like to separate your bibliography into three sections: "Cited sources", "Uncited sources", and "Scientific works". The titles for the sections can be added as ordinary sections or subsections. Since `bibtopic` removes the bibliography title, you have manually re-add that, too (as a chapter* or section*, for instance).

3.1.3 Multiple Bibliographies

Multiple bibliographies, e. g. a bibliography for each section or chapter of the document, are not supported by BibTEX itself. But the `bibtopic` package, which is used for the creation of sectionated bibliographies in LYX (cf. section 3.1.2), provides an easy way to solve this task, if you are willing to use some LATEX-Code (ERT, cf. section 2.4).[3]

First, go to Document ▷ Settings and select the Bibliography pane (with the xforms frontend: the Extras tab). Check Sectionated bibliography. In the document, you have to enclose the sections, which shall contain their own bibliography (including the BibTEX reference itself), between `\begin{btUnit}` and `\end{btUnit}` (those commands have to be inserted as ERT). The bibliography will contain all references which have been cited in the current btUnit. N. B.: If you are using this approach, then *every* citation reference has to be inside some btUnit. Also, the btUnits cannot be nested.

3.2 Making an Index

A good index is one of the hardest things to make in a lengthy document, but LYX helps make things a bit simpler by interfacing to the `makeindex` program which is found in most recent LATEX distributions.[4] Inserting an index and marking words to include in it works much the same way as preparing a bibliography as mentioned in the last section.

First, go to the end of your file and select Insert ▷ List / TOC ▷ Index List. Then, for each word you would like to include in the index, go to the end of that word and click on Insert ▷ Index Entry. This will insert a tag showing the word as it will appear in the index. That's all there is to it; LYX will automatically call `makeindex` for you and create the index itself. The text in the dialog available from right-clicking on the index button accepts LATEX, so you'll need to be careful to avoid using any special characters. On the positive side, you can use the advanced options - have a look at the documentation which

[3] An alternative approach is to use the `chapterbib` or `bibunits` package, respectively.

[4] In the Outputs ▷ LaTeX section of the preferences dialog, however, you can customize the index command, if you prefer an alternative program like `xindy`.

comes with your LaTeX distribution to find out how to do things like "nested entries", etc.

Be careful not to put spaces between the word in the text and the index marker; apparently the wrong page number can be produced if this happens.

3.3 Multipart Documents

3.3.1 General Operation

When you are working on a large file with many sections, it is often convenient to break up the document into several files, or perhaps you have something where a table may change from time to time, but the preceding text does not. In these cases, you should seriously consider using multipart documents. For example, scientific papers often have five major sections: the introduction, observations, results, discussion, and conclusion. Each of these could be its own separate LyX file, with one "master" file which contains the title, authors, abstract, references, etc., plus the five included files. It is important to note that each of these files is a full LyX file which can be formatted and printed on its own, as well as included in a master file. Each of these files must have the same document class, however— don't attempt to mix book classes with article classes. You may also include LaTeX files; however, these files must not have their own preamble *(i.e.* everything up to and including the `\begin{document}` line as well as the `\end{document}` line must be deleted) or else errors will be generated when you try to make a DVI file.

LyX allows you to include files quite easily with Insert▷Child Document. When you click on this selection a small box is inserted into the file at the current cursor location. Clicking on the box raises a dialog which allows you to select the file to be included, and the method of its inclusion.

The file selection box should by now be obvious. The three inclusion methods are "include", "input", and "verbatim". The difference between "include" and "input" is really only meaningful to LaTeXperts, but the practical difference is that files which are "included" are typeset beginning on a new page, while files which are "inputted" are typeset starting on the current page. Perhaps the labeling in LyX will be changed someday to reflect this.

Generally, the master file is converted into a full LaTeX file before typesetting, while the included files are converted to LaTeX files which do not have all the preamble information. Checking the Don't typeset button prevents this conversion.

A "verbatim" included file allows you to include a file typeset exactly as it appears in the file, i.e. verbatim mode, with the characters set in a fixed-width typewriter font. Normally, spaces in this file are invisible, though two consecutive spaces are conserved, unlike LyX's normal treatment of spaces. However, setting the Mark spaces in output checkbox typesets a mark to unambiguously define the presence of a space.

3.3.2 Cross-References Between Files

It is possible to set up cross-references between the different files. First, open all the files in question: let's call them A and B in a two file example, where B is included in A. Let's say you insert a label in A, then want to reference it in B. Open the cross-reference dialog in whilst in document B, and you can select the "buffer" to use.

3.4 Algorithms

The package algorithm is needed by LyX to be able to output algorithm floats. These are useful in placing short algorithms across page breaks and support an index of algorithms too.

3.5 Subfigures

The package subfigure is used by LyX when you select "subfigure" in the graphics dialog and enter the subfigure caption. Several figures marked in this way can be packed into a single float with individual sub-captions.

3.6 Fancy Headers and Footers

The default page layout is rather plain; for an article document class, all you get is a centered page number at the bottom of the page. This document is the book class, so it appears to be a bit fancier, but to really put on a show, you need to set the document page style to "fancy", as mentioned in the User Guide. This section describes the LaTeX codes you need to insert in your LaTeX preamble or the text in order to get the desired effects.

The page header is divided into three fields, not surprisingly labeled "left", "center", and "right". The footer is also divided into these three fields. The LaTeX commands to set these fields in the simplest manner are `\lhead`, `\chead`, `\rhead`, `\lfoot`, etc. Suppose you wish to put your name in the upper left hand corner of each page. Simply insert the following command in the preamble:

```
\lhead{John Q. DocWriter}
```

You will now see your name in the upper left. If a field has a default entry that you would like to get rid of (often the page number appears in the central footer, simply include a command with a blank argument, e.g.

```
\cfoot{}
```

Let's get really fancy: lets put the section number with the word "Section" (e.g. Section 3) in the upper left, the page number (e.g. Page 4) in the upper right, your name in the lower left, and the date in the lower right. The following commands should now appear in the preamble:

```
\lhead{Section \thesection}
\chead{}
```

```
\rhead{Page \thepage}
\lfoot{John Q. DocWriter}
\cfoot{}
\rfoot{\today}
```

The codes `\thesection` and `\thepage` access LaTeX's section and page counters, and so print out the current section and page numbers. `\today` simply prints out today's date.

The thicknesses of the horizontal rules drawn beneath the header and above the footer can also be modified. If you don't want one of the headers, set its thickness to 0. The header rule has a default thickness of 0.4pt, the footer rule is 0pt. Use the commands, e.g. `\renewcommand{\headrulewidth}{0.4pt}` and `\renewcommand{\footrulewidth}{0.4pt}` to set the thicknesses.

You can switch the header/footer settings on and off for individual pages using commands like `\thispagestyle{empty}`, `\thispagestyle{plain}`, and `\thispagestyle{fancy}`. Simply insert them in the text on the page you want changed and mark them as TeX code. In fact, title pages are marked as plain by default, while following pages are marked fancy when using the global fancy setting.

There are more complex commands which will let you insert things in the upper left on odd numbered pages, etc., but I will refer you to the `fancyhdr` package documentation for more descriptions. For example, if you have a teTeX installation, look for `/usr/share/texmf/doc/latex/fancyhdr/fancyhdr.dvi`.

As a final example, it is possible to include an Encapsulated PostScript® file in the header or footer. Suppose you want to put a company logo in the upper lefthand corner. You might try something like

```
\lhead{\resizebox{1in}{!}{\includegraphics{logo.eps}}}
```

(you may need to preface this with `\usepackage{graphics}` if you don't include EPS files elsewhere in your document).

3.7 Minipages

LaTeX provides a mechanism to produce essentially a page within a page, called minipages. Within a minipage, all the usual rules of indentation, line wrapping, etc. apply. LyX also provides some of the minipage capability.

Minipages in LyX have their own collapsable box; insert one via Insert ▷ Minipage. Right-clicking on the box allows you to alter the minipage's width and alignment within the page. Warning: if the minipage is too long to fit on a page, it is truncated, not wrapped onto the next page.

If you place two minipages side-by-side, you can use Insert ▷ Special Character to insert a special instruction known in the LaTeX world as an `hfill` to put a maximum amount of space between them; it forces one minipage to the left edge, the other to the right edge. The examples below show the difference.

This is a minipage which does not use hfill. This is the second sentence of a minipage which does not use hfill.

This is a second minipage which does not use hfill. This is the second sentence of a second minipage which does not use hfill.

Here is some normal text to separate the two examples.

This is a minipage which does use hfill. This is the second sentence of a minipage which does use hfill.

This is a second minipage which does use hfill. This is the second sentence of a second minipage which does use hfill.

3.8 Wrapping Text Around Figures

A very frequently asked question is whether text can be made to "wrap" around figures so that a figure occupies some fraction of the column width and text fills the rest. If you have the LaTeX package `floatflt` installed (you can find out about it in the *LaTeX Configuration* manual) you can do this.

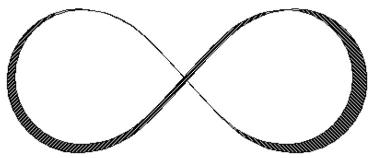

Figure 3.1: This is a wrapped figure, and this is the brilliant caption that describes it

At the right is a figure of a mobius strip—you should have already seen this in the *User's Guide*. To wrap the text like this insert a wrap box via Insert ▷ Floats ▷ Floatflt Figure.

Note: this package is very fragile! For example, having a figure too close to the bottom of the page will mess things up, as will having two figures close together. Use this package sparingly and do read the documentation that came with it (which will also tell you how to wrap text around tables).

3.9 Extra Table Options

While the standard table layout will suffice in 99% of all tables you generate, occasionally you will run into one which requires a bit of extra tweaking. The table dialog which appears on a right-click of a table allows these tweaks to be made. It will give you access to some extra column alignment parameters. A little bit of LaTeX background is useful here: when you set up a table in LaTeX, each column is given an alignment type. For example, you would give it "`l`", "`c`", or "`r`" for left-aligned, centered, and right-aligned columns respectively (which appear as the left/center/right radio buttons in LyX). A fourth type is "`p`", which will make a column of a specified width (the width box in LyX), and will wrap text within that box. A fifth type is "`|`" (vertical bar) which rather than making a column will make a vertical rule at that point; this manifests itself in LyX as the "borders" buttons. Finally, there is a type "`@`", which allows you to use whatever is enclosed in the accompanying braces as the column separator, including a null argument. The reasons for doing this may not be obvious, but they can be very powerful. They are best demonstrated by example.

3.9.1 Removing Extra Column Space

Here is a standard table:

Type	Example
Rock	Granite
Mineral	Quartz

Notice that the horizontal rule extends a bit past the text on both sides. If you wanted the line to end even with the text, we can put a null separator on the ends to get rid of the bit of extra space LaTeX adds by default. Here is the example:

Type	Example
Rock	Granite
Mineral	Quartz

In this case, the column specifier for the left column was set to "`@{} l`", while the right column was set to "`l @{}`", in order to put the null characters on the edges.

3.9.2 Changing the Column Separator Character

Now suppose you really wanted, for reasons that are completely opaque, to use $\sqrt{\pi}$ with some space around it for the column separator. Simply turn off the vertical border, then set the right column specifier to "`@{~$\sqrt{\pi}$~} l`". You could now make a table like this:

Type	$\sqrt{\pi}$	Example
Rock	$\sqrt{\pi}$	Granite
Mineral	$\sqrt{\pi}$	Quartz

3.9.3 Making a Decimal Point Aligned Column

Okay, that last example was very silly, but here is one that is not. Suppose you want to make a table that has a column which is aligned on a decimal point. A standard LaTeX trick to do this is to set the whole number part in a right-aligned column, use a decimal point for the column separator, then set the fractional part as a left-aligned column. A variation on this is to include the decimal point explicitly with the whole part, then use just a null separator in between. The latter variation is demonstrated here:

Expression	Value
π	3.1416
π^π	36.462
$(\pi^\pi)^\pi$	80663.
π^{π^π}	1.3402×10^{18}

Though it appears a bit funny in LyX, on paper it will produce what appears to be a 2-column table in which the right column is aligned on the decimal point and the header appears to be centered over it.

Perhaps it is best if I described just what I did: first, create a 3×3 table and remove all the borders. Then re-add a bottom border to the top row, and a right border to the first column. Type in the values for the first column and set its alignment to center. Type in the `3.`, `36.`, `80663.`, and `1.` and set that column's alignment to right. Type in the `1416`, `462`, and `3402`$\times10^{18}$ and set the extra column alignment to `@{} l`. Finally type in the word `Value` in the middle column, highlight it and the blank entry to its right, and check the Special Cell entry multicolumn. Easy, right?

3.9.4 A Better Decimal-Alignment Solution

An alternative way to have decimal alignment in tables is through the `dcolumn` package. Add the following to the LateX preamble:

```
\usepackage{dcolumn}
\newcolumntype{d}[1]{D{.}{.}{#1}}
```

To have a column decimally aligned, enter in the Special Column Alignment box of the Table dialog the following:

```
d{number of decimals of the data}
```

To create extra column space just increase the number of decimals in `d{}`. Setting the multicolumn attribute for a single cell makes it insensitive to the decimal alignment which comes in handy as well. A drawback of this method is that math mode is not allowed in a column with decimal alignment except if the multicolumn attribute is set.

This method offers the same flexibility as the `dcolumn` package. One could, for example, change the alignment separator, and have different alignment separators for different columns by defining multiple column types in the preamble. The syntax is as follows:

```
D{inputsep}{outputsep}{decimal places}
```

The interested reader is directed towards the `dcolumn` package documentation for more details.

3.10 Itemize Bullet Selection

by Allan Rae

3.10.1 Introduction

LyX provides 216 bullet shapes that can be accessed from a simple dialog. Using this dialog you can easily specify what bullet shape to use at each level of an itemized list. These settings are document-wide so you won't be able to specify different sets of bullets for different paragraphs[5].

3.10.2 How it looks

Open the dialog by selecting the Document ▷ Settings menu item and then select the Bullets tab.

The dialog provides you with a table of bullet shapes. A column of buttons on the left of the table provides access to the six different panels of bullet shapes. The row of buttons across the top is used to select which bullet depth you are changing. A text entry under the table shows the currently selected bullet shape's LaTeX equivalent and this can be edited if desired. If you do modify the text you will also need to specify any needed packages in the LaTeX preamble.

The six panels are divided up by the packages they require. The following table shows the mappings from button name to LaTeX packages.

Button	Packages Required
Standard	base LaTeX
Maths	`amssymb.sty`
Ding1	`pifont.sty`
Ding2	`pifont.sty`
Ding3	`pifont.sty`
Ding4	`pifont.sty`

LyX doesn't stop you using bullets from packages you don't have. If you get errors from LaTeX when you try to view or print the file then its likely you are missing a package. LyX doesn't restrict your use since you may be editing locally and exporting elsewhere.

3.10.3 How to use it

Select which bullet depth you want to change then select the bullet shape and size. Any changes will not be visible in LyX, but are visible when viewing the document using xdvi or ghostview.

You can reset a bullet shape to the default simply by clicking your right mouse button on the appropriate bullet depth button.

[5]Well, actually you can but you'll have to do it by hand.

If you *really* want to have multiple sets of paragraphs with different sets of bullets in each then you're going to have to get your hands dirty. The itemize bullet selection dialog can help though because it provides you with the LaTeX code for a wide range of bullet shapes. To make your own custom paragraphs you have the following options:

- $\sharp$ Use the LaTeX command `\renewcommand{}{}` to specify a new bullet shape for a given depth. You'll also need to save the current bullet shape so you can restore it again afterwards. In this itemized list the following LaTeX code was used to change the bullet used for the first depth.
  ```
  \let\savelabelitemi=\labelitemi
  \renewcommand\labelitemi[0]{\small\(\sharp\)}
  ```
  Note that the itemize depth is specified in Roman numerals as part of the `\labelitem` command.

- $\star$ Specify each individual entry by starting each item with the bullet shape enclosed in square brackets and set as TeX. For example, this item was started with `[\(\star\)]`.

You'll also need to revert the labelitem back to its previous setting for the global bullet shape settings to remain in effect. The way used here was:

```
\renewcommand\labelitemi[0]{\savelabelitemi}
```

Chapter 4

Special Document Classes

4.1 AMS LaTeX

by David Johnson

The AMS LaTeX layouts are set up to conform to suggested styles for mathematical papers to be submitted to American Mathematical Society publications. The layouts are not tailored to a specific journal, but easily can be. You should refer to the AMS documentation for specific instructions for each journal (usually it will entail only changing a single line in the TeX output). That documentation is available on the Web at `http://www.ams.org` or by ftp at `ftp://ftp.ams.org/pub/tex/amslatex/`.These layouts are appropriate, and useful, for any mathematical writing. There are currently 4 distinct AMS LaTeX layouts:

1. amsart: The standard AMS-article format. All results and similar statements are numbered as $(n.m)$, where the first number refers to the section, and the second refers to the total number of results (Theorems, Corollaries, Propositions, Definitions and Remarks, etc.) in that section. There are also many (but not all) environments available unnumbered, which is occasionally needed. Unnumbered environments indicated by an asterisk at the end.

2. amsart-seq: Here, numbering for each type of statement is in its own sequence, with no reference to the section number. There are also many (but not all) environments available unnumbered, which is occasionally needed. Unnumbered environments indicated by an asterisk at the end.

3. amsart-plain: This one is even more terse, since all the environments are unnumbered.

4. amsbook: the standard AMS book (really, monograph) format. Numbering is similar to the amsart layout, except that all numbering is by

(*n.m.p*), where the first number refers to the chapter, the second to the section, and the third is the number of the results (Theorems, Corollaries, Propositions, Definitions and Remarks, etc.) in that section. There are also many (but not all) environments available unnumbered, which is occasionally needed. Unnumbered environments indicated by an asterisk at the end.

Any AMS LyX file can be converted to either of the numbering schemes by simply changing the document class in the Document▷Settings dialog.

4.1.1 What these layouts provide

There is a long list of included environments provided by these layouts. Most mathematical papers or books will set as special statements most of these environments, in AMS-LaTeX there is an opportunity to define an unlimited variety of such declarations. However, the AMS recommends the environments that are available in LyX. The list of environments (not counting the standard environments such as sections, bibliography, title, author, date), is:

Theorem This is typically used for the statements of major results. The word "Theorem" appears in bold type, along with an automatically-determined number (an unnumbered version, Theorem*, is also available). The text is italicized.

Corollary This is used for statements which follow fairly directly from previous statements. Again, these can be major results. Unnumbered version Corollary* is available.

Lemma These are smaller results needed to prove other statements.

Proposition These are less major results which (hopefully) add to the general theory being discussed.

Conjecture These are statements provided without justification, which the author does not know how to prove, but which seem to be true (to the author, at least).

Criterion A required condition.

Algorithm A general procedure to be used.

Axiom This is a property or statement taken as true within the system being discussed.

Definition Guess what this is for. The font, both on-screen and in the output, is different for this environment than for the previous ones. The heading ("definition") is still set in boldface, along with the number, if any, but the rest is set upright.

Example Typeset similarly to Definition.

Condition

Problem

Exercise

Remark This environment is also a new type of theorem. This is set with the word Remark in italics, and the rest upright.

Note Set similarly to the Remark environment.

Notation

Claim

Summary

Acknowledgement

Case Generally, these are used to break up long arguments, using specific instances of some condition. The numbering scheme for cases is on its own, not together with other numbered statements.

Conclusion

Fact

Proof The word "*Proof*" is set in italics, but the rest is set upright. At the end of this environment (other environments can be nested within this one, of course) a QED symbol (usually a square, but it can vary with different styles) is placed.

Address This should be the author's permanent address.

Current Address This should be the author's temporary address at the time of submission, if different from the Address.

Email Author's e-mail address

URL Author's Web address, if desired.

Keywords Key words or phrases used to identify specific topics discussed in the paper.

Subjectclass These refer to the AMS Subject Classifications, published and described in *Mathematical Reviews.* These are also available online at the AMS cites listed above.

Thanks

Dedicatory

Translator

In addition, these environments automatically provide the AMS LaTeX and AMS fonts packages. They need to be available on your system in order to use these environments.

4.2 Dinbrief

The document class dinbrief can be used to type letters according to German conventions. A template file is included in `.../lyx/share/templates` for you to use as a starting point.

4.3 Paper

The document class paper provides an alternative to the standard article class. It provides similar functionality, but you might prefer this layout with sans serif sections, headings, and more.

4.4 A&A Paper

by Peter Sütterlin

4.4.1 Introduction

This section describes how LyX can be used to write articles for submission to the scientific journal *Astronomy and Astrophysics* (www.edpsciences.fr/aa/ `http://www.edpsciences.fr/aa/`) using Version 5.01 of the document class `aa.cls`. This package can be downloaded from the ftp site

`ftp://ftp.edpsciences.org/pub/aa/readme.html`

A manual comes together with that package, and this text is not meant to replace the original manual but merely a short guide how to realize the correct form of your paper.

Please note that the publisher of the journal was changed from Springer to EDP Sciences starting January 1, 2001. That change implicated also some slight changes of the style files, namely the removal of the thesaurus command. The LyX class aa supports the newest version of these style files, V 5.01. If you have an older version installed, please upgrade. For compatibility, the old (version 4) layout has been kept as article (A&A V4). Please refer to the comments in `LyXDir/layouts/aapaper.layout`.

4.4.2 Getting started

It is recommended you start from the example template distributed with LyX. If you are not using a template, note the following settings:

- Select article (A&A) in the Document▷Settings dialog (OK, that one was obvious).
- Don't change the option Page style: Leave it set to default. The whole layout is done by the macros, you shouldn't change anything.

4.4.3 The header block

First thing to enter is the header information. It consists of seven entries, of which some are optional. They are

- Title: [required]
- Subtitle: [optional]
- Author: [required]
- Address: [required]
- Offprints: [optional] if more than one author: whom to contact for offprint requests.
- Mail: [optional] mail address for contacts.
- Date: [required]. Suggested format is `Received: <date>; Accepted <date>`

There is no need to issue the `\maketitle` command, this is done automatically by LyX when the header is finished. Although the order of the single header entries doesn't matter it is advised to keep the above sequence, just to get the best optics and meets the layout of the real document.

If you want to place footnotes in the header block, e.g. to state your present address, just use the standard footnote via Insert ▷ Footnote. LyX will automagically use the term `\thanks{}` in that case.

In addition to these topics, the macros use three additional LaTeX commands that have no counterpart in LyX:

- `\and` to separate different names for more than one author and institute, respectively.
- `\inst{<nr>}`to mark corresponding author/institute pairs. The institutes are numbered sequentially as they appear in the Address field, so you have to put a marker to each author.
- `\email{address}` to supply an email address for fast contact.

In all cases, the appropriate command has to be entered in LyX an marked as LaTeX code. See the examples.

4.4.4 The abstract

The abstract should immediately follow the header block. With version 5 the abstract environment was changed to a command, and there is now a resctriction to only one paragraph. In addition, it should contain an entry with the keywords. This is not yet implemented for LyX, therefore you have to enter the LaTeX command `\keywords{}` by hand and mark it as LaTeX code. Refer to the example paper.

4.4.5 Supported environments

The A&A paper layout supports the following environments for structuring your text:

- Standard
- Section
- Subsection
- Subsubsection
- Itemize
- Enumerate
- Description
- Caption
- Abstract
- Acknowledgment
- Bibliography
- LaTeX

4.4.6 Commands not supported by LyX

Some commands are not yet supported by the paper (A&A) layout for LyX. Some have already been mentioned. For the sake of completeness, they are listed all together here:

- `\and`
- `\email`
- `\appendix`
- `\authorrunning`
- `\inst{}`
- `\keywords{}`
- `\object{}`
- `\titlerunning{}`

If you want to use any of these commands, you have to enter them yourself. **Do not forget to mark them as LaTeX code!**

4.4.7 Figure and Table Floats

LYX provides support for the necessary float environments figure, figure*, table and table*, therefore we won't tell much about it here. Refer to the *User's Guide*. Just remember that tables should be left-aligned. For that, select the table and change the alignment in Edit ▷ Paragraph Settings.

There is only one special thing: the figures with caption besides the figure. To create such a figure, you have to do the following:

1. Create a wide figure float: Insert ▷ Float ▷ Figure, then right click in the figure and select Span columns.
2. Enter your caption text.
3. Press Return to move the cursor above the caption.
4. Insert your figure
5. Position the cursor behind the figure and insert a horizontal fill: Insert ▷ Special Character ▷ Horizontal Fill.
6. Switch to LATEX mode: M-c t.
7. Enter `\parbox[b]{55mm}{`. **Do not close the brace!**
8. Position the cursor behind the caption text, switch to LATEX mode and insert the closing brace: M-c t }.

Also, refer to the figures in the example paper.

4.4.8 Referee layout

For submission, the paper has to be formated in a special double-spacing layout. For this purpose, you have to give the option `referee` to the documentclass. This must be done using the extra class options field in the Document ▷ Settings dialog. Just enter the string `referee` there.

4.4.9 The example paper

The Examples directory contains an example paper written with LYX. It is the example paper from the original macro package, translated to LYX. Use it for inspiration, and compare the original LATEX code with LYX way of writing.

4.5 AASTEX

by Mike Ressler

4.5.1 Introduction

AASTEX is a set of macros produced by the American Astronomical Society to facilitate electronic manuscript submission to the three journals they publish: the Astrophysical Journal (including the Letters and Supplement), the Astronomical Journal, and the Publications of the Astronomical Society of the Pacific. LYX has proven to be an excellent tool for generating these documents, especially given its equation, citation, and figure handling capabilities. LYX requires version 5.0 (or higher) of these macros; preferably 5.2, which is the version described here, or higher. Versions prior to 5.0 are intended for use with LATEX2.09 and are fundamentally incompatible with LYX. The AASTEX package may be downloaded from the AASTEX Web site

`http://www.journals.uchicago.edu/AAS/AASTeX`

A complete user guide is contained in that package and you should familiarize yourself with it thoroughly before embarking on writing a paper in LYX. LYX will not reduce the need to figure out all the AASTEX commands, it will only reduce the drudgery of typing everything in. It is your responsibility to ensure that the final exported LATEX document conforms completely to the requirements of the journal to which you are submitting your paper.

4.5.2 Starting a New Paper

I strongly suggest that you start with the AASTEX template file. Click on File ▷ New from Template, enter the new file name, then choose the `aastex.lyx` template. This will show the most common fields found in a manuscript. Simply overwrite the existing text (including the brackets, <>) with the correct information. Many of the AASTEX commands and environments can be implemented directly in LYX, but some cannot: most noticeably `\altaffilmark` and `\altaffiltext`, which should stick out like a sore thumb if you actually just opened the template file. For commands such as these, the LATEX code must be entered directly and marked as such. Such commands are referred to as ERT, or Evil Red Text. I tried to minimize the amount of ERT needed in an AASTEX document, but there is still a bit more required than any of us would like.

4.5.3 Finishing Your Paper

When the paper is finished to your satisfaction and previews/prints correctly, there are a few "postprocessing" actions which need to be done before you submit it to the journals.

1. Export your paper as a LATEX file (File ▷ Export ▷ LaTEX).
2. Edit the resulting `.tex` file with your favorite text editor
   (a) remove the comment lines before the `\documentclass` command

(b) remove the `\usepackage...{fontenc}` line if it appears (usually just after `\documentclass}`; also remove the `\secnumdepth` line if it appears.

(c) remove everything between (and including) the `\makeatletter` and `\makeatother` commands, except for any commands you specifically put into the LATEX preamble (which should appear immediately after the "User specified LATEX commands" comment in the `.tex` file).

3. Run the resulting file through LATEX to make sure it still processes correctly.

4. Reread the journal requirements to make sure your filenames and formats are correct.

5. Submit it.

4.5.4 Comments On Specific Commands

I will not describe the detailed usage of the individual AASTEX commands: the AASTEX User Guide (`aasguide.tex`) gives a good description of each. Thus it's probably easiest for me to go down the list as found in the guide and offer comments where necessary. So let's begin ...

4.5.4.1 Things that work as expected

Because they work as you might expect, I simply list them and the section they are found in: `\documentclass` (2.1.1), `\begin{document}` (2.2), `\title` (2.3), `\author` (2.3), `\affil` (2.3), `\abstract` (2.4), `\keywords` (2.5), `\section` (2.7), `\subsection` (2.7), `\subsubsection` (2.7), `\paragraph` (2.7), `\facility` (2.10), `\begin{displaymath}` (2.12), `\begin{equation}` (2.12), `\begin{eqnarray}` (2.12), `\begin{mathletters}` (2.12), `\begin{thebibliography}` (2.13.1), `\bibitem` (2.13.2), all the cite commands and their variations (2.13.2), the generic graphicx figure commands (2.14.1), `\begin{table}` (2.15.4), `\begin{tabular}` (2.15.4), `\caption` (2.15.4), `\label` (2.15.4, amongst other places), `\tablerefs` (2.15.5), `\tablecomments` (2.15.5), `\url` (2.17.4), `\end{document}` (2.18).

The following style options also work correctly: `longabstract` (2.4), `preprint` (3.2.1), `preprint2` (3.2.2), `eqsecnum` (3.3), `flushrt` (3.4). Simply put them in the Options box in Layout ▷ Document.

4.5.4.2 Things that work, but require more comment

The following items work, but require a little more discussion:

- These items are reserved for use by the journal editors, but you can put them into the LATEX preamble if you feel compelled to do so: `\received`, `\revised`, `\accepted`, `\ccc`, `\cpright` (all from 2.1.3)

- These items may be placed in the LaTeX preamble, and are included as blanks in the template file: `\slugcomment` (2.1.4), `\shorttitle` (2.1.5), `\shortauthors` (2.1.5)

- `\email` (2.3) – can only be used "standalone", not in the middle of a paragraph. Use ERT if you need to embed it.

- `\and` (2.3) – will have extra {} after it. This should not cause an error.

- `\notetoeditor` (2.6) – can only be used "standalone", not in the middle of a paragraph. Use ERT if you need to embed it.

- `\placetable` (2.8) – can't insert a cross-reference tag, you must type the tag name by hand

- `\placefigure` (2.8) – same as for `\placetable`

- `\acknowledgements` (2.9) – will have extra {} after it. This should not cause an error.

- `\appendix` (2.11) – will have extra {} after it. This should not cause an error.

- `\figcaption` (2.14.2) – you can insert an optional filename argument by placing the cursor at the beginning of the text and selecting Insert ▷ Short Title. "Short Title" inserts an optional argument of the type needed by `\figcaption`. Hopefully it will be renamed someday.

- `\objectname` (2.17.1) – same as `\figcaption` for the catalog ID optional parameter

- `\dataset` (2.17.1) – same as `\figcaption` for the catalog ID optional parameter

4.5.4.3 Things not implemented, use ERT

`\altaffilmark` (2.3), `\altaffiltext` (2.3), `\eqnum` (2.12), `\setcounter{equation}` (2.12), Journal name abbreviations (2.13.4), `\figurenum` (2.14.1), `\epsscale` (2.14.1), `\plotone` (2.14.1), `\plottwo` (2.14.1), `\tablenum` (2.15.4), `\tableline` (2.15.4, insert it as the first element in the lefthand cell after where you want it. Don't use any of LyX's rules in the table), `\tablenotemark` (2.15.5), `\tablenotetext` (2.15.5), much of Misc (2.17, except `\objectname`, `\dataset`, `\url`, and `\email`; see above), `\singlespace` (3.1), `\doublespace` (3.1), `\onecolumn` (3.2), `\twocolumn` (3.2)

4.5.4.4 Things that cannot be implemented

... at least in any meaningful sort of way, so I suggest ignoring them. They are the references environment (2.13.3), and the deluxetable environment (2.15). If you really, really need to use deluxetable, I suggest editing it in a separate file with a text editor, then using Insert ▷ Child Document to include it in your LyX document. See the `aas_sample.lyx` file to see an example of this.

4.5.5 FAQs, Tips, Tricks, and Other Ruminations

4.5.5.1 Getting LyX and AASTEX to cooperate

It can be a bit tricky to get LyX to recognize a new layout and document class. When all else fails, do this:

1. Make certain that LATEX can find AASTEX. Copy sample.tex (and perhaps table.tex) from the AASTEX distribution into a directory completely unrelated to LATEX or AASTEX and run LATEX on `sample.tex`.
2. Make certain that `aastex.layout` appears in `/usr/.../share/lyx/layouts` or `~/.lyx/layouts`.
3. Rerun Tools ▷ Reconfigure in LyX, then restart LyX.
4. Open a regular new file, not from a template. Does AASTEX appear in the class list in Document ▷ Settings?

If you get a warning from an existing AASTEX document about not being able to find the AASTEX layout or a message about "You should not mix title layouts with normal ones", things haven't been installed correctly.

4.5.5.2 LATEX error processing a table

LyX, by default, attempts to center the table caption/title. This seems to produce a bad interaction in AASTEX so you should click somewhere in the caption/title, then select Edit ▷ Paragraph Settings, then set the Alignment to Block. This took care of it for me.

4.5.5.3 References

A couple of things: 1) I have noticed some funny spacing in the reference entries in the text. When you enter the bibliography item data, make sure their is *no* space between the last author and the parenthesis setting off the year; *e.g.* type `Ressler(1992)`, not `Ressler (1992)`. 2) Entering the references at all is not obvious. The easiest thing is to start typing your first reference at the end of the document, then mark it as type References. That will put a small gray box in front of what you just typed. Click on the box to fill in the rest of the information. For new references, go to the end of an existing reference and press return. That will create a new line with its own box, etc.

4.5.5.4 Including EPS files

Even though AASTEX provides its own figure commands (`\plotone`, for example), I much prefer LaTeX's standard figure commands (with the default graphicx). You can insert the `\plotone`, etc. commands as ERT into a Figure Float box if you desire, but I never have much luck getting the layout right. With the standard graphics, LyX will insert a `\usepackage{graphicx}` command into the LaTeX preamble and handle the figures in the standard LaTeX 2$_\varepsilon$ way, interspersing the figures in the text. I believe ApJ accepts figures exactly this way now; AJ might still use the "stack everything at the end" technique.

4.5.5.5 Things I could have done, but didn't

There are a few "pretty" things I could have implemented, but chose not to. For instance, I saw no point in double-spacing the text in the LyX window, even though it is double-spaced in the paper manuscript. Also, I chose not to make separate layouts for the preprint and preprint2 styles. Since I assume you will spend most of your time in the plain manuscript mode anyway, I decided not to chew up more disk space with this.

4.5.6 Final Caveat

Your mileage may vary. I've now had papers published by both ApJ and AJ that have had 98% of the effort done in LyX; the last 2% was the LaTeX post-processing and a few cleanups. I have had no trouble with the submission process, and I'm sure the journals were never aware that there might be a difference. So, go forth and publish!

4.6 ijmpd

by Panayotis Papasotiriou

4.6.1 Overview

The ijmpd package is a set of macros that facilitates electronic manuscript submission to the *International Journal of Modern Physics D* published by World Scientific. The name of the document class is `ws-ijmpd.cls`. This file, together with instructions for the authors, can be downloaded from the site `http://www.worldscinet.com/ijmpd/mkt/guidelines.shtml`. The ijmpd package is a modified version of the standard "article" package. Most of its features are supported by LyX. I have recently used LyX successfully to write an article submitted to the *International Journal of Modern Physics D.*

4.6.2 Writing a paper

As usual, the easiest way to write a paper is to start with a template. Click on File ▷ New from Template, then choose the `ijmpd.lyx` template. This will give an (almost) empty document that includes the most common fields found in a manuscript. Simply overwrite the existing text (including the brackets, <>) with the correct information. You should keep in mind the following remarks.

1. LyX won't let you change the font size and the page style of the document, because the ijmpd package does not allow such modifications.

2. The ijmpd package requires that the language of the document should not be changed. Before previewing your paper, be sure that the babel package is not used. To do this, click on Tools ▷ Preferences, deselect the Use babel checkbox in the language settings, and click on Apply (or Save, if you wish to make this change permanent).

3. Two new environments, named "Theorem" and "Proof" are available (their use is obvious).

4. Appendices may be added to the paper. LyX offers a special environment, called "Appendix" which marks the beginning of the appendix. An appendix can contain normal sections, subsections, or subsubsections.

5. The ijmpd package implements table captions quite differently than LyX does. As a result, a table created by LyX is printed correctly, but its caption is ignored. If you need table captions, you should implement the whole table float in a `.tex` file, then include this file to the LyX document (Insert ▷ Child Document). Details on how to create an ijmpd table float can be found in the file `ws-ijmpd.tex`, which is included in the ijmpd package.

4.6.3 Preparing a paper for submission

Before you submit your paper you must export the LyX document as a LaTeX file (File ▷ Export ▷ LateX), then make the following changes to the resulting `.tex` file.

1. Remove the comment lines before the `\documentclass` command.

2. Remove everything between (and including) the `\makeatletter` and `\makeatother` commands, except for any commands you specifically put into the LaTeX preamble.

The modified `.tex` file should be saved and processed through LaTeX as many times as necessary. You may also want to check the resulting `.dvi` document.

4.6.4 Use of ERT

The use of ERT is optional, and is reduced to three commands, which affect the look of the page. If you started writing your paper by using the `ijmpd.lyx` template, the ERT needed is already in its place; you usually don't need to delete it. You may only change the first ERT to specify the information printed to the top of odd and even pages (authors' names and short paper's title, respectively). This ERT must have the form `\markboth{Authors' Names}{Short Paper's Title}`.

4.7 Kluwer

by Panayotis Papasotiriou

4.7.1 Overview

The Kluwer package is a set of macros produced by Kluwer Academic Publishers that facilitates electronic manuscript submission to the journals they publish. Most known of them (at least in my domain of interest) are *Astrophysics and Space Science* and *Solar Physics*, but there are many others (see a complete list at `http://www.wkap.nl/jrnllist.htm/JRNLHOME`). The Kluwer package may be downloaded from the site `http://www.wkap.nl/kaphtml.htm/STYLEFILES`. A complete user guide is contained in that package (but it can also be downloaded separately).

LyX supports many features of the package but not everything. However, the ERT needed is reduced to some "peculiar" commands of the package (see 4.7.4). I have recently used LyX to write an article submitted to the *Astrophysics and Space Science* without any problem.

4.7.2 Writing a paper

The easiest way to write a paper is to start with the Kluwer template file. Click on File ▷ New from Template, then choose the `kluwer.lyx` template. This will give an (almost) empty document that includes the most common fields found in a manuscript and a short description of their use. As in most templates, simply overwrite the existing text (including the brackets, <>) with the correct information.

4.7.3 Preparing a paper for submission

As in the AASTeX package, before you submit your paper to a journal you must "postprocess" it as follows.

1. Export your paper as a LaTeX file. To do this, click on File ▷ Export ▷ LateX.
2. Edit the resulting `.tex` file with a text editor and make the following changes

   (a) remove the comment lines before the `\documentclass` command,
   (b) remove everything between (and including) the `\makeatletter` and `\makeatother` commands, except for any commands you specifically put into the LaTeX preamble.

   Save the resulting `.tex` file.

3. Run the `.tex` file through LaTeX as many times as necessary (usually up to three).
4. View the resulting `.dvi` document using, e.g., `xdvi`, and check if everything is ok (it should, if you didn't make any mistake).

4.7.4 "Peculiarities" of the Kluwer package

The Kluwer package has the following "peculiarities".

1. It is possible to write multiple articles in the same LaTeX file[1]. Each article must be included in the environment "article". Unfortunately, this environment cannot be omitted, even if you write just one article. Therefore, each article starts with the command `\begin{article}` and, obviously, ends with the command `\end{article}`. Although this can be implemented in LyX, I didn't included it, since it looks ugly and can confuse the novice user. Therefore, you need to enter them directly and mark them as LaTeX code (the well-known "ERT").
2. Information given at the beginning of the article (i.e., title, subtitle, author, institution, running title, running author, abstract and keywords) must be included in an environment called "opening". This is not implemented in LyX, so you must enter title, subtitle etc. between two ERT lines (`\begin{opening}` and `\end{opening}`).
3. According to the user manual, the label of each bibliography item must be written as `\protect\citeauthoryear{`*author(s)*`}{`*year*`}`.

The `kluwer.lyx` template takes care of all these "peculiarities". If you start a new paper using this template you don't need to do anything special. Just

1. don't delete the ERT included in the template, and
2. copy the example bibliography item included in the template and modify it as necessary to enter new bibliography items.

4.8 Koma-Script

by Bernd Rellermeyer

[1] I can't imagine any good reason to do this.

4.8.1 Overview

The LyX document classes *article (koma-script)*, *report (koma-script)*, *book (koma-script)*, and *letter (koma-script)* correspond to the LaTeX document classes `scrartcl.cls`, `scrreprt.cls`, `scrbook.cls`, and `scrlettr.cls`, resp. of the Koma-Script family. They are replacements for the standard document classes `article.cls`, `report.cls`, `book.cls` and `letter.cls`, resp., and fit better to European typography conventions in a number of points.

- Standard character size is 11pt in *article (koma-script)*, *report (koma-script)*, and *book (koma-script)*, and 12pt in *letter (koma-script)*.
- Headings, labels of the description environment, and a number of elements of the *letter (koma-script)* document class are set in a bold sans serif font.[2] The numbering of chapter headings is made in the same way as the numbering of section headings, that is without the extra line "Chapter…". In addition, the appearance of the headings can be modified by using a number of options (in LyX to be entered in the field Extra Options of the dialog Layout ▷ Document). A detailed German description of these options can be found in the Koma-Script documentation *scrguide*.
- The main means in the Koma-Script document classes to design the type area are the options BCOR and DIV (in LyX to be entered in the extra class options field in the dialog Document ▷ Settings). They make a clearer modification of page margins possible as do the options of the dialog Document ▷ Settings. A detailed German description of these and other type area options can be found in the Koma-Script documentation *scrguide*.
- The LaTeX document classes of the Koma-Script family define a number of additional commands. Those part of it which makes sense in LyX is implemented in corresponding paragraph types.

A detailed German description of the LaTeX document classes of the Koma-Script family can be found in the Koma-Script documentation *scrguide*.[3] The following sections describe only those aspects, which are relevant in LyX.

4.8.2 article (koma-script), report (koma-script), and book (koma-script)

The document classes *article (koma-script)*, *report (koma-script)*, and *book (koma-script)* are implemented in the layout files `scrartcl.layout`, `scrreprt.layout`,

[2]There is a big difference between the bold sans serif old cm fonts and new ec fonts, especially in the appearance of headings. In comparison, the ec bold sans serif fonts look a bit thin. Here the LaTeX package `cmsd.sty by` Walter Schmidt helps to produce the "usual" appearance when using the ec fonts.

[3]There is an English translation *screnggu*, but it is not a complete one.

and `scrbook.layout`, resp. They contain all the paragraph types of the corresponding standard document classes *article*, *report*, and *book*, resp., partly modified, with the exception of the LyX specific List-type, which is replaced by the new Labeling-type having the same functionality. Beside the Labeling-Type there is a number of new paragraph types added. They are *not* part of *letter (koma-script)*.

- Addpart, Addchap, Addsec: are equivalents to Part*, Chapter* and Section*, resp., additionally inserting an entry in the table of contents. Addpart and Addchap are not contained in *article (koma-script)*.
- Addchap*, Addsec*: behave exactly as Addchap and Addsec, resp., additionally clearing running heads. Addchap* is not contained in *article (koma-script)*.[4]
- Minisec: generates a heading directly above the following paragraph in the standard character size without affecting the structure of the document.
- Captionabove and Captionbelow are special captions which respect the different space settings needed for captions placed above or below an element (if you follow strict typographic rules, you might want to place table captions always above the table). You can also use the class option `tablecaptionsabove`, which will switch caption to captionabove for tables and captionbelow for figures. You need at least Koma-Script version 2.8q to use this.
- Dictum: can be used to set a bonmot, e. g. at the beginning of a chapter. If you use the optional argument (Insert ▷ Short Title), you can insert the dictum's author there. Dictum and author are separated by a line. You need at least Koma-Script version 2.8q to use this. Dictum is not contained in *article (koma-script)*.

The following types, together with the standard types Title, Author, and Date, form the title area of the document. They must be entered ahead of the first "ordinary" paragraph.[5] When such a type is used more than once, the latter usage overwrites the former one, that means, for every type only the latest usage is valid. The order of the different types however has, like Title, Author, and Date, no effect on the appearance of the produced document.

- Subject: produces a centered paragraph above the ordinary title (Title, Author, Date) for the subject of the document.
- Publishers: produces a centered paragraph below the ordinary title (Title, Author, Date) for the publishers' name.

[4] There is also an `\addpart*` command in *book (koma-script)* and in *report (koma-script)*, but since this is identical to Part*, is has not been implemented in LyX.

[5] The corresponding LaTeX commands must appear before the `\maketitle` command.

- Dedication: in *report (koma-script)* and *book (koma-script)* produces a centered paragraph on its own page behind the title page, or in *article (koma-script)* produces a centered paragraph below the ordinary title (Title, Author, Date, Publishers) for a dedication.

- Titlehead: produces a left aligned paragraph above the ordinary title (Title, Author, Date, Subject) for a document‘s head.

- Uppertitleback: produces in a double-sided print in *report (koma-script)* and *book (koma-script)* a left-aligned paragraph at the top of the title page‘s back or has no effect in a single-sided print or in *article (koma-script)*.

- Lowertitleback: produces in a double-sided print in *report (koma-script)* and *book (koma-script)* a left-aligned paragraph at the bottom of the title page‘s back or has no effect in a single-sided print or in *article (koma-script)*.

- Extratitle: produces a special "dirty" page ahead of the actual document containing a paragraph without special formatting.

The layout files for the document classes *article (koma-script)*, *report (koma-script)*, and *book (koma-script)* do include the file `scrmacros.inc`. This is thought of as a place to define your own types. Copy `scrmacros.inc` in your personal layout directory and edit the file!

4.8.3 letter (koma-script)

The document class *letter (koma-script)* is implemented in the layout file `scrlettr.layout`. It contains all the paragraph types of the corresponding standard document class *letter*, partly modified, with the exception of the LyX specific types LyX-Code and Comment and the List type, which is replaced by the new Labeling type. In addition, it contains, in contrast to the standard document class, the standard types LaTeX, Quotation, Quote, and Verse. Furthermore, there are a number of new letter specific types.

The appearance of the letter produced by this document class can be controlled by a number of LaTeX commands, which you can put in the LaTeX preamble.[6] A detailed German description of such LaTeX commands can be found in the Koma-Script documentation *scrguide*. With it, the letter's author can produce his personal letter layout.

[6]For example, the standard appearance of the letter‘s heading, consisting of name and address, is quite self-willed. An "ordinary" heading is produced by the following LaTeX commands in the preamble:

```
\firsthead{\parbox[b]{\textwidth}
  {\ignorespaces \fromname\\ \ignorespaces \fromaddress}}
\nexthead{\parbox[b]{\textwidth}
  {\ignorespaces \fromname \hfill \ignorespaces \pagename\ \thepage}}
```

The types Letter and Opening define the beginning of the letter and must be used in every letter. To emphasize them in the LyX document class, they are marked with the letter *L* or *O*, resp. in the left margin. It is possible to write any number of letters in one file. An Opening type produces a new letter using the same addressee and a Letter type produces a new addressee. The types Closing, PS, CC, and Encl are ordinary paragraph types and can also be used several times in one and the same letter.

- Letter: produces a paragraph for the addressee and implicitly defines the beginning of the letter.
- Opening: produces a paragraph for the form of address and implicitly produces a new letter.
- Closing: produces a paragraph for a close.
- PS: produces a paragraph for a postscript.
- CC: produces a paragraph for a distribution list.
- Encl: produces a paragraph for enclosures.

The types Name, Signature, Address, Telephone, Place, Backaddress, Specialmail, Location, Title, and Subject are input types provided with a label to enter information, which will be processed by the document class.[7] The types must be used ahead of the corresponding Opening type.

An implementation of these types in a WYSIWYG fashion does not seem to make sense, because the real appearance of the produced letter does not only depend on the usage of the particular type, but also on other factors. For example, a signature entered in the Signature type will in the standard behavior appear in the produced letter only, when in the same letter also a Closing type is used. The entered value of the Telephone type will in the standard behavior not appear in the produced letter at all. The possibility to design the letter‘s heading freely is already indicated in a footnote above.

The input types can also be used as empty paragraphs. This makes sense e. g. for the Signature type. If the Signature type is not used at all, in the standard behavior the value of the Name type is used as signature, whereas if an empty Signature type is used, no signature value is defined.

By using the input types it is possible to write a letter template, containing filled input types with your personal dates (name, address, etc.) and empty input types for other dates you want to enter.

- Name: sender’s name, in the standard behavior appears as a centered paragraph in small caps in the letter‘s heading.

[7]It could be seen as a matter of inconsequence, that the types Letter and Opening described above are not such input types as well. Because of the special meaning of those types, however, I have implemented them as ordinary paragraph types with a one letter mark in the left margin. Moreover, it would affect my feeling of symmetry, if the Opening type and the Closing type had such a serious different appearance.

- Signature: sender's signature, in the standard behavior appears below the Closing type. If no Signature type is used, the value of the Name type appears instead.
- Address: sender's address, in the standard behavior appears in a centered paragraph in the letter's heading below the sender's name.
- Telephone: sender's telephone number, in the standard behavior only sets the LaTeX variable `\telephonenum`.
- Place: place of the letter's making.
- Date: date of the letter's making. Place and Date, in the standard behavior, produce the place and the date in a right-aligned line below the addressee's field. If an empty Date type is used, neither place nor date appear, independent of the value of the Place type. If no Date type is used, the date of the letter 's production is used.
- Backaddress: sender's back address, in the standard behavior appears above the addressee's field in a small sans serif font.
- Specialmail: special mail information, in the standard behavior appears underlined above the addressee's field below the back address.
- Location: additional information, in the standard behavior appears on right side below the addressee's field.
- Title: the letter's title, in the standard behavior appears in a big, bold, sans serif font above the subject.
- Subject: the letter's subject, in the standard behavior appears in a bold font above the Opening paragraph.

The types Yourref, Yourmail, Myref, Customer, and Invoice produce a business letter like line above the Title line containing the fields "Your ref.", "Your letter of", "Our ref.", "Customer no.", "Invoice no.", and "Date". For the date field, the value of the Date type is used. If one of these "business letter types" is used, the value of the Place type however does not appear, but only the LaTeX variable `\fromplace` is set. The ordinary output of place and date in a right-aligned line below the addressee's field is suppressed. The types are implemented as input types provided with a label and must be used ahead of the corresponding Opening type.

- Yourref: Your ref.
- Yourmail: Your letter of.
- Myref: Our ref.
- Customer: Customer no.
- Invoice: Invoice no.

4.8.4 The new letter class: letter (koma-script v.2)

by Jürgen Spitzmüller

Koma-Script version 2.8 has introduced a new letter class `scrlttr2` which superceeds the now unsupported `scrlettr`. It has — on the LaTeX side — a completely new interface and is not compatible with the old class. Therefore, LyX supports both, though it is recommended to use the new class.

This class covers the same functionality as *letter (koma-script),* and a few more. The basic items are Address (receiver's address, same as Letter in the old layout), Opening, and Closing. NextAddress will start a new letter (i. e. you can write several letters per document). New elements are sender's E-Mail, URL, Fax, Bank and the possibility to use a Logo (via Insert ▷ Graphics) in the header.

The biggest improvement is, though, that the letter's layout is configurable at almost any needs. This can be done via the preamble or with a special style file (Letter Class Option, extension `*.lco`), that will be read in as a class option.[8] Have a look at the *koma-letter2* template that is included in LyX for examples. A detailed description is to be found in the Koma-Script documentation (*scrguide*).

4.8.5 Problems

Visualizing the Koma-Script document classes in LyX, the LyX internals cause some problems.

- The chapter number of a Chapter type appears on a line of its own above the chapter heading instead of appearing in the same line ahead of it. The cause for that is the LyX internal behavior for the labeltype Counter_Chapter in the layout file.
- The headings of the types Addchap and Addsec are only put in the "true" LaTeX table of contents, but not in the LyX table of contents (Document ▷ Table of Contents).
- The paragraphs in a *letter* document class appear in a skip separation mode, not indented. This is the standard behavior, no special LaTeX commands are needed for that. But in the Document ▷ Settings dialog the corresponding radio button indicates Indent. A Skip value always has the effect that extra LaTeX commands are inserted in the document to produce the gap, which is not what is wanted in this case.

4.9 Springer Journals (svjour)

by Martin Vermeer

[8]The KOMA package comes with some default `*.lco` files. There is, for instance, a `DIN.lco` file that follows german typesetting rules, or a `KOMAold.lco` that provides the default layout of the old `scrlettr` class. The latter can be loaded with the class option `KOMAold`, inserted via the Layout ▷ Document ▷ Extra Options field.

4.9.1 Description

These are the layout files for some of the journal formats used by Springer Verlag and listed on `http://www.springer.de/author/tex/help-journals.html`, where you should also go to fetch the class files (yes, these are LaTeX 2ε now!). It is a modular system: the things common to all journals are implemented in `svjour.inc`, which journal-specific layout files (such as, e.g., `svjog.layout` for Journal of Geodesy) can include.

This means that implementing support for any other Springer journal on this list is as simple as writing your own `sv`<myjournal>`.layout` file following the outline given in `svjog.layout`.

It is reasonably well tested only for the Journal of Geodesy. `svjour` and `svjog` come with the standard LyX distribution. Install the relevant class file (downloaded from Springer) in a proper directory, reconfigure LaTeX (in the teTeX case by running `texhash`, as root if necessary — doesn't LyX take care of this?), reconfigure LyX and it should work.

4.9.2 New styles

A large number of theorem-like styles — Claim, Conjecture, . . . Theorem.

Headnote, Dedication, Subtitle, Running_LaTeX_Title, Author_Running, Institute, Mail, Offprints, Keywords, Acknowledgements, Acknowledgement. See the Springer class file documentation for details.

4.9.3 Supported journals

- *Journal of Geodesy*: `svjog.layout` — Martin Vermeer
- *Probability Theory and Related Fields*: `svprobth.layout` — Jean-Marc Lasgouttes

Add your own, it isn't so hard!

4.9.4 Credits

These files are partly based on the older `ejour2.layout`, which was again based on a tinkered-with version of an old LaTeX 2.09 style file from Springer. All this, and the `ejour2` layout, are now defunct. Jean-Marc Lasgouttes helped out big in making me find my way around the LyX layout file mechanism.

4.9.5 Bugs

Probably. But probably less than in the old hacked-LaTeX `ejour2`.

Limitations e.g.: does not display the number for theorem-like layouts, just #.

4.10 AGU journals (aguplus)

by Martin Vermeer

4.10.1 Description

These are the layout files for some of the journals of the American Geophysical Society. It is assumed that you have both the AGU's own class files and AGUplus installed (everything to be found at`ftp://ftp.agu.org/journals/latex/journals`).

4.10.2 New styles

Redefined are Paragraph, Paragraph*. They are still called this in the LyX GUI, though their LaTeX equivalents in the AGU classes are Subsubsubsection and Subsubsubsection*.

Newly defined styles are Left_Header, Right_Header, Received, Revised, Accepted, CCC, PaperId, AuthorAddr, SlugComment. These are mostly manuscript attributes and defined in the AGU class documentation.

I suspect this is still badly incomplete.

4.10.3 New floats

Planotable and Plate. We also have a new Table_Caption.

4.10.4 Supported journals

- *Journal of Geophysical Research*: `jgrga.layout` — Martin Vermeer

Add your own, it isn't so hard! Look at the `jgrga.layout` example and `aguplus.inc`.

4.10.5 Bugs and things to remember

In order to use the new layouts, you must remember to do the following for a new document:

1. *Turn off babel.* This can be done in the layout ▷ document or document ▷ settings menu item. (AGU articles are always in English, right? So *don't* choose a language.)

2. Enter `jgrga` into the document's Extra Options field. (Yes, this is a bug.)

3. Make sure you use the `agu.bst` bibliography style, by entering agu into the second field of the BibTeX inset. None of the standard styles will do.

4.11 EGS journals (egs)

by Martin Vermeer

4.11.1 Description

This is the layout file for the European Geophysical Society journals. The needed `egs.cls` can be downloaded from the web site of the EGS under `www.copernicus.org`.

4.11.2 New styles

Right_address, Latex_Title, Affil, Journal, msnumber, FirstAuthor, Received, Accepted, Offsets. The current layout file is unfortunately very unmodular and would benefit from using the various `std*.inc` file inclusions.

4.12 Slides [aka SliTeX]

by John Weiss

4.12.1 Introduction

This section describes how to use LyX to make slides for overhead projectors. There are two document classes that can do this: the default slides class and the FoilTeX slides class. This section documents the former.

I'm going to say this again, nice and clear, so that there's no misunderstanding:

This section documents the class "slides (default)" *only.*

If you're looking for the documentation for "slides (FoilTeX)", check out section 4.13. The foils class ["slides (FoilTeX)"] is actually somewhat better than the default slides class,[9] which this section documents.

This class is the LaTeX 2$_\varepsilon$ improvement of the old SliTeX package. Every LaTeX 2$_\varepsilon$ distribution includes this class [which I'll just refer to as "slides" from now on], so you're bound to have it. As I noted earlier, there are other classes, such as foils, which also produce slides for overhead projectors and do a better job at it. However, there are some things which slides can do which the others can't, such as generate overlays. Read on to learn more!

[9]...or so I've been told repeatedly by its advocates. Having never used it, I have no idea if this claim is true or not.

4.12.2 Getting Started

Obviously, to use this document class, you need to select "slides (default)" from the class list in the Document ▷ Settings dialog. There are some other special things you should know about this class:

- Don't bother changing the options Sides and Columns. They're not supported by the slides class, anyways.
- The option Page style behaves a bit differently for this class. The possible choices and what they do are as follows:

  plain The final output contains page numbers in the lower right corner.

  headings Like plain, but also prints out any time markers you've put in. This is the default.

  empty The final output contains no page numbers, time markers, or alignment markers.

- The slides class has an extra option: `clock`. To use it, put "`clock`" in the extra class options.

  Using this options allows you to add time markers to Notes. See section 4.12.4.3 for more details.

You can also use the template file "`slides.lyx`" to automatically set up a document to use the slides class [using File ▷ New from Template to open your new document]. The template file also contains some examples of the special paragraph environments used by this class. I'll describe those next.

4.12.3 Paragraph Environments

4.12.3.1 Supported Environments

The first thing you'll notice when you start up a new slides document is the font size and type: it's the equivalent of the size "Largest" in the Sans Serif font. This is also what's used in the output. Think of this as a "visual cue" to remind you that this is a slide. Your final slides will use a larger font; ergo, you'll have less space. Of course, the larger default screen font isn't WYSIWYG, only a reminder.

The next thing that becomes obvious is the changes to the paragraph environment pull-down box [at the far-left end of the toolbar]. Most of the paragraph environments you're used to seeing are missing. There are also five new ones. That's because the slides class itself only supports certain paragraph environments:

- Standard
- Itemize

- Enumerate
- Description
- List
- Quotation
- Quote
- Verse
- Caption
- LyX-Code
- Comment

All of the other standard environments, including the section-heading environments, aren't used in the slides class.

On the other hand, you'll notice the following new environments:

- Slide
- Overlay
- Note
- InvisibleText
- VisibleText

These five are kind of quirky, due to a "feature" in LyX. You see, LyX doesn't permit you to nest any other paragraph environment into an empty environment. Now, that's fine and dandy, but it means that you wouldn't be able to start a slide with anything except plain text. To deal with this, I've performed a little "LaTeX magic."

4.12.3.2 Quirks of the New Environments

All five of the new paragraph environments are somewhat quirky due to inherent limitiations in the current version of LyX. As I just mentioned, LyX forbids environments that begin with another environment. To get around this, the Slide environment isn't a paragraph environment as described in the *User's Guide*.

You should consider Slide, Overlay, and Note to be "pseudo-environments." They look like a section heading or a "Caption," but really begin a [and, if necessary, end the previous] paragraph environment. Likewise, treat InvisibleText and VisibleText as "pseudo-commands." These two perform some action.

A common feature of all five environments, Slide, Overlay, Note, InvisibleText and VisibleText, is a rather long-ish label. The text following this label —

ordinarily the contents of the paragraph environment — is utterly irrelevant for Slide, Overlay, Note, InvisibleText and VisibleText. LyX completely ignores it. In fact, you can leave these five environments completely empty.

While you don't *have* to put any text after the rather long-ish label, you might want to. This could be a short description of the contents of the Slide, for example. In that case, enter in your descriptive comment and hit Return as you normally would.

If, on the other hand, you don't want to enter in any descriptive text, you'll hit another LyX quirk. LyX, like nature, abhors a vacuum, and will not let you start a new paragraph environment until you put something in the old one. So, do this:

- Start entering the text that will *follow* the new Slide, Overlay, Note, InvisibleText or VisibleText.
- Now move to the beginning of that paragraph.
- Next, hit Return.
- Finally, change this new, empty paragraph to a Slide, Overlay, Note, InvisibleText or VisibleText.

Some future version of LyX will, hopefully, resolve this quirkiness...

4.12.4 Making a Presentation with Slide, Overlay and Note

4.12.4.1 Using the Slide Environment

If you're expecting this section to teach you how to actually make a presentation, you'll be sorely disappointed. Naturally, I'll describe all of the ways the slides class can assist you in preparing the materials for a presentation. Filling in the contents, however, is up to you. [Then again, that *is* the LyX philosophy.]

Choosing the Slide environment [in the manner described in section 4.12.3.2] tells LyX to begin a new slide [duh]. The label for this environment/"pseudo-command" is an "ASCII line," in cool blue, followed by the label, "NewSlide:". Any text or paragraph environments that follow this one go on the new slide. It's that simple.

Slides are probably the only time you'll need to forcibly end pages in LyX (this can be specified in the Paragraph Layout dialog). In fact, you'll want to, once you finish entering the contents of one slide. If you've entered more text than can physically fit on a slide, the extra overflows onto a new slide. I don't recommend doing this, however, since the overflow slide won't have any page number on it. Furthermore, it may interfere with any Overlay you've made to accompany the oversized Slide.

The Overlay and Note environments work the same way as the Slide environment. They both create an "ASCII line" followed by a label ["NewOverlay:" and "NewNote:", respectively]. The color is a stunning magenta instead of blue, and the "ASCII line" will look different, in style and in length. The label fonts of all three also differ from one another.

As with a Slide, if the contents of a Note or Overlay exceed the physical size of a slide or sheet of paper, the extra will overflow onto a new sheet. Again, you should avoid this. It defeats the whole purpose of Notes and Overlays.

4.12.4.2 Using Overlay with Slide

The idea behind an Overlay is a slide that sits atop another slide. Perhaps you wish to discuss a figure on the main Slide before displaying the text associated with it. One way to accomplish this is tape a flap of dark paper over the part of the Slide you want to display later. This method fails, however, if you wish to overlap one graph with another, for example. You would then have to fumble while speaking to align the two separate, overlapping Slides to align the two graphs. The use of an Overlay environment in both cases makes life much easier.

Each Overlay receives the page number of its "parent" Slide, appended by "-a".[10] Clearly, you want the contents of both the Slide and the Overlay to each fit on a single physical slide! You should probably consider an Overlay as "part of" a Slide. Indeed, the LyX slides class provides a visual cue for this: the label at the start of an Overlay is shorter than that at the start of a Slide. Lastly, when you generate printable output, you'll find alignment markers in all four corners of both the Overlay page and its parent Slide. These will assist you in lining up the two physical slides.

The major problem in overlaying two slides is aligning the contents of the two transparencies. How much space should you leave for that graph on the second slide? Worse still, what if you want a graph and a sentence on second slide, but there is text on the main transparency that goes in between them? You could try and insert vertical space of the right size. The better way is to use InvisibleText and VisibleText.

As their names imply, InvisibleText and VisibleText are two command-like paragraph environments that make all subsequent text invisible and visible, respectively. Note from section 4.12.3.2 that you don't place anything *into* these two environments, however. When you create an InvisibleText, it inserts a centered, sky-blue label into the page reading "<Invisible Text Follows>". For paragraphs following this label, the parts of the Slide [or Overlay; it doesn't matter which] where they would be contain instead blank space.

For VisibleText, the corresponding centered label is "<Visible Text Follows>" in blazing green. Paragraphs following this label behave normally. Note that the beginning of a new Slide, Overlay, or Note automatically shuts off an InvisibleText. It's therefore not necessary to use VisibleText at the end of a Slide.

By now, it should be obvious how to create overlay transparencies using the proper combination of InvisibleText and VisibleText on a Slide and Overlay:

[10]Presumably, mutliple Overlays would have "-a", "-b", "-c", etc. appended to the page number of the parent Slide.

1. Create a Slide, including everything that will appear on it, whether on the main slide or on the Overlay.
2. Before each figure or paragraph that will appear only on the Overlay, insert an InvisibleText environment. If necessary, insert a VisibleText environment after the Overlay-only text.
3. Start an Overlay immediately following the Slide.
4. Copy the contents of this Slide into the Overlay.
5. Within the Overlay, change all of the InvisibleText lines to VisibleText and vice-versa.

That's it. You've just made an Overlay.

There's one problem with the way I've designed the LyX slides class: you can't make text in the middle of a paragraph invisible, nor make text in the middle of an invisible paragraph visible again. To accomplish this feat, you'll need to use some inlined LaTeX codes.[11]

4.12.4.3 Using Note with Slide

Like an Overlay, a Note is associated with a "parent" Slide. Here, too, the LyX slides class provides visual cues. The label for a Note is shorter than that of a Slide [yet longer than that of an Overlay] and, like the label of an Overlay is shockingly magenta. Additionally, the printed Note has the page number of its "parent" Slide, appended by "-1", "-2", "-3", etc. You can have multiple Notes associated with a single Slide, and, as with Slide and Overlay, you'll probably want to break up long Notes so that they fit on a single sheet of paper.

The purpose of a Note is obvious: it contains anything additional you might want to say about a Slide. It could also be used as a sheet of reminders for a particular Slide. In the case of the latter, you might want to make use of time markers. Currently, the LyX slides class has no "native" support for time markers, a SLITEX feature. So, you'll have to resort to using the LaTeX codes.

To use time markers, you'll need to specify the extra class option "`clock`" [see section 4.12.2]. This option turns on timing marks, which will appear in the lower-left-hand corner of every Note you generate. To set what appears in the time marker, you use the LaTeX commands "`\settime{}`" and "`\addtime{}`". The arguments of both commands are time measured in seconds. "`\settime{}`" sets the time marker to a given time. "`\addtime{}`" increments the time marker

[11] The commands of interest are:

- `{\invisible ... }`
- `{\visible ... }`

...and need to be marked as TEX. The text whose "visibility" you wish to change goes in between the brackets [and after the `\invisible` or `\visible` command]. If you don't know how to mark text as TEX, see the apprpriate section of the *User's Guide*.

by the specified amount. Using time markers and Notes in this fashion, you can remind yourself how much time to spend on a particular Slide.

There's one last feature to describe. Clearly, you'd like to print out all of your Slides and Overlays on transparencies while printing all of your Notes on plain paper. However, a Note *must* follow the Slide with which it is associated. What's a person to do?

Luckily, there are two LaTeX commands that allow you to select what to print out. Both must be placed into the preamble of your document. The command "`\onlyslides{\slides}`" will cause the output to contain only the Slides and Overlays. Correspondingly, the command "`\onlynotes{\notes}`" prevents the output of anything but Notes. I'd advise placing both commands in the preamble and initially comment both out. You can then preview your entire presentation as you write. When you're done writing, you can then uncomment one of the two to select what you want to print. I like to uncomment "`\onlyslides{\slides}`" , print to a file with "`-slides`" in its name, comment it back out, then uncomment "`\onlynotes{\notes}`" and print to a "`*-notes.ps`" file. I can then send either file to a printer, loading transparencies or plain paper as appropriate.

You can also provide other arguments to the "`\onlyslides{}`" and "`\onlynotes{}`" commands. See a good LaTeX book for details.

4.12.5 The slides Class Template File

I have also provided a template file, "`slides.lyx`", with the slides class. To use it, begin your new presentation with File ▷ New from Template. Your new LyX presentation file will contain an example Slide – Overlay – Note triplet. The Slide and Overlay additionally contain an example of the use of InvisibleText and VisibleText. Lastly, the preamble will contain:

```
% Uncomment to print out only slides and overlays
%
%\onlyslides{\slides}

% Uncomment to print out only notes
%
%\onlynotes{\notes}
```

One final thing: I created this class to support the LaTeX 2ε "SliTeX emulation" class, one of the built-in LaTeX 2ε classes. Neither I nor the rest of the LyX Team endorse or oppose the use of this built-in slide class. It's here if you want it or need it. There exist other LaTeX 2ε classes for creating presentations, such as the Foils class [see section 4.13] or the "`seminar`" package [present on some TeX distributions]. The latter is not yet supported under LyX.[12] I know nothing about these other classes. Try them out to see what sort of alternative they provide.

[12]Perhaps you can take on the task...

4.13 Foils [aka FoilTEX]

by Allan Rae

4.13.1 Introduction

This section describes how to use LYX to make slides for overhead projectors. There are two document classes that can do this: the default slides class and the FoilTEX slides class. This section documents the latter.

I'm going to say this again, nice and clear, so that there's no misunderstanding:

This section documents the class "slides (FoilTEX)" *only.*

If you're looking for the documentation for "slides (default)", check out section 4.12. If your machine doesn't have the foils class ["slides (FoilTEX)"] installed, you'll probably have to use the default slides class, which isn't quite as good as foils.

The foils class is designed for use with version 2.1 of the foils.cls LATEX class file which is now an integral part of LATEX 2$_\varepsilon$.

4.13.2 Getting Started

Obviously, to use this document class, you need to select "slides (FoilTEX)" from the Class entry in the Document Layout dialog. There are some settings in the Document Layout dialog that you should know about that are specific to this class:

- Don't change the options Sides and Columns on the Document Layout dialog. They're ignored by the foils class.

- The default font size is 20pt with the other options being 17pt, 25pt and 30pt.

- The default font is sans serif but all math equations are still typeset in the usual roman font.

- FoilTEX supports A4 and Letter paper sizes as well as a special size for working with 35mm slides. It doesn't support A5, B5, legal or executive paper sizes.

- Don't bother changing the Float Placement settings because they are ignored anyway. All floats appear where they are defined in the text.

- The Pagestyle setting behaves a bit differently for this class. FoilTEX provides extensive footer and header capabilities including a user-defined logo. See section 4.13.4.6 for more details. The title page is treated differently to all other pages in the document and is *always* unnumbered and *always* has the logo centered at the bottom of the page (if one is defined). The possible page style choices and what they do are as follows:

empty	The final output contains no page numbers, or other headers or footers (except footnotes of course).
plain	The final output contains page numbers centered at the bottom of the page. No other headings or footers (other than footnotes).
foilheadings	Page numbers in lower right corner. Additional headers and footers are also shown. This is also the default.
fancy	Gives you access to the fancyheadings package although its use with FoilTEX is discouraged by the writer of the FoilTEX package because of some potential page layout clashes.

4.13.2.1 Extra Options

The following options may be used in the extra class options in the Document ▷ Settings dialog.

35mmSlide	This sets up the page layout for 7.33in by 11in paper, which is about the same aspect ratio as a 35mm slide, making it a bit easier to work with this medium.
headrule	Places a rule across the page below the header on every page except the title page.
footrule	Places a rule across the page above the footer on every page except the title page.
dvips	This is automatically set each time you create a new foils document. This option tells FoilTEX to use the dvips driver to rotate those pages that are set as landscape foils.
landscape	Simply changes the page dimensions to those of a landscape page but doesn't do any rotation. Thus if you use this option you need to use an external program to rotate each page or feed your paper through your printer as landscape. Note that this option effectively reverses the roles of the Foilhead and Rotatefoilhead environments (don't worry these are described in the next section).
leqno	Equation numbers on the left.
fleqn	Flush-left equations.

4.13.3 Supported Environments

Most of the environments commonly supported in other classes are also supported by the foils class. There are several additional environments provided by FoilTEX as well as a couple added by LYX. The following environments are shared with other classes:

- Standard
- Itemize
- Enumerate
- Description
- List
- LYX-Code
- Verse
- Quote
- Quotation
- Title
- Author
- Date
- Abstract
- Bibliography
- Address
- RightAddress
- Caption
- Comment

That is, all the major environments apart from the sectioning environments. Since foils are essentially self-contained sections, with a title and body, FoilTEX provides specific commands for starting new foils and these are:

- Foilhead
- Rotatefoilhead

LYX also provides slightly modified versions of these two environments called:

- ShortFoilhead
- ShortRotatefoilhead

and the differences will be explained in the next section.

Since foils are often used in presenting ideas or new theorems and such FoilTEX also provides a comprehensive box of goodies for presenting them:

- Theorem
- Lemma
- Corollary
- Proposition
- Definition
- Proof
- Theorem*
- Lemma*
- Corollary*
- Proposition*
- Definition*

The starred versions are unnumbered while the unstarred versions are numbered. There are also two list environments added by LyX and these are:

- TickList
- CrossList

FoilTeX provides some powerful header and footer capabilities that are best set in the preamble although they may be set at any point in a document. If you want to change these settings in your document the best place to do so is at the very top of a foil, *i.e.* straight after the foilhead.

For this purpose, the following command styles are provided [Martin Vermeer]:

- My Logo
- Restriction
- Right Footer
- Right Header
- Left Header

There are also a few commands provided by FoilTeX that aren't directly supported by LyX but I'll tell you what they do and how to use them in section 4.13.5.

4.13.4 Building a Set of Foils

This section will give a simple introduction to using the different environments to build a set of foils. If you want to see an example set of foils take a look at the `Foils.lyx` file accessible from the File ▷ Open... dialog under the Examples button.

4.13.4.1 Give It a Title Page

Unlike other classes that provide Title, Author, Date and Abstract environments, foils creates the title on a page of its own. If you leave out the Date environment LaTeX will substitute the current date (every time you regenerate the output).

4.13.4.2 Start a New Foil

As I mentioned earlier, there are four ways of starting a new foil. For portrait foils you should use Foilhead or ShortFoilhead. The difference between these two environments is the amount of space between the title of the foil (the foilhead) and the body of the foil.

Landscape foils are generated using the Rotatefoilhead and ShortRotatefoilhead environments. Again the only difference is the spacing between foilhead and body. Both of the short versions have 0.5 inches less separation between the foilhead and the body.

One problem with the support for landscape foils is the requirement that you have to use the `dvips` driver to generate the PostScript® output otherwise the foils won't be rotated. It is possible to get landscape foils even if you haven't got the `dvips` driver provided you can feed your foils sideways through your printer ;-)

4.13.4.3 Theorems, Lemmas, Proofs and more

Due to a small bug in LyX you can't have two of the same type of these environments directly following each other. They must be separated by something. If you try, you will just be extending the previous environment as if you had merged the two environments together. So, how do you get around this problem? The simplest option is to insert some text between the two environments or add a LaTeX environment between the two with just a "%" in it. This will force LyX to produce two separate environments and hence the correct LaTeX output. An example is provided in the example file included with the LyX distribution. Remember, this problem only occurs if you are trying to place two of the same type of theorem-like environments one directly after the other.

4.13.4.4 Lists

You get all the commonly supported list styles found in other classes as well as two new ones. I'll only describe the new ones here. If you want to find out more about the other list environments check out the *User's Guide.* If you intend to use itemized lists you might also want to read about the Itemize Bullet Selection dialog described above in section 3.10.

The two new list styles, TickList and CrossList, are designed to make it easier for you to create lists of do's and don'ts or right and wrong by providing dedicated environments that use a tick or a cross as the label of the list. These lists are in fact dedicated variants of the Itemize environment. They do however require that you have the `psnfss` packages installed.

4.13.4.5 Figures and Tables

FoilTeX redefines the floating tables and figures so that they appear exactly where they are in the text rather than pushing them to the top of the page or to some user specified location. In fact if you change the float placement settings they are simply ignored.

4.13.4.6 Page Headers and Footers

My Logo and Restriction are two commands used to control the left-footer text string. The first is meant to allow you to include a graphic logo on your foils and defaults to "-Typeset by FoilTeX-". While the second is meant to provide a classification for the audience, *e.g.* Confidential. It is empty by default.

The remaining page corners can be filled by Right Footer (which defaults to page numbers), Right Header (top right) and Left Header (top left).

4.13.5 Unsupported FoilTEX Goodies

All the commands mentioned below need to be set in a LATEX environment or as TEX within another environment.

4.13.5.1 Lengths

All lengths are adjusted using the `\setlength{`*lengthname*`}{`*newlength*`}` command. Where *lengthname* should be replaced by the name given to the length you want to change and *newlength* is the length value. All lengths should be specified in units of length such as inches (`in`), millimeters (`mm`) or points (`pt`) or relative to some document or font-based length such as `\textwidth`.

It's possible to change the spacing between a foilhead and the body of the foil by adjusting the length specified by `\foilheadskip`. For example, to make *all* foilheads 0.5 inches closer to their bodies put the following in the preamble: `\setlength{\foilheadskip}{-0.5in}`

The spacings around floats can be adjusted by setting these lengths:

`\abovefloatskip`	Separation between the text and the top of the float
`\abovecaptionskip`	Separation between the float and the caption
`\belowcaptionskip`	Separation between the caption and the following text
`\captionwidth`	You can make the captions narrower than the surrounding text by adjusting this length. Best done relative to `\textwidth`.

There are also several title page related lengths that you may find useful if you have a long title or several authors:

`\abovetitleskip`	Separation from headers to Title
`\titleauthorskip`	between Title and Author environments
`\authorauthorskip`	between multiple Author lines
`\authordateskip`	between the Author and the Date
`\dateabstractskip`	between the Date and the Abstract

The last length related command affects all the list environments. If you place `\zerolistvertdimens` *inside* a list environment then all the vertical spacing between the list items is removed. Note that this is a command not a length so it doesn't require `\setlength` like the stuff mentioned above.

4.13.5.2 Headers and Footers

The `\LogoOn` and `\LogoOff` commands control whether the logo in the MyLogo definition appear on a given page. If you put `\LogoOff` in the preamble then none of the foils will have the logo on them. If you don't want the logo on a particular page place the `\LogoOff` directly after the foilhead of that page and the `\LogoOn` directly after the next foilhead.

If you decide to use the fancy page style setting in the Document Layout dialog you should probably add `\let\headwidth\textwidth` to your preamble so headers and footers on landscape pages are correctly placed when rotated. This is due to some clashes between the page layouts provided by the fancyheadings package and the foils class.

4.14 Latex8 (IEEE Conference Papers)

by Allan Rae

4.14.1 Introduction

Since this class is specifically for writing submissions to IEEE sponsored conferences I strongly recommend that you get a copy of their Authors Kit. The latex.sty package and associated bibliography style file is included in the kit. The Authors Kit is usually sent out by email once your initial submission has been accepted. There is a lot of useful information in the Authors Kit explaining formatting restrictions and so on and I will assume you have read this since that means I don't have to repeat it all here.

4.14.2 Getting Started

[AR. more to come]

4.14.3 Supported Environments

- Standard
- Title
- Author
- E-mail
- Affiliation
- Abstract
- Section
- SubSection
- Caption

4.14.4 Differences Between Screen and Paper

There are slight differences in appearance mainly with the presentation of section counters. On screen the trailing period of the section counter is missing but it will appear in the output so don't let this worry you.

4.15 Hollywood (Hollywood spec scripts)

by Garst Reese

4.15.1 Introduction

Getting the format of a Hollywood script right is a "rite of passage." It is designed to make the readers focus on content and to be easy and familiar for the actors to read. Each page of a script should be one minute of film. Nothing goes in a script that you cannot see or hear on screen. The courier 12 pt font should be used throughout. No italics.

4.15.2 Special problems

Speakers' lines should NEVER break in mid-sentence. If a speaker's lines continue over a page break, repeat the Speaker title followed by (Cont'd).

4.15.3 Special features

Insert the Speaker names as labels then cross-reference the label to insert the name. The cross-reference dialog will show the current cast of characters. You can use this to insert the speaker name in narratives also.

4.15.4 Paper size and Margins

USLetter, left 1.6in, right 0.75in, top 0.5in, bottom 0.75in

4.15.5 Environments

The following environments are available. You can use hollywood.bind to get the bind keys shown at the right.

- Standard
  Used where nothing else works. Try to avoid it.
- FADE_IN: M-z S-I
  Usually followed by something like "on Sally waking up."
- INT: M-z i
  Introduces a new INTERIOR camera set-up. Always followed by DAY or NIGHT, or something similar to define the lighting required. Everthing on this line in CAPS.

- EXT: M-z e
  Introduces a new EXTERIOR camera set-up. Everthing on this line in CAPS.
- Speaker M-z s
  The character speaking.
- Parenthetical M-z p
  Instructions to the speaker. The () are automatically inserted, but only the (will show in LyX. Both will be printed.
- Dialogue M-z d
  What the Speaker says.
- Transition M-z t
  Camera movement instruction. e.g. CUT TO:
- FADE OUT: M-z S-I
- Author M-z S-A
- Title M-z S-T
- Right_Address M-z r

4.15.6 Script jargon

- (O.S) — off screen
- (V.0) — voice over
- b.g. — background
- C.U. — close-up
- PAN — camera movement
- INSERT — cut to close-up of

4.16 Broadway

by Garst Reese

4.16.1 Introduction

Broadway is for writing plays. The format is more decorative than Hollywood, and much less standardized. This format should be suitable for workshops.

4.16.2 Special problems

The same as in Hollywood.

4.16.3 Special features

Insert the Speaker names as labels then cross-reference the label to insert the name. The cross-reference dialog will show the current cast of characters.

4.16.4 Paper size and Margins

USLetter, left 1.6in, right 0.75in, top 0.5in, bottom 0.75in

4.16.5 Environments

The following environments are available. You can use broadway.bind to get the bind keys shown at the right.

- Standard
  You should not have to use this, but it is here for anything that does not fit otherwise.
- Narrative M-z n
  Used to describe stage setting and the action. First use of speaker names in all CAPs.
- ACT M-z a
  Automatically numbered. On screen it will be arabic, but will print as Roman.
- ACT* M-z S at
  Subtitle for ACT. It is just centered text.
- SCENE M-z S-S
  Not automatically numbered. You supply the number. This is because I couldn't figure out how.
- AT_RISE: M-z S-R
  A special case of Narrative to describe the setting and action as the curtain rises.
- Speaker M-z s
  The speaker's (actor's) title, centered in all CAPS.
- Parenthetical M-z p
  Instructions to the speaker. The parentheses are automatically inserted. The (will appear on screen, but both will be in the printed play. This environment is only used within Dialogue.
- Dialogue M-z d
  What the Speaker says.
- CURTAIN M-z S-C
  The curtain comes down.

- Title M-z S-T
- Author M-z S-A
- Right_Address M-z r

Hello there.

4.17 RevTEX4

by Amir Karger

The Revtex 4 textclass works with the American Physical Sociey's RevTEX 4.0 (the β release of May, 1999) class.

LYX has a Revtex textclass, which works with RevTEX 3.1. However, v3.1 is basically obsolete, as it works with LATEX 2.09. That means that it doesn't interact very well with LYX, which requires LATEX 2$_\varepsilon$, although it has been kludged to work. Since RevTEX 4.0 has been designed to work much more cleanly with LATEX 2$_\varepsilon$, LYX with the RevTEX 4 textclass should also be pretty easy to use.

These documents are supposed to be used in *addition* to the RevTEX 4.0 documents, so we don't describe any of the special RevTEX macros, and assume you'll know what to put in the preamble if necessary.

4.17.1 Installation

All you need to do is install RevTEX 4, as described in the package's README file. the package can be found at The RevTeX 4 Web Site `http://publish.aps.org/revtex4/`. Install it somewhere that LATEX can see it. Test it by trying to LATEX a short RevTEX 4 document in some random directory (i.e., not the directory where you installed the class file.) Then, if you reconfigure LYX, it will find the class file and let you use the RevTEX4 textclass.

Probably the easiest way to get started is either to import a RevTEX 4 document using `reLyX`, or to use the Revtex 4 template, found in the templates directory.

4.17.2 Preamble Matter

Optional arguments to `\documentclass`, like "preprint" and "aps", go in the Extra Options field in the Document Layout dialog, as usual. Remember that in RevTEX, at least one optional argument is required!

Other preamble matter, like `\draft` etc. goes in the Latex Preamble dialog, also as usual.

4.17.3 Layouts

The layouts basically correspond to the commands in RevTEX4.0. For example, the Email layout corresponds to `\email{}`. Note that (at least as of RevTEX 4.0 Beta), the Address and Affiliation layouts are exactly equivalent, so you shouldn't need to use both.[13]

4.17.4 Important Notes

There are a couple of important unique aspects of RevTEX 4 which might cause bugs that will be even more confusing in LyX.

In RevTEX, the `\thanks` command goes *outside* the `\author` command. The LyX equivalent is that there is a separate Thanks layout. Do *not* write footnotes in the Author layout, or weird things may happen. See the RevTEX 4 documentation for more details.

Also, the Author Email, Author URL, and Thanks layouts must be placed *in between* the Author layout and the corresponding Address (or equivalent Affiliation) layout. If you put the Thanks after the Address, the LaTeX won't compile.

4.17.5 Drawbacks

The main problem with this layout is that you can't use the optional arguments to layouts like Email and Title. (The problem is not unique to this layout; you can't use optional arguments to the Section layouts either.) This means that after you export that file to LaTeX (which you'll need to do eventually to send it in to APS), you'll need to edit the LaTeX file with a text editor to add the optional arguments to set, e.g., the running title for the page headers. Lacking these layouts makes the `\altaffiliation` (and the equivalent `\altaddress`) useless, so the corresponding layouts don't exist, and will have to be added by hand.[14]

4.18 Article (mwart), book (mwbk) and report (mwrep)

by Tomasz Luczak

The LyX document classes *article (mwart)*, *report (mwrep)* and *book (mwbk)* correspond to the LaTeX document classes `mwart.cls`, `mwrep.cls` and `mwbk.cls`, resp. They are replacements for the standard document classes `article.cls`, `report.cls` and `book.cls`, resp., and fit better to Polish typography conventions in a number of points.

Basic differences:

[13] In case you're curious, both were included so that `reLyX` would be able to translate both `\address` and `\affiliation`.

[14] *Note from JMarc:* actually, LyX 1.3.0 supports some forms of optional arguments, but this layout has not been updated yet to take advantage of it.

- Unnumbered titles (with star, eg. Section*) are added into table of contents,
- Additional page styles:

  uheadings header with separated lines,

  myheadings custom header, contents headers via commands: `\markright` and `\markboth`,

  myuheadings custom header with separated lines,

  outer page number is placed on outer side of page

- Options

  rmheadings serif titles — default,

  sfheadings sansserif titles,

  authortitle on title page first placed is author next title — default,

  titleauthor on title page first placed is title next author,

  withmarginpar reserve place on page for margins.

4.19 Elsevier Journals

By Rod Pinna

Elsevier Science Publishers B.V. provides a standard LaTeX document class (`elsart.cls`) for submitting articles to their various journals. The style file can be downloaded directly from their web site: `http://authors.elsevier.com/`. Instructions are supplied along with the class file, which details the requirements of the publishers. LyX includes package that allows for the use of this class, by a layout and a template file. Installation of the class file is the same as for any other LaTeX package; instructions are provided in the Elsevier documentation.

To make use of `elsart.cls`, a file `elsart.layout` is supplied. As the Elsevier class file is based mainly on the standard article class, most of the normal functionality is provided. The Elsevier class defines a number of mathematical environments, which are similar to the AMS environments. These commands are all described in the Elsevier documentation, and are available in LyX.

The easiest way to use the Elsevier style is to base documents on the included template file. It is best not to use options such as fancy headings or the geometry package, as elements such as these are defined by Elsevier in their style file. Ideally, no extra packages except those mentioned in the Elsevier documentation should be used. Essentially, Elsevier require as "clean" a LaTeXfile as possible, as their intention is to take the supplied file and replace the class file with one for the particular journal to which the paper has been submitted. This also means that not too much time should be spent on the formating of the document. When it comes to be published, this will change anyway. The rest of the usage for this layout is substantially the same as for the normal article class. For details of what Elsevier do and don't allow, refer to their documentation.

4.20 Memoir

By Jürgen Spitzmüller

4.20.1 Overview

Memoir is a very powerful and constantly evolving class. It has been designed with regard to fictional and non-fictional literature. Its aim is to let the user have maximum control over the typesetting of his document. Memoir is based on the standard book class, but it can also emulate the article class (see below).

Peter Wilson, the developer of Memoir, is known as the author of lots of useful packages in the LaTeX world. Most of them have been merged with Memoir. Therefore, it is much easier to layout the table of contents, appendices, chapter designs and such. LyX, though, does not support all of these goodies natively. Some of them might be added to forthcoming releases[15], lots will probably never, due to the limitations of LyX's framework. Of course you can still use all features with the help of some native LaTeX commands (ERT[16]). In this section, we can only list those features which are natively supported by LyX. For detailed descriptions (and for the rest of features) we are recommending to have a look at the detailed manual of the Memoir class[17], which is not only a user guide for the class, but also both a comprehensive description on good typesetting and a superb example for good typesetting itself.

4.20.2 Basic features and restrictions

Memoir supports basically all features of the standard book classes. There are, however, some differences, as follows:

Font sizes: Memoir has a broader range of font sizes: 9, 10, 11, 12, 14, 17

Page style: The fancy page style is not supported, due to a command clash between Memoir and the fancyhdr package (they are both defining a command with the same name, which confuses LaTeX). Instead, Memoir comes with a bunch of own page styles (see Layout ▷ Document ▷ Page Style). If you want to use these for the chapter pages, you have to use the command `\chapterstyle` in the main text or in preamble (e. g. `\chapterstyle{companion}`).

Sectioning: Sectionings (chapter, section, subsection etc.) are coming with an optional argument in the standard classes. With this, you can specify an alternative version of the title for the table of contents and the headers (for instance, if the title is too long). In LyX, you can do this via Insert ▷ Short Title at the beginning of a chapter/section. Memoir features a second optional argument and thus separates the table of contents from the header. You can define three variants of a title with this: one for the main

[15]You are invited to send suggestions to `lyx-devel@lists.lyx.org`.
[16]Cf. section 2.4 for details.
[17]Cf. `CTAN:/macros/latex/memoir/memman.pdf`.

text, one for the table of contents, and one for the headers. Simply insert two optional arguments if you need this feature, the first one containing the short title for the Table of Contents, the second one containing an alternative short title for the headers.

TOC/LOT/LOF: In the standard classes (and in many other classes), the table of contents, the list of figures and the list of table start a new page automatically. Memoir does not follow this route. You have to insert a page break yourself, if you want to have one.

Titlepage: For some unknown reason, Memoir uses pagination on the title page (in the standard classes, title pages are "empty", i. e. without pagina). If you want an empty title page, type `\aliaspagestyle{title}{empty}` in the preamble.

Article: With the class option *article* (to be inserted in Layout ▷ Document ▷ Extra Options), you can emulate article style. That is, counters (footnotes, figures, tables etc.) will not be reset on new chapters, chapters don't start a new page (but are—in contrary to "real" article classes—still allowed), parts, though, use their own page, as in book.

Oldfontcommands: By default, Memoir does not allow the use of the deprecated font commands, which have been used in the old LaTeX version 2.09 (e. g. `\rm`, `\it`). It produces an error and stops LaTeX whenever such a command appears. The class option *oldfontcommands* reallows the commands and spits out warnings instead (which does at least not stop LaTeX). Since a lot of packages and particularly BibTeX style files are still using those commands, we have decided to use this option by default.

4.20.3 Extra features

We will only describe the features supported by LyX (which is not much currently). Please consult the Memoir manual[18] for details.

Abstract: You may wonder why an abstract is an extra feature. Well, it is in book class. Usually books don't have abstracts. Memoir, however, has. You can use it whereever and how often you like.

Chapterprecis: You may know this from belletristic: The contents of a chapter is shortly described below the title and also in the table of contents (e. g. *Our hero arrives in Troia; he loses some friends; he finds others*). Chapterprecis does exactly this. It is therefore only sensible below a chapter.

Epigraph: An epigraph is a smart slogan or motto at the beginning of a chapter. The epigraph environment provides an elegant way of typesetting such a motto. The motto itself (text) and its author (source) are divided

[18]Cf. `CTAN:/macros/latex/memoir/memman.pdf`.

by a short line. Unfortunately, we have to fool LyX a bit here again, since the environment needs two arguments (text and source). In this case, we have to use curly brackets (in TeX mode) between the two arguments: *<smart slogan>* `}{` *<author of the slogan>*.

Poemtitle: Memoir has lots of possibilities to typeset poetry (up to very complex figurative poems). Lyx can only support a few of them. One is poemtitle, which is a centered title for poems, which will also be added to the table of contents (verse is the standard environment for poems. Memoir has some enhanced versions of verse, but you need to use ERT, because they have to be nested inside regular verse environments, which is not possible with LyX).

Poemtitle*: Same as poemtitle, but it adds no entry to the table of contents.

Chapter 5

Importing and Exporting Alternate File Formats

5.1 Considerations

Importing and exporting LyX documents from/to other formats has been touched on briefly in the *User Guide*. Here we describe more of the gory details needed to understand just what is going on when you click on the File ▷ Import and File ▷ Export menu items.

5.2 Importing Other Formats

5.2.1 LaTeX

Translating from LaTeX into LyX is performed by a Perl script called reLyX. Although it is a standalone program which can be called from the command line, LyX will call it automatically when a LaTeX document is imported. See section 5.4 for a complete description. There are no user tunable parameters for reLyX within LyX.

5.2.2 ASCII Text

When importing plain ASCII text, there are two methods of reading the file. Importing "as lines" preserves all the linebreaks in the ASCII; to LyX, then, each line looks like a paragraph. Importing "as paragraphs" assumes that consecutive lines separated by only a single linebreak form a single paragraph. Successive linebreaks with no intervening text are thus assumed to be paragraph delimiters.

5.2.3 Noweb

[Editor's note: Needs to be written, obviously - any volunteers? — mer]

5.3 Exporting Other Formats

5.3.1 LaTeX

LyX generates two types of LaTeX files: stripped down versions for the normal processing (View DVI, etc.) which one normally never sees[1], and human readable forms which are suitable for exchanging with your colleagues. The only settable option for the translation is the line length of the output file. The default is 65 characters, but it can be set in Tools ▷ Preferences using the Ascii line length field.

5.3.2 Device Independent Files

Device Independent files (DVI files) are produced by running LaTeX on your document. There are no user settable options.

5.3.3 PostScript®

The next step in the conversion chain is converting a DVI file into Postscript®. You can either use File ▷ Export ▷ Postscript or, if you need more control on the result, File ▷ Print. If you use the later, note that it is possible to configure, in Tools ▷ Preferences, the options passed to the dvips program to achieve different effects.

5.3.4 ASCII text

Exporting as ASCII attempts to preserve the "shape" of the document as well as possible, but things like centering and indentation are thrown out; paragraphs are separated by blank lines. Section numbering and cross-references are done correctly, so the resulting text files is remarkably readable. The only changeable option is the length of lines, as for LaTeX output.

5.3.5 HTML

LyX documents can be converted to hypertext markup, usually by converting to LaTeX first, then converting that to HTML. Three LaTeX→HTML converters are currently known to LyX: `tth`, `latex2html`, and `hevea`. Though they are autodetected, you can overide the selection in preferences. You can also include further command line options in this dialog.

5.3.6 PDF

by Dekel Tsur (mostly)

[1]The resulting file is a perfectly valid LaTeX file, though the preamble might look a bit strange since it includes some definitions used by LyX which wouldn't show up in most human-written files.

The fastest way to generate a basic PDF file (no tags, links, etc.) with any version of LyX is to save the document as a Postscript® file, then run the `ps2pdf` command on it. Starting with version 1.1.6, the menu item File->Export->PDF will do all this for you. There are some issues with fonts that you need to pay attention to: see Section 5.3.6.2. Also, as of version 1.1.6, there is a better method that will generate much more sophisticated files.

5.3.6.1 Use pdfLaTeX

With pdfLaTeX you need to convert your eps figures to PDF (see Section **??**), and you cannot use pstricks. On the other hand, with pdfLaTeX it is possible to insert directly images in JPEG or PNG format, use TrueType fonts, and more.

5.3.6.2 Why does the text look so bad when viewed with Acrobat Reader?

The problem is that bitmap fonts are displayed poorly by Acrobat Reader. When creating a PDF from the LyX file, you need to use outline font instead of the default bitmap fonts (in fact, you should also use outline fonts for Postscript files). Recent LaTeX distributions come with Postscript® Type 1 version of the standard (Computer Modern) fonts. pdfLaTeX uses these font by default. Dvips doesn't use these fonts by default, so to make it use them, add the following to lines to your `~/.dvipsrc` file

```
p+ psfonts.cmz
p+ psfonts.amz
```

If the default LaTeX font encoding (OT1) is used, nothing else need to be done. However, if the T1 font encoding is used, then LaTeX uses the newer EC fonts, for which there are no Type1 version. The solution is to use the ae package which emulates T1 coded fonts using the standard CM fonts. This is done by adding `\usepackage{ae,aecompl}` to the preamble of the LyX file. However, some glyphs are missing from the CM fonts (e.g. eth, thorn), and they are taken from the EC fonts. Therefore you get these glyphs as bitmaps.

Note: LyX uses by default the T1 font encoding. If you wish to use the default font encoding (this is not recommended, unless you only write English documents), clear the field TeX encoding in preferences (tabs Outputs, Misc).

An alternate option is to use the standard Postscript® fonts instead of the Computer Modern fonts. To do that, you need to select pslatex as the global font in the document layout dialog. When using the Postscript® fonts, the result PDF file is smaller as the fonts are not saved into the file. Furthermore, the Postscript® fonts include all T1 glyphs. On the other hand, the Postscript® fonts have no bold symbol font, so poor man's bold must be used (see Section 5.3.6.3). The Postscript® fonts also look different from the Computer Modern fonts.

To sum up, both the Computer Modern and the Postscript® fonts gives good results (with few exceptions). The decision of which one to use is a matter of taste.

5.3.6.3 Why doesn't the \boldsymbol{} command work when I use pslatex?

The Postscript® fonts do not have a bold symbol font. The solution is to use the `\pmb{}` (poor man's bold) command.

It is possible to redefine the `\boldsymbol` command to use `\pmb` by putting

```
\renewcommand{\boldsymbol}[1]{\pmb{#1}}
```

in the preamble.

5.3.6.4 Is it possible to do write latex code which is processed only when running pdfLaTeX?

Yes. Here is an example:

```
\newif \ifpdf
   \ifx \pdfoutput \undefined
      \pdffalse
   \else
      \pdftrue
\fi
\ifpdf
   \pdfinfo { /Author (your name and e-mail address)
      /Title (official title -- i.e., title element)
      /Subject (one line description of the document)
   }
   \pdfcatalog { /PageMode (/UseNone)
   % /OpenAction (fitbh)
   }
   \usepackage[pdftex]{hyperref}
\else
   \usepackage[ps2pdf]{hyperref}
\fi
```

5.3.6.5 How can I make URLs clickable ?

See the references here :
`http://wiki.lyx.org/pmwiki.php/FAQ/PDF`

5.3.7 Custom

Custom exports are possible if you have some particularly weird format you wish to convert to, assuming you have the relevant converter, of course. The format of the *input* file can be chosen in the File ▷ Export ▷ Custom dialog; LyX will automatically convert the file to this point, then feed it to your custom converter. The possible values are all formats that LyX can produce from its own documents.

The converter command is also specified in the dialog.It should be a completely qualified command line which uses the variable `$$FName` to specify the name of the file. If this variable is not given, then the file will be sent to the standard input of your command. You may have to apply a bit of ingenuity to escape this sequence correctly so that it is compatible with your shell.

While it is not possible to save this command using the Preferences dialog, you can manually edit your `.lyx/preferences` to add a line like

```
\custom_export_command "mycommand $$FName"
```

5.4 The Complete reLyX Description

5.4.1 Synopsis

The simplest way to use reLyX is via the File ▷ Import command in LyX. That runs reLyX on the given file and loads the resulting file into LyX. You should try that first, and call it from the command line only if you need to use more complicated options.

reLyX [**-c** *textclass*] [**-df**] [**-o** *outputdir*] [**-r** *renv1*[,*renv2...*]] [**-s** *sfile1*[,*sfile2...*]] *inputfile*

reLyX -p -c *textclass* [**-df**] [**-o** *outputdir*] [**-r** *renv1*[,*renv2...*]] [**-s** *sfile1*[,*sfile2...*]] *inputfiles*

reLyX -h

5.4.2 Options

-c Class. By default, when reLyX sees a `\documentclass{foo}` command, it creates a file of textclass "foo" and reads the LyX layout file for that class. Use **-c** to declare a different textclass (and read a different layout file).

-d Debug. By default, reLyX gives sparse output and deletes the temporary files which were created during translation. Using the **-d** flag will create much more output (both to stdout and stderr) and leave the temporary files around.

-f Force. reLyX will not run if the `.lyx` file it would generate already exists Use the **-f** option (carefully) to clobber any existing files.

-h Help. Print out usage information and quit

-o Output directory. With this option, all temporary files and LyX output files (for the given input file, for any included files, or for any file fragments given with the **-p** option) will be put into *outputdir*. Otherwise, for each file *dir/foo.tex*, the temporary files and the LyX output file will be created in *dir*. This can be useful if a file includes files from other directories which you want to consolidate in one directory, or if you don't have write permission on the directory the LaTeX files are in.

-p Partial file. The input files are LaTeX fragments, with no preamble matter or `\begin{document}` commands. This option requires the **-c** option, since there are no `\documentclass` commands in the files reLyX is translating. When using this option, you can translate more than one file, as long as all files are the same class. The LyX file created by reLyX can be included in an existing LyX file using Insert ▷ File ▷ LyX Document.

-r Regular environments (see the Section 5.4.5.4). If you give more than one environment, separate them with commas (not spaces). You'll probably need to quote the environment list, especially if it has asterisk environments (foo*) in it. If you use this command often, considering creating a personal syntax file.

-s Syntax files. Input (one or more quoted, comma-separated) syntax files to read in addition to the default. (see the section Section 5.4.5.4 for details).

5.4.3 Description

5.4.3.1 Introduction

reLyX will create a LyX file *dir/foo.lyx* from the LaTeX file *dir/foo.tex* (unless the **-o** option is used).

Suffixes `.tex`, `.ltx` and `.latex` are supported. If *inputfile* does not exist and does not have one of these suffixes, reLyX will try to translate *inputfile.tex*. (This is similar to the behavior of LaTeX.)

The purpose of reLyX is to translate *well-behaved* LaTeX 2ε into LyX. If your LaTeX file doesn't compile—or if you do weird things, like redefining standard LaTeX commands—it may choke. LaTeX209 will often be translated correctly, but it's not guaranteed.

reLyX has some bugs and lacks a few features. However, its main goals are:

- Get through a well-behaved LaTeX 2ε file without crashing
- Translate a lot of that file.
- Localize the parts that can't be translated and copy them in TeX mode

It achieves these main goals pretty well on most files.

There are many improvements that can and will be made to reLyX in the future. However, we wanted to get reLyX out there early on, to make it easier for new LyX users to read in their existing LaTeX files.

5.4.3.2 Usage

Here's a more lengthy description of what you should do to translate a LaTeX document into LyX.

- Run reLyX.

  reLyX will inform you of its progress and give any warnings to stderr, so if you don't want any output at all, try (in csh) "`reLyX foo.tex >& /dev/null`" or (in bash) "`reLyX foo.tex 2>&1 >/dev/null`". You should NOT redirect standard output to `foo.lyx`.

- Run LyX on the resulting .lyx file.

  In theory, most of the file will have been translated, and anything that's untranslatable will be highlighted in red (TeX mode). In theory, LyX will be able to read in the file, and to create printed documents from it, because all that untranslated red stuff will be passed directly back to LaTeX, which LyX uses as a backend. Unfortunately, reality doesn't always reflect theory. If reLyX crashes, or LyX cannot read the generated LyX file, see Section 5.4.3.5 or the `BUGS` file.

- Change things that are in ERT boxes (TeX code) by hand in LyX.

  As mentioned above, you should be able to print out the LyX file even without doing this. However, changing a command in TeX mode to the corresponding LyX object will allow you to take advantage of LyX's WYSIWYM editing.

  reLyX is not guaranteed to create a LyX file which generates exactly the same output as the LaTeX file, but it should come close. reLyX will generally err on the side of translating less to ensure that dvi or ps files are accurate, even though this leads to more "evil red text" and less WYSIWYM.

- PROOFREAD THE DOCUMENT!!

  I'm sure you were planning on doing this anyway, but it's particularly important after translating a LaTeX document. reLyX is, at least now, better at "macro-translating" (translating the whole document) than "micro-translating" (translating every little detail). For example, you may see extra spaces or deleted spaces. Space handling has improved, but it's not perfect.

5.4.3.3 What reLyX Can Handle

reLyX understands many LaTeX commands. It will translate:

- regular text, including mini-commands like ˜, ”, `\@`, `\TeX`, as well as accented characters like `\'{a}`, and the special cases ¿ and ¡

- title commands like `\author`, `\date`, `\title`, `\thanks` and the abstract environment
- heading commands like `\section` including starred commands (`\section*`)
- Environments: `quote`, `quotation`, and `verse`; `center`, `flushright`, and `flushleft`
- `itemize`, `enumerate`, and `description` environments, and their `\item` commands. Also, well-behaved nested lists
- cross-referencing commands: `\ref`, `\pageref`, `\label`, and `\cite`
- `\footnote` and `\margin`
- font-changing commands including `\em`, `\emph`, `\textit`, and corresponding commands to change family, size, series, and shape
- `\input{foo}` (or `\input{foo.blah}`) and `\include{foo}`. Plain TEX `\input` command "`\input foo.tex`" is also supported.
- `tabular` environment, and commands that go inside it like `\hline`, `\cline`, and `\multicolumn` (but see below)
- float environments `table` and `table*`, as well as `\caption` commands within them
- `thebibliography` environment and `\bibitem` command, as well as BibTEX's `\bibliography` and `\bibliographystyle` commands
- miscellaneous commands: `\hfill`, `\\`, `\noindent`, `\ldots`...
- documentclass-specific environments (and some commands) which can be translated to LYX layouts
- arguments to certain untranslatable commands (e.g. `\mbox`)

Some of this support may not be 100% yet. See below for details

reLYX copies math (almost) verbatim from your LATEX file. Luckily, LYX reads in LATEX math, so (almost) any math which is supported by LYX should work just fine. A few math commands which are not supported by LYX will be replaced with their equivalents, e.g., `\to` is converted to `\rightarrow`. See the section on *Syntax Files* for more details.

reLYX will also copy any preamble commands (i.e., anything before `\begin{document}`) verbatim, so fancy stuff you've got in your preamble should be conserved in dvi and printed documents, although it will not of course show up in the LYX window. Check the preamble to make sure.

5.4.3.4 What reLyX Can't Handle — But it's OK

- figures and `tabular*` tables
- minipages
- spacing commands (`\vspace`, `\pagebreak`, `\par`)
- `\centering`, `\raggedleft`, `\raggedright`
- `\verb` and `verbatim` environment. reLyX is careful to copy *exactly* in this case, including comments and whitespace.
- some unknown (e.g., user-defined) environments and commands

reLyX copies unknown commands, along with their arguments, verbatim into the LyX file. Also, if it sees a `\begin{foo}` where it doesn't recognize the "foo" environment, it will copy verbatim until it sees `\end{foo}` (unless you use the `-r` option). Hopefully, then, most of these unknown commands won't cause reLyX to break; they'll merely require you to do some editing once you've loaded the file up in LyX. That should be less painful than editing either the `.tex` or the `.lyx` file using a text editor.

5.4.3.5 What reLyX Handles Badly — a. k. a. BUGS

Since reLyX is relatively new, it's got a number of problems. As it matures, these bugs will be squished. A number of bugs and missing features can be found listed on the LyX bug tracker, LyX Bugzilla `http://bugzilla.lyx.org/`.

If reLyX is choking on something, or LyX can't read it after reLyX translates it, the best thing to do is to put `\begin{reLyXskip}` before the offending text, and `\end{reLyXskip}` after it. I call this a "skip" block. reLyX will copy this block exactly, in TeX mode. Then edit the resulting LyX file, and translate the unknown stuff by hand. The `reLyXskip` environment is magical; the `\begin` and `\end` commands will not be put into the LyX file.

- "Exact" copying of unknown environments and commands isn't quite exact. Specifically, newlines and comments may be lost. This will yield ugly LyX, but in almost all cases the output will be the same. However, certain parts of the file will be copied perfectly, including whitespace and comments. This includes: the LaTeX preamble, `verbatim` environments and `\verb` commands, and skip blocks.
- reLyX translates only a few options to the `\documentclass` command. (Specifically 1[012]pt, [letter|legal|executive|a4|a5|b5]paper, [one|two]side, landscape, and [one|two]column.) Other options are placed in the extra class options field in the Document ▷ Settings dialog.

  More importantly, reLyX doesn't translate `\usepackage` commands, margin commands, `\newcommands`, or, in fact, anything else from the preamble. It simply copies them into the LaTeX preamble. If you have margin commands in your preamble, then the LyX file will generate the right margins. However, these margins will override any margins you set in the LyX Document▷Settings dialog. So you should remove the options from the preamble to be safe. The same goes for setting your language with babel, `\inputencoding`, `\pagestyle`, etc.

- The foil class has a couple bugs. reLyX may do weird things with optional arguments to `\foilhead` commands. Also, it may handle `\begin{dinglist}` incorrectly (although the stuff in the environment should translate normally).

reLyX is hopefully rather robust. As mentioned above, it may not translate your file perfectly, but it shouldn't crash. If it does crash—and the problem is not one of those mentioned above or in the *BUGS* file—see Section 5.4.5.1.

5.4.3.6 What LyX Can't Handle

LyX itself is missing a couple features, such that even if reLyX translates things perfectly, LyX may still have trouble reading it. If you really need these features, you can export your final document as LaTeX, and put them back in. See *BUGS* for more details on these bugs.

- For a number of commands, LyX does not support the optional argument. Examples include `\sqrt`, `\chapter` (and other sectioning commands), and \\. reLyX will automatically discard the optional arguments with a warning to stdout. LyX also ignores the width argument for the `thebibliography` environment.
- Centering (or right or left justifying) works on full paragraphs.
- LyX support for tables isn't perfect. For complicated tables, use a "skip" block, so that they will be copied in TeX mode.
- The LyX math editor can't handle the AMS-LaTeX math environments align, split, etc. So those environments will be copied in TeX mode. You can change `equation*` environments to the exactly equivalent displaymath, and then they will be translated correctly.

5.4.4 Examples

```
reLyX -df -o ‘‘my/dir’’ -r ‘‘myenv’’ foo.tex > foo.debug
```

The above will create a file my/dir/foo.lyx from foo.tex, overwriting if necessary. When it finds a `\begin{myenv} ... \end{myenv}` block, it will translate the stuff within the block, but copy the `\begin` and `\end` commands in TeX mode. Finally, I'm going to keep the temporary files around (they will also be in my/dir/) and output lots of debugging information into the file foo.debug.

5.4.5 Notes

5.4.5.1 Bug Reports

If reLyX is crashing or otherwise acting strangely—in ways other than those described in Section 5.4.3.5 or the bug tracker—then please run reLyX **-d**. That will allow you to figure out where in the reLyXing process it crashed. That, in turn, will allow you to write a better bug report, which will allow the developers to fix it more quickly and easily.

Bug reports should be sent to the LyX developers' mailing list. Its address is currently `lyx-devel@lists.lyx.org`. If you are running reLyX on a huge file, please do not send all of the output in your bug report. Just include the last ten or twenty lines of output, along with the piece of the LaTeX file it crashed on. Or, even better, attach a small but complete file which causes the same problem as your original file.

5.4.5.2 Implementation Details:

reLyX makes several "passes" in order to translate a TeX file. On each pass, it creates one or two files.

Pass 0
Before doing anything, read the syntax file (or files).

Pass 1a
Split preamble (anything before a `\begin{document}` command) off the rest of the file. It saves the two pieces in separate files. This is necessary because there may be very strange stuff in a preamble. It also ignores anything after the `\end{document}`, on the assumption that it isn't LaTeX.

Pass 1b
Translate the preamble. Currently, that just means translating the `\documentclass` command and copying the rest exactly into the LyX preamble.

Once you know what class the document is, read the LyX layout file for that class.

Pass 2
"Clean" the TeX file, generating slightly stricter LaTeX. This includes:

- Change, e.g., `x^2` to the equivalent but clearer `x^{2}`
- Removing optional arguments that LyX can't handle (e.g., from `\sqrt`)
- Changing `{\em foo}` to `\emph{foo}`, etc. This is necessary because LyX always writes out the non-local forms anyway. This should very rarely make a difference.

Pass 3
Translate LaTeX text, commands, and environments to LyX.

Pass 4
Put the two pieces back together, and do some final tweaking, to generate the LyX file

If there are any `\input` or `\include` commands, reLyX will loop back to the beginning and translate those. It assumes that the included files are the same class as the main file, and that they have no preamble matter. (If you have an `\input` command in the preamble of a file, the command will be copied exactly into the LaTeX preamble portion of the LyX file, so the included file won't be translated.) So when translating included files, it skips passes 0 and 1.

If reLyX doesn't find a file you wanted to include, it will give a warning, but will continue to translate any files it does find.

5.4.5.3 Layout Files

reLyX reads a LyX layout file to know how to handle LaTeX environments and commands which get translated to LyX layouts. This file will include all "normal" non-math environments (i.e., including quote and itemize, but not tabular, minipage, and some other fancy environments), and commands like `\section` and `\title`. If you want to reLyX a class that doesn't have an existing layout file, then you'll have to create a layout file. But you have to do this anyway, in order to LyX the file, since LyX depends on layout files to know how to display and process its files. Check the LyX documentation for help with this task (which can be hard or easy, depending on the class you want to create a layout file for.) If your class is quite similar to a class that has a layout file, then consider using the **`-c`** option.

5.4.5.4 Syntax Files

reLyX always reads at least one syntax file, called the default syntax file. reLyX will read your personal syntax file if it exists; otherwise it will read the system-wide file. reLyX will read additional syntax files if you specify them with the **`-s`** option. (These extra files should have the same format as the default file, but will tend to be shorter, since they only have to specify extra commands not found in the default file.) A syntax file tells reLyX a few things.

First, it describes the syntax of each command, that is, how many required arguments and how many optional arguments the command takes. Knowing this makes it easier for reLyX to copy (in TeX mode) commands that it doesn't know how to translate. The syntax file simply has a command, followed by braces or brackets describing its arguments in the correct order. For example, a syntax file entry `\bibitem[]{}` means that the `\bibitem` command takes an optional argument followed by a required one, while the entry `\bf` means that the `\bf` command takes no arguments at all. When reLyX encounters a token that it doesn't know how to translate into LyX, it will copy the token—along with the correct number of arguments—exactly. If the token is not in the syntax file, then reLyX just copies as many arguments as it finds. This means that it may copy too much. But since the user can specify additional syntax files, that shouldn't happen often.

Some commands that cannot be translated to LyX, like `\mbox`, have as one of their arguments regular LaTeX text. If the string "translate" is put into an argument of an (untranslatable) command in the syntax file, then reLyX will translate that argument instead of copying it verbatim. So, for example, the default syntax file has `\raisebox{}[][]{translate}`. This means that the `\raisebox` command and the first argument (and optional arguments if they exist) are copied in TeX mode, but the last argument (which may contain math, complicated LaTeX, other untranslatable commands, etc.) will be translated into LyX. You can't use "translate" on optional arguments.

User-defined syntax files are allowed to define new commands and their syntax, or override the number of arguments for a command given in the default syntax file. (E.g., if you're using a style that gives an extra argument to some command...) However, this will only be useful for commands copied in TeX mode. Commands which are actually translated by reLyX (like `\item`) have their argument syntax hard-coded. The hard-coded commands are identified in the default syntax file.

Second, the syntax file describes any "regular environments". Usually, an entire unknown environment will be copied in TeX mode. If you define a regular environment "foo", though, then only the `\begin{foo}` and `\end{foo}` commands will be copied in TeX mode; the text within the environment will be treated (i.e., translated) by reLyX as regular LaTeX, rather than being copied into TeX mode. Don't try to declare `tabbing` and `picture` as regular environments, as the text within those environments will confuse reLyX; use this capability for new environments you create that have plain text or math or simple commands in them. You also can't declare unknown math environments (like `equation*`) as regular environments, either, since the LyX math editor won't understand them. The names of regular environments appear, whitespace-separated, between `\begin{reLyXre}` and `\end{reLyXre}` statements in the syntax file. (If you have a regular environment which you won't use very often, you can use the `-r` option rather than writing a syntax file.)

Third, the syntax file describes a math translation table. The LyX math editor doesn't support a few commands. For example, `_` is supported, but the equivalent `\sb` is not. Put any commands you'd like translate between `\begin{reLyXmt}` and `\end{reLyXmt}` statements. The statement "`\| {\Vert}`" means that any `\|` in math mode will be converted to "`\Vert` " (in cases where a token made up of a backslash and a non-letter is translated to something with letters at the end, a space is added by reLyX. That way, "`\|a`" is correctly translated to "`\Vert a`").

5.4.5.5 Miscellaneous

You need Perl version 5.002 or later to run reLyX. <plug> If you don't have Perl, you should get it anyway (at Perl `http://www.perl.com/`), because it's a really useful tool for pretty much anything. </plug>

5.4.6 Diagnostics

reLyX should always explain why it crashes, if it crashes. Some diagnostics may be very technical, though, if they come from the guts of the code. reLyX gives much more information while running if you use the **-d** option, but you shouldn't need that unless something goes wrong.

When it's finished, reLyX will tell you if it finished successfully or died due to some error.

5.4.7 Warnings

Always keep a copy of your original LaTeX files either under a different name or in a different directory. There are a couple ways in which using LyX could lead to overwriting the original LaTeX file.

If you import `foo.tex` to create `foo.lyx`, then edit `foo.lyx` and want to re-export it, note that it will overwrite the original `foo.tex`. (LyX will *not* ask you if you want to overwrite it.)

If you have chosen not to use a temporary directory in the preferences, then LyX will create its temporary files in your current directory, which means your LaTeX original may be overwritten (without a warning from LyX) when you "view dvi" or print the LyX document.

5.4.8 Files

`MY_LYXDIR/layouts/*.layout`
User's personal layout files for document classes

`MY_LYXDIR/reLyX/syntax.default`
User's personal syntax file

`LIBDIR/layouts/*.layout`
System-wide layout files for document classes

`LIBDIR/reLyX/syntax.default`
System-wide LaTeX syntax file

`LIBDIR` is the system-wide LyX directory, usually something like `/usr/local/share/lyx/`. `MY_LYXDIR` is your personal LyX directory, something like `.lyx/` in your home directory. You can see their actual values in the Help ▷ About LyX dialog.

5.4.9 See also

lyx(1), *latex*(1)

5.4.10 Authors

Copyright (c) 1998–9 Amir Karger (`karger@voth.chem.utah.edu`)

Code contributors:

- John Weiss wrote the original CleanTEX pass.
- Etienne Grossmann
- José Abílio Oliveira Matos
- David Suarez de Lis

Other contributors:

- Jean-Marc Lasgouttes worked on the wrapper script and offered lots of bug reports, advice, and feature suggestions.
- Asger K. Alstrup Nielsen and Marc Pavese provided advice.
- Various members of the LYX developers' and users' lists provided bug reports and feature suggestions.

reLYX uses a modified version the Perl TEX parser `Text::TeX` package written by Ilya Zakharevich (`ilya@math.ohio-state.edu`), available on CPAN.

Chapter 6

LyX Features needing Extra Software

6.1 Using LyX with SGML-Tools (aka LinuxDoc)

by Paul Evans

6.1.1 Overview

LinuxDoc is a document class available in LyX if you have the `sgml-tools` package installed. You can use it to produce documents in the so-called Standardized General Mark-up Language (SGML) in the particular format used by the Linux Documentation Project. That is obviously helpful if you are contributing to that project. You can use the SGML format with the `sgml-tools` package of scripts and programs (to produce other formats, including Latex, HTML, plain text, man pages and...). You may therefore prefer to use this document class if you want to write something that can be easily translated into other formats.

You will find that LinuxDoc has fewer layout options than the other text classes in LyX. This is mainly so that the translations into other formats have a chance of making some sense. In this section we describe:

- how to setup and use a document in LinuxDoc
- how to use the tags in LinuxDoc to layout your document
- how to use the SGML packages to produce the various formats
- how to sort out some problems.

6.1.2 Preparing and using a LinuxDoc document

6.1.2.1 Getting started

You start by selecting the LinuxDoc class using the Document ▷ Settings dialog. Then you will find that there are fewer paragraph environments than for most other classes. You can see them on the pull down box on the left of the tool bar. How to use them is described in section 6.1.3.2.

You *must* enter a title for the document, followed by an author, marking each with the appropriate paragraph environment. If you don't do this, you will get errors when you try to print the file. You can then enter the date and an abstract. The document proper must start with a Section paragraph environment rather than any standard layout.

After that you can prepare a document as usual using the available range of paragraph environments. See section 6.1.3.2 for the full list and their uses.

6.1.2.2 Output from LinuxDoc

You can print and save these documents in the normal way. To use the other features of the SGML package you need to save your document as LinuxDoc; this is a version in which the document is translated into the basic sgml tags. Use File ▷ Export ▷ LinuxDoc. You will get a file with the same name and a `.sgml` extension rather than a `.lyx` extension. See 6.1.4 on how you than make use of this file.

6.1.3 Using the paragraph environments in LinuxDoc

6.1.3.1 The Structure of a LinuxDoc Document

There is a formal structure for LinuxDoc which limits how you can place tags. There are two parts to all documents:

Header: this is everything up to the first time you insert a Section layout marker. It can include title, author, date, abstract and ToC. You must include the first two.

Body: from the beginning of the first section onwards. All other tags are allowed.

6.1.3.2 The LinuxDoc Paragraph Environments

Here is a list of all the tags you will find listed on the layout bar in the order they come there, with some comments where the purpose or use is not obvious:

- Standard: works as described in [cross reference]
- Title: This will appear at the top left of the document when printed, above a heavy horizontal rule, although you will not see this on the LYX screen.

- Section, Subsection, Subsubsection, Paragraph and Subparagraph: all do what you would expect and in the usual order. Whether they are numbered or not is controlled by the Section number depth setting. You cannot get the equivalent number free versions in any other way; there is no Section* or similar
- Enumerate: As usual this produces a numbered and indented list as described in the *User's Guide*.
- Itemize: Again much the same as in the other classes: see the *User's Guide*.
- Description: As explained in the *User's Guide*. Remember that if you want the bold element at the start of a description to be more than one word then you need to put protected spaces between the words.
- Verbatim: As usual.
- Code: similar to the Lyx-Code environment
- Author: Anything you mark with this will appear on the left of the heading of the document, under the heavy rule.
- Date: Anything you mark with this will appear on the right of the heading under the rule. You do not have to make this a date. Any text can be entered, e. g. a version number.
- Abstract: You can use this to produce a free standing paragraph after the author and date, and before the first section. You are only allowed one such paragraph.[1]
- Displaymath:[2]

6.1.3.3 Other document features

You can also use the Layout menu to set fonts or to emphasis words. You can also use the table of contents as usual; see the corresponding section of the *User's Guide*. Although you will find some some other features on the menus e. g. inserting footnotes. There is some doubt about whether these will work correctly.[3]

6.1.3.4 Cross references and HTML

On the Insert menu you will find two new options relating to the inclusion of URL addresses. If you use either option you will find some highlighted TEX code inserted into your document in three separate blocks with spaces available between. The blocks will be:

[1] *Author's note.* This needs checking —*pe.*
[2] *Author's note:* I have not yet checked this —*pe.*
[3] *Author's note:* Again still checking to see whether this is my system —*pe.*

`\htmlurl{` or `\url{` space `}{` space `}`

You insert a full HTML tag between the first and second blocks. This can be `http://any.address` or other valid tags such as `mailto:me@my.address.` Then you insert some description between the second and third blocks. The differences are:

- URL: both the HTML tag and the description will appear in the document
- HTML URL: only the description appears in the printed version

6.1.4 Using the LinuxDoc Sgml scripts

You can use LinuxDoc as a text class without any additional scripts or programs, but there is not much point in doing this. All you will get is a document that looks like a *Linux Documentation Project Howto*. To do the document translation you need to get and install the `sgml-tools-1.0.x.tar.gz` (with $x \geq 3$) package from the SGML-Tools WWW Page at

```
http://pobox.com/~cg/sgmltools
```

Alternatively, you can go to the `sunsite` archive at[4]

```
ftp://sunsite.unc.edu/pub/Linux/utils/text/sgml-tools-1.
0.x.tar.gz
```

The file `sgml-tools-1.0.x.tar.gz` contains everything that you need to write SGML documents and convert them to groff, LaTeX, HTML, GNU info, LyX, and RTF.

This package was renamed from `linuxdoc-sgml-1.5.tar.gz` in January 1997.

Follow the instructions in that package on how to install it and how to use it. All this has to be done outside of LyX, before you can use the File ▷ Export ▷ as LinuxDoc option.

6.1.5 Troubleshooting LinuxDoc

When you print or preview a LinuxDoc document some checking is done of the tags before LaTeX is run. Some errors are trapped here, especially those concerning the structure of the document. LyX may produce an error message, but not leave an error box in the document for you to open. You may have to look at the files directly to discover what is wrong. Most problems seem to come from the use of options that are not fully available in the text class.

[4]Note that, at the time of this writing (01/1998), version 1.0.3 of sgml-tools has not yet been made available at `sunsite`.

6.2 Checking TEX

by ASGER ALSTRUP

6.2.1 Introduction

Under the Tools menu, you'll find a Check TEX command. This feature requires you to have the `chktex` program installed, and is grayed out if you don't have it. You can get it from your nearest CTAN mirror, or over the Web from `http://www.ifi.uio.no/~jensthi/chktex/`.

The ChkTEX package is a program that was written by JENS T. BERGER THIELEMANN in frustration because some constructs in LATEX are sometimes non-intuitive, and easy to forget. The program runs over your LATEX file and checks the integrity of the file, and flags some common errors. In other technical words, it is `Lint` for LATEX.

Well, what is a syntax checker doing in LYX which is supposed to produce correct LATEX anyways? The answer is simple: Just as `Lint` not only checks the *syntax* of C programs, but also does *semantic* checks for type-errors, ChkTEX catches some common *typographic* errors, in addition to the syntactical ones. Specifically, ChkTEX is capable of detecting several common errors, such as

- Ellipsis detection:
  Use . . . instead of ...
- No space in front of/after parenthesis:
  (wrong spacing)
- Enforcement of normal space after common abbreviations:
  e. g. is too wide spacing.
- Enforcement of end-of-sentence space when the last sentence ends with a capital letter:
  This is a TEST. And this is wrong spacing.
- Space in front of labels and similar commands:
  The label should stick right up to the text to avoid falling to a wrong page. [5] The label is separated too much.
- Space in front of references, instead of hard spaces:
  In you are in bad luck, the text will break right between the referenced text and reference number, and that's a pity. See section 6.2.1.
- Use of "x" instead of $\times$ between numbers:
  2x2 looks cheap compared to 2×2.

and more . . . It is an invaluable tool when you are "finishing up" your document before printing, and you should run it right after the obligatory spelling check, and before you go fine tuning the typesetting.

[5]This footnote is in danger of falling off to a wrong page

6.2.2 How to use it

If you have the program installed, usage is as simple as choosing Tools ▷ Check TeX. This will make LyX generate a LaTeX file of your document, start ChkTeX to check it, and then make LyX insert "error boxes" with the warnings from ChkTeX, if there were any. The warnings will be placed close to the point of the mistake, and you can quickly find them by using the Navigate ▷ Error menu item, or the shortcut key C-g from the default `cua` bind file. Open the error boxes by clicking on them with the mouse, or use the shortcut key C-i from `cua` bindings, or the corresponding C-o for the alternate `emacs` bind file. Read the warning and correct the mistake, if it is a mistake. If you have trouble understanding what the warning is about, you can safely ignore it. Remember that there is a hidden layer between the document on screen and the technical details in invoking ChkTeX, and this gap can make some warnings seem arcane or just right down plain silly.

This document is an excellent testing bed for the feature, and it should provide quite a few warnings for you to fiddle with. Since computers are only so smart, expect most of the warnings to be false alarms, though.

6.2.3 How to fine tune it

Sometimes, you'll find that ChkTeX makes more noise than suits your mood. Then you can choose not to use it, wait until your mood changes, or try to customize ChkTeX to get better along with you. Another choice in the most desperate situations is to use View ▷ Remove All Error Boxes, which will get rid of all warnings instantly.

Although ChkTeX *is* very configurable and extensible, you shouldn't expect to solve all problems with ChkTeX in LyX this way. Since LyX has to generate a somewhat special LaTeX file to be able to match the line numbers from the ChkTeX output[6] to the internal document structure, some of the warnings will not seen to appear correctly. There are two things you can do about this:

- Fine tune the ChkTeX invocation command line in Preferences (tabs Outputs, Misc), or the global ChkTeX installation configuration file (usually with the file `/usr/local/share/chktexrc`). See below to learn what warnings can be enabled and disabled on the command line.

- Export your document as a raw LaTeX file using File ▷ Export ▷ LaTeX and run `chktex` manually on that. Invoked in this way, it can be a hassle to find the corresponding place in the document inside LyX, but with a little patience, you should be able to do it.

Here follows the warning messages that can be enabled and disabled in Preferences. Use `-n#` to disable a warning, and `-w#` to enable a warning. The

[6] You can inspect the specific output from chktex by using Edit ▷ View LaTeX Log right after a chktex run.

emphasized entries are disabled by default, because the default is `"chktex -n1 -n3 -n6 -n9 -n22 -n25 -n30 -n38"`.

Notice that you should only use the options that enable and disable warnings, because LYX relies on some of the other command line parameters to be set in a specific way to have a chance to communicate with `chktex`.

1. *Command terminated with space.*
2. Non-breaking space (“`~`”) should have been used.
3. *You should enclose the previous parenthesis with “`{}`”.*
4. Italic correction (“`\/`”) found in non-italic buffer.
5. Italic correction (“`\/`”) found more than once.
6. *No italic correction (“`\/`”) found.*
7. Accent command “`cmd`” needs use of “`cmd`”.
8. Wrong length of dash may have been used.
9. *“`%s`” expected, found “`%s`”.*
10. Solo “`%s`” found.
11. You should use “`%s`” to achieve an ellipsis.
12. Inter-word spacing (“\ “) should perhaps be used.
13. Inter-sentence spacing (“`\@`”) should perhaps be used.
14. Could not find argument for command.
15. No match found for “`%s`”.
16. Math mode still on at end of LATEX file.
17. Number of “`char`” doesn’t match the number of “`char`”.
18. You should use either `‘‘` or `’’` as an alternative to “`"`”.
19. You should use `"’"` (ASCII 39) instead of `"´"` (ASCII 180).
20. User-specified pattern found.
21. This command might not be intended.
22. *Comment displayed.*
23. Either `’’\,’` or `’\,’’` will look better.
24. Delete this space to maintain correct page references.
25. *You might wish to put this between a pair of “`{}`”.*
26. You ought to remove spaces in front of punctuation.

27. Could not execute LaTeX command.
28. Don't use `\/` in front of small punctuation.
29. `$\times$` may look prettier here.
30. *Multiple spaces detected in output.*
31. This text may be ignored.
32. Use `‘‘` to begin quotation, not `’`.
33. Use `’` to end quotation, not `‘‘`.
34. Don't mix quotes.
35. You should perhaps use "`cmd`" instead.
36. You should put a space in front of/after parenthesis.
37. You should avoid spaces in front of/after parenthesis.
38. *You should not use punctuation in front of/after quotes.*
39. Double space found.
40. You should put punctuation outside inner/inside display math mode.
41. You ought to not use primitive TeX in LaTeX code.
42. You should remove spaces in front of "`%s`"
43. "`%s`" is normally not followed by "`%c`".

In later versions of LyX, we hope to provide a more complete interface to this tool (and it's smaller cousin `lacheck`) to exploit the full power of it. But it's not exactly useless as it is now: go try it on one of your existing documents of a certain length and be surprised.

6.3 Version Control in LyX

by Lars Gullik Bjønnes

6.3.1 Introduction

A friend of mine wanted to try LyX for a group project. When he didn't find support for version control or file locking, he dropped it. This angered me a bit, so I thought that I should at least make support for RCS (with the possibility of CVS and/or SCCS as a future improvement.) This has now been done. LyX now supports some of the most basic RCS commands. If you need to something a bit more sophisticated you will have to do that manually in an xterm.

Before you begin to use the version control features in LyX, you should read "rcsintro" (a man file, read it with `man rcsintro`). This file describes all the basic features of RCS. You should especially notice the comment about a RCS directory, and the notion of a master RCS file (the file ending in `,v`).

The implementation in LYX assumes a recent version of the GNU RCS package—no guarantees are made for older versions.

6.3.2 RCS commands in LYX

The following sections describe the RCS commands supported by LYX. You can find them in the File ▷ Version Control submenu.

6.3.2.1 Register

If your document is not under revision control, this is the only item shown in the menu. And if it is under revision control, the Register item is grayed out.

This command registers your document with RCS. You are asked interactively to supply an initial description of the document. The document is now set in Read-Only mode and you have to Check Out For Edit, before making any changes to it. A document under revision control has a "[RCS:<version> <locker>]" item tagged to the filename in the minibuffer.

RCS command that is run: `ci -q -u -i -t-"<initial description>" <file-name>`

Read `man ci` to understand the switches.

6.3.2.2 Check In Changes

When you are finished editing a file, you check in your changes. When you do this, you are asked for a description of the changes. This is stored in the history log. The version number is bumped, your changes are applied to the master RCS file, the document is unlocked and set to Read-Only mode.

RCS command: `ci -q -u -m"<description>" <file-name>`

6.3.2.3 Check Out For Edit

By doing this you lock the document so that only you can edit it. This will also make the document Read-Write only for you. You will usually continue editing for a while and when you are finished you check in your changes. The status line is changed to reflect that you have locked the file.

RCS command: `co -q -l <file-name>`

6.3.2.4 Revert To Last Version

This will discard all changes made to the document since the last check in. You get a warning before changes are discarded.

RCS command: `co -f -u<version> <file-name>`

6.3.2.5 Undo Last Checkin

This makes as if the last check in never happened. No changes are made to the document loaded into LyX, but the last version is removed from the master RCS file.

RCS command: `rcs -o<version> <file-name>`

6.3.2.6 Show History

This show the complete history of the RCS document. The output of `rlog <file-name>` is shown in a browser. See `man rlog` for more info.

6.4 Literate Programming

Updated by Kayvan Sylvan (kayvan@sylvan.com), original documentation written by Edmar Wienskoski Jr. (edmar-w-jr@technologist.com)

6.4.1 Introduction

The main purpose of this documentation is to show you how to use LyX for literate programming. Where it is assumed that you are familiar with this programming technique, and know what "tangling" and "weaving" means. If that is not the case, please follow the web links provided in the following sections. There is a lot of good documentation out there covering old development history to the latest tools tips.

It is also assumed that you are familiar with LyX itself to a point that you are comfortable changing your LyX preferences, and X resources file. If that is not the case please refer to other LyX documentation to cover your specific needs.

6.4.2 Literate Programming

From the Literate Programming FAQ:

> Literate programming is the combination of documentation and source together in a fashion suited for reading by human beings. In fact, literate programs should be enjoyable reading, even inviting! (Sorry Bob, I couldn't resist!) In general, literate programs combine source and documentation in a single file. Literate programming tools then parse the file to produce either readable documentation or compilable source. The WEB style of literate programming was created by D.E. Knuth during the development of his TeX typesetting software.

Another excerpt says:

> *How is literate programming different from verbose commenting?*
>
> There are three distinguishing characteristics. In order of importance, they are:
>
> - flexible order of elaboration
> - automatic support for browsing
> - typeset documentation, especially diagrams and mathematics

Now that I sparked your curiosity, take a look in the references.

6.4.2.1 References

The complete Literate Programming FAQ can be found at:

> Literate Programming FAQ `http://shelob.ce.ttu.edu/daves/lpfaq/faq.html`

The FAQ lists 23 (twenty three!) different literate programming tools. Where some are specialized or "tailored" for particular programming languages, while other have general scope. I selected NOWEB for my own use for several reasons:

- It can generate the documentation either in latex or html.
- It has a open architecture, i.e., it is easy to plug in new filters and to perform special processing that you may need.
- There is a good selection of filters available already (the html is one of them).
- It is free.

The Noweb web page can be found at:

> Noweb home page `http://www.cs.virginia.edu/~nr/noweb/`

Starting from there you can reach many other interesting links and even some literate program examples.

6.4.3 LyX and Literate Programming

The LyX support for Literate Programming is provided by using the generic LyX convertors mechanism. This support is provided in a "Noweb independent" way, i.e., you will be able to use this new LyX feature with some other literate programming tool of your choice by just changing your LyX preferences.

6.4.3.1 Generating documents and code (weaving and tangling)

Selecting the document class If you have installed Noweb and LyX successfully, whenever you open a new document or try to change the document class of an existing one, you will find that there are three new document classes available:

- Article (Noweb)
- Book (Noweb)
- Report (Noweb)

You must select one of them to create your literate documents from.

Note that literate documents are not limited to these three classes. New classes can be generated from other styles like letter or in combination with other class variations like Article (AMS). If you have special needs that cannot be covered by one of the existing classes, let the LyX developers list (lyx-devel@lists.lyx.org) know and we will arrange to insert a new entry, or teach you how to do it.[7] Moreover, if you use a literate tool other than Noweb you may need to create a new set of document classes for it.

Typing code in LyX enables you to write code with a layout named Scrap.[8] Noweb delimits scraps like this:

```
<<My scrap>>=
  code
  more code
  even more code
  @
```

The problem is that whatever is written in between the << and the `@` must be taken literally, i.e., LyX should be prevented from making any special interpretation of what has been written. This is handled by a special layout named Scrap, that works like a normal paragraph but has a free spacing capability.

The down side of the Scrap paragraph layout is that consecutive paragraphs of code will be spaced with one empty line in the source code and also in the printed documentation. The work around is to enter each line of code within a single Scrap, with a newline (ctrl-return). The example above will look like this:[9]

[7] It is very simple, it involves the creation of a file with four lines, and re-running of the auto configuration.

[8] The equivalent Noweb term is "Chunk". For historical reasons, I got used to the term "scrap" introduced by other literate tool named Nuweb, which I used for many years before rendering myself to Noweb.

[9] If you have a printed version of this document you will not see any difference between the previous example and this one.

```
<<My scrap>>=
  code
  more code
  even more code
  @
```

This layout works fine. The only real inconvenience is that you have to type ctrl-return instead of a plain return.[10]

As a special note, you can also use the "%def" construct of Noweb in your scraps to add items to Noweb's identifier cross-reference:

```
<<My scrap>>=
  def some_function(args):
    "This is the doc string for this function."
    print "My args: ", args
@ %def some_function
```

For an example of this usage and the resulting cross-reference output, look at the Literate python program in *LIBDIR/examples/listerrors.lyx* which should make this all clear.

Generating the documentation At this point you already have a new document file with a proper document class, and with some code and text on it. How do I print it? The answer is simple, you select View ▷ DVI, etc. Just like you would do for a plain document. No special procedure is required.

To help orientate you, I will now explain what happens inside LyX:

1. When the Update ▷ DVI menu option is chosen, a latex file is generated.

   If the document is of any literate class the generated file will be named with an extension name defined by the "literate" format (defined in the Preferences panel), otherwise the file will have the usual `.tex` extension.

2. Note that the only difference so far is in the name of the file, no special processing is required by LyX. Given that you formatted the code using the Scrap layout that, by itself, takes care of the business.

3. If the document is of any literate class LyX will then use the internal LyX to Noweb converter, followed by the Noweb to LaTeX converter[11] to generate the LaTeX file.

   Otherwise it will just skip this step.

4. Finally, LaTeX is invoked and the regular post processing continues as in a plain document.

Independence from a particular "literate tool" is easily achieved by changing the commands that are run by the various converters.

[10]It is in my list of "improvements" to fix that.

[11]The converters are defined in the Tools ▷ Preferences panel, under the "Conversion" tab.

Generating the code When the build menu option is chosen or the corresponding button in the toolbar is pressed, a latex file is generated just like step 1 above. Next, LyX invokes the `Noweb->Program` converter. Typically, this converter (like any other converter), has two parts:

1. The converter program itself. This program performs the conversion from the one format to the other (in this case, from the Noweb format to the Program pseudo-format).
2. The error log parser. This is a program whose sole purpose is to rewrite error messages in a format that LyX understands. This makes it possible for LyX to place error boxes in the right places in the file buffer.

The first part, the "Converter" setting, should be set to "`build-script $$i`". This basically means that LyX will call "build-script" (a program or script) with the name of the Noweb file (generally a file in the LyX temp directory).

This is an implementation of "build-script" that you can place in a directory on your path:

```
#!/bin/sh
#
notangle -Rbuild-script $1 | env NOWEB_SOURCE=$1 sh
```

The next part of the converter setting is the "Flags" which is to be set to "`originaldir,parselog=listerrors`". This will run any errors that are generated by the "build-script" process through the "listerrors" program.

The converter code looks in *MYLYXDIR/scripts* first, then in *LIBDIR/scripts* then on the path for the "listerrors" program.

Build instructions in the document The last piece of the integration between LyX and noweb is the "build-script" scrap. Generally, the instructions for building your program should be embedded in a scrap of its own. The noweb-specific "build-script" above uses the notangle command to look for this scrap (called "build-script") and runs its contents through "sh".

Typically, such a scrap would look something like this:

```
<<build-script>>=
#!/bin/sh

if [ -z "${NOWEB_SOURCE}" ]
then
  NOWEB_SOURCE=myfile.nw
fi
[... code to extract files ...]
[... code to compile files ...]
@
```

Look in *LIBDIR/examples/listerrors.lyx* or in *LIBDIR/examples/Literate.lyx* which implement two versions of the "listerrors" program for some illustrations of how all of these pieces go together or in *LIBDIR/examples/noweb2lyx.lyx.* Interestingly, these three files show off the language-indepence of the LyX literate programming support since they are written in Python, C and Perl respectively.

6.4.3.2 Configuring LyX

All the Literate Programming support is configured by the Tools▷Preferences panel in the "Conversion" tab. The important parts are:

the "literate" format Set up via the Formats tab, this is where the Noweb-specific pieces are set up. The GUI Name is set to `NoWeb`, the file extension is set to `.nw`. This tells LyX to create a file with a `.nw` extension in the first step of the conversion process.

the Program format This is an empty format whose sole purpose is to be the endpoint of a conversion (which then allows us to set up a converter for it).

NoWeb->LaTeX This converter performs the "weaving" of the literate document. For Noweb, it is set to "`noweave -delay -index $$i > $$o`"

NoWeb->Program This performs the "tangling step". As stated above, the Converter is set to "`build-script $$i`", with Flags set to "`originaldir,parselog=listerrors`".

6.4.3.3 Debug extensions

There is also a new function implemented in the LyX server, the "server-goto-file-row" function, to be used with ddd/gdb or other debugger.

When debugging code with ddd/gdb, it is possible to invoke a text editor at the current execution position with a single key stroke. The default ddd configuration for that is shift-ctrl-V. It happens that you can define the editor command line invocation in ddd by accessing the Edit▷Preferences▷Helpers dialog and changing the "Edit Sources" entry.

I take advantage of the new created LyX server function and this ddd feature, and set "Edit Sources" to:

```
echo "LYXCMD:monitor:server-goto-file-row:@FILE@ @LINE@" >~/.lyxpipe.in
```

With this, whenever you are using ddd and find a point in the program that you want to edit, you just press shift-ctrl-V (in the ddd window), and ddd you forward this information to LyX through the LyX server and then the LyX window will show the same file with the cursor at the same position ddd was pointing to. No more guessing or long scrolling to locate a point in the program back from debugging !

Note however that you must enable the LyX server to get this feature working (it is disabled by default). You can enable it in Preferences (tabs Inputs, Paths) by entering in the LyXserver pipe a path like "`/home/<your-home-directory>/.lyx/lyxpipe`"

Read the LyX server documentation in the *Customization Manual* for further information.

6.4.3.4 Toolbar extensions

There are six new buttons that can be added to your LyX toolbar. Five of these buttons are short cuts to layout styles: Standard, Section, LaTeX, LyX-Code, and Scrap. The last one is a short cut to the "Build Program" File menu entry.

LyX has a range of buttons that are available for tool bar customization. In my toolbar I like to combine the six short cuts above with two more: One for View ▷ Update ▷ DVI and the other for View ▷ DVI File menu entries. Here is how it looks like:

```
Toolbar
  Layouts
  Icon "layout Standard"
  Icon "layout Section"
  Icon "layout LaTeX"
  Icon "layout LyX-Code"
  Icon "layout Scrap"
  Separator
  Icon "buffer-view"
  Icon "buffer-typeset"
  Icon "build-program"
  Separator
.
.
.
End
```

6.4.3.5 Colors customization

There are a number of colors in LyX that can be customized in Preferences. One of the things that bothers people is the LaTeX font color. The default color is red, since the scraps uses LaTeX font, and there is a lot of scraps in literate documents, you may get tired of seeing everything in red. You can change it by going to the tabs Look&Feel, Colors.

The next thing is the visible presence of the newline character in the screen. You can choose the color of this particular character and make it blend in the background. I recommend you choosing a color that is close to the background but not equal, that way you still can see it is there, but it is not bothering you anymore.

Chapter 7

Secrets of the LaTeX Masters

Though LyX is a powerful tool, it cannot hope to support everything that can be done with pure TeX/LaTeX. However, many familiar dirty TeX and LaTeX tricks can be done within LyX, as long as you are not afraid to use that "TeX" button on the toolbar or add things to the LaTeX preamble. This section lists some tips, tricks, and otherwise cool ideas to give your document that extra little flair. *Do try this at home*, just start with something a little smaller and less important than your dissertation!

Most ideas in this section require less common files in your LaTeX installation. If you have a system like teTeX, most will already be available. A few, however, will need to be downloaded from one of the CTAN archives. Often, there are several ways to do something, or several LaTeX style files which do the same thing. We do not endorse one choice over another, we simply claim that we have done a particular task with a particular file. Put on your wizard hat, keep an eye out for dragons, and let us begin.

7.1 Tricks for Footnotes and Margin Notes

suggested by Robin Socha

7.1.1 Footnotes

LyX cannot yet take care of setting the footnote numbering back to 1 after each section in the "article" document class or changing the counter style. You'll need to insert LaTeX commands like the following to achieve that:

Using `\setcounter{footnote}{0}` will set the counter back to 1[1].

The following command will change the numbering to small letters. Take a look at the next footnote in your xdvi or ghostview :[b]

[1]The counter has been set back to 1.

[b]This is an example for a footnote with alphabetic numbering.
Use `\renewcommand{\thefootnote {\alph{footnote}}` to get this.

The next command sets the counter style back to default, i.e. `\arabic`[c].

You can use `\arabic`, `\roman`, `\Roman`, `\alph` or `\Alph` and others as counter styles. Just replace the LaTeX command in the above example and rerun TeX to see what those styles can do.

7.1.2 Margin Notes

Here are two examples of neat things you can do to margin notes using LaTeX commands.

The following command will make a vertical line appear alongside your text—great for "thumbing": `\marginpar{\rule[-10mm]{30mm}{5mm}}`.

Check your dvi- or ghostview-output to see what the `\reversemarginpar` command does to the following margin note.

This is a margin note.

7.2 Multiple Columns

by Lars Gullik Bjønnes

7.2.1 Purpose

The aim for this chapter[d] is to show how the LaTeX package `multicol` can be used in a LyX document. As LyX doesn't support the `multicol` package natively yet, we have to use some small hacks. By reading this section it should be obvious how to do this.

7.2.2 Limitations

The `multicol` package allows switching between one and multicolumn format on the same page. Footnotes are handled correctly (for the most part), but will be placed at the bottom of the page and not under each column. LaTeX's float mechanism, however, is partly disabled in the current implementation. At the moment only page-wide floats can be used within the scope of the environment.

7.2.3 Examples

7.2.3.1 Two columns

If you want to have two columns in your text, you have use LaTeX mode to insert `\begin{multicols}{2}` at the point where you want the two column layout to start, and then `\end{multicols}` where you want it to end. Like this:

[c]Use `\renewcommand{\thefootnote}{\arabic{footnote}}` to set the counter–style back to LyX's default, i.e. \arabic.

[d]Editor's note: Lars' original chapter was a masterful description of how to use the `multicol` package. However, it was too long to flow smoothly in this document. I have therefore chosen to excerpt the most important sections here (sorry, Lars); you can read the original chapter (and more of the story!) in the example file `examples/multicol.lyx`. — mer

The Adventure of the Empty House
by Sir Arthur Conan Doyle

It was in the spring of the year 1894 that all London was interested, and the fashionable world dismayed, by the murder of the Honourable Ronald Adair under most unusual and inexplicable circumstances. The public has already learned those particulars of the crime which came out in the police investigation, but a good deal was suppressed upon that occasion, since the case for the prosecution was so overwhelmingly strong that it was not necessary to bring forward all the facts. Only now, at the end of nearly ten years, am I allowed to supply those missing links which make up the whole of that remarkable chain. The crime was of interest in itself, but that interest was as nothing to me compared to the inconceivable sequel, which afforded me the greatest shock and surprise of any event in my adventurous life. Even now, after this long interval, I find myself thrilling as I think of it, and feeling once more that sudden flood of joy, amazement, and incredulity which utterly submerged my mind. Let me say to that public, which has shown some interest in those glimpses which I have occasionally given them of the thoughts and actions of a very remarkable man, that they are not to blame me if I have not shared my knowledge with them, for I should have considered it my first duty to do so, had I not been barred by a positive prohibition from his own lips, which was only withdrawn upon the third of last month.

7.2.3.2 Multiple columns

The same pattern is used when you want more than two columns:

It can be imagined that my close intimacy with Sherlock Holmes had interested me deeply in crime, and that after his disappearance I never failed to read with care the various problems which came before the public. And I even attempted, more than once, for my own private satisfaction, to employ his methods in their solution, though with indifferent success. There was none, however, which appealed to me like this tragedy of Ronald Adair. As I read the evidence at the inquest, which led up to a verdict of willful murder against some person or persons unknown, I realized more clearly than I had ever done the loss which the community had sustained by the death of Sherlock Holmes. There were points about this strange business which would, I was sure, have specially appealed to him, and the efforts of the police would have been supplemented, or more probably anticipated, by the trained observation and the alert mind of the first criminal agent in Europe. All day, as I drove upon my round, I turned over the case in my mind and found no explanation which appeared to me to be adequate. At the risk of telling a twice-told tale, I will recapitulate the facts as they were known to the public at the conclusion of the inquest.

You can have have more than 3 columns if you want to, but that might not be very pleasant for the eye.

7.2.3.3 Columns inside columns

You can even have columns inside columns:

The Honourable Ronald Adair was the second son of the Earl of Maynooth, at that time governor of one of the Australian colonies. Adair's mother had returned from Australia to undergo the operation for cataract, and she, her son Ronald, and her daughter Hilda were living together at 427 Park Lane.

The youth moved in the best society–had, so far as was known, no enemies and no particular vices. He had been engaged to Miss Edith Woodley, of Carstairs, but the engagement had been broken off by mutual consent some months before, and there was no sign that it had left any very profound feeling behind it. For the rest {sic} the man's life moved in a narrow and conventional circle, for his habits were quiet and his nature unemotional. Yet it was upon this easy-going young aristocrat that death came, in most strange and unexpected form, between the hours of ten and eleven-twenty on the night of March 30, 1894.

Ronald Adair was fond of cards–playing continually, but never for such stakes as would hurt him. He was a member of the Baldwin, the Cavendish, and the Bagatelle card clubs. It was shown that, after dinner on the day of his death, he had played a rubber of whist at the latter club. He had also played there in the afternoon. The evidence of those who had played with him– Mr. Murray, Sir John Hardy, and Colonel Moran–showed that the game was whist, and that there was a fairly equal fall of the cards. Adair might have lost five pounds, but not more. His fortune was a considerable one, and such a loss could not in any way affect him. He had played nearly every day at one club or other, but he was a cautious player, and usually rose a winner. It came out in evidence that, in partnership with Colonel Moran, he had actually won as much as four hundred and twenty pounds in a sitting, some weeks before, from Godfrey Milner and Lord Balmoral. So much for his recent history as it came out at the inquest.

Please do read the file `examples/multicol.lyx` for more advanced examples including column and header spacing, vertical separator lines, and more.

7.3 Numbering in the Enumerate Paragraph Environment

by John Weiss

The default numbering for the Enumerate paragraph environment begins with Arabic numbers and ends with uppercase letters. Suppose, however, you wanted a different type of numbering scheme. Here's a quickie example of how to change the numbering scheme:

```
\renewcommand{\labelenumi}{\Roman{enumi}.}
\renewcommand{\labelenumii}{\Alph{enumii}.}
\renewcommand{\labelenumiii}{\arabic{enumiii}.}
\renewcommand{\labelenumiv}{\alph{enumiv}.)}
```

. . . which changes the numbering scheme to uppercase Roman numerals, uppercase letters, Arabic numbers, and lowercase letter.

Additionally, the previous example also adds a little bit extra to the numbering scheme. For example, the first level label actually looks like: "I.". For ease of reading, we'll describe what the numbering schemes look like using a notation something like this: <"I.", "A.", "1.", "a.)">.

As you can see in the example, there is a label command for each nesting level, `\labelenumi` . . . `\labelenumiv`, as well as a counter, `enumi` . . . `enumiv`. There are also five "number printing" commands, `\arabic{}`, `\roman{}`, `\Roman{}`,

`\alph{}`, and `\Alph{}`, each of which take one counter as an argument. You can add characters before or after these, but there's no need to add spaces.

You can get really fancy with these. For example:

```
\renewcommand{\labelenumi}{\#\Alph{enumi}\#}
\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}
\renewcommand{\labelenumiii}{\alph{enumiii}+}
\renewcommand{\labelenumiv}{(\roman{enumiv})}
```

produces the somewhat out of hand numbering scheme: <"#A#", "A.1", "a+", "(i)">.

7.4 Extra Space Between Table Rows

by Mike Ressler

LaTeX allows you to put a bit of extra space between rows in a table by giving an optional argument to the end-of-row specifier (`\\`). LyX has not yet implemented this in a formal way, so here are two dirty little tricks to do the same job.

The first is the more formal, but longwinded way to do it. In the LaTeX preamble, add the following command definition:

`\newcommand{\extratablespace}[1]{\noalign{vskip#1}}` This command takes a single argument—the amount of space you would like to insert. Insert the command in the first column of the row *after* where you would like the space to appear. Here is an example (I've removed all the borders using Layout ▷ Table):

Minerals	Calcite	Dolomite
	Quartz	Graphite
Rocks	Limestone	Sandstone
	Granite	Andesite

The second method is faster, but will make typographers and TeXperts all over the world groan. Simply put an end of row specifier with optional argument at the same spot. No fancy definitions are needed as in the above example, but there will be more space inserted than you specified because you essentially added a blank row plus the extra space. If the space added is too much, simply use a negative number, like so:

Minerals	Calcite	Dolomite
	Quartz	Graphite
Rocks	Limestone	Sandstone
	Granite	Andesite

It's short, sweet, and gets the job done quickly, even if it is really ugly. You may put away the rotten vegetables now! I promise I won't suggest anything else like that!

7.5 Dropped Capitals

by Mike Ressler

Those of you who like the style of old books probably also like "dropped capitals"—those large capital letters which begin each new chapter or section. Implementing them with plain LyX/LaTeX is straightforward (assuming you know some plain TeX!) but does require a lot of work and many iterations, as you can see by all the ugly TeX-mode stuff at the beginning of this paragraph.

`\bigdrop{-1em}{3}{ptmri}{T}`here is a much easier way of doing this, of course. The `dropcaps` (or the newer `dropping`) package from CTAN allows a simple way to add such letters to your documents. Since this package is not a standard part of teTeX, I can't demonstrate it within this document, but if you copy this paragraph to a new document, delete the "`\verb`" and the pluses from the TeX code at the beginning of the paragraph, and add `\usepackage{dropcaps}` to your LaTeX preamble, you will get a nice Times Roman Italic "T", whose height is three lines of text and which protrudes 1 em into the margin. (Make certain you have copied "`dropcaps.sty`" into a directory where TeX can see it.) The first argument is the amount of indentation; in this case the negative sign moves it into the margin. The second argument is the height of the letter in number of lines of text. The third argument is the font name: virtually anything which has a tfm file should work (wade through the `.../texmf/fonts/tfm` directory for possibilities). My personal favorite is "`yinit`", a fancy German font specifically designed for dropped capitals. The fourth argument is the letter (or letters) to be dropped. The `dropping` package also offers the `\bigdrop` command, as well as a slightly simplified `\dropping` command.

7.6 Non-standard Paragraph Shapes

by Mike Ressler

There are times when the tyranny of rectangular paragraphs must be overthrown. In such situations, a call to the delightful plain TeX command `\parshape` is called for. As you can see, completely arbitrary shapes can be laid out with a suitable set of linelength definitions. While this parshape may look a bit silly and useless,

one could conceive of situa-
tions such as finely tuned
dropped capitals, word
wrapping around non-
rectangular graphics,
etc. which will benefit
from such handcrafting.

The syntax is `\parshape numlines #1indent #1length #2indent #2length ... #nindent #nlength`, where `numlines` is the number of lines of text which define the paragraph. If there turn out to be fewer lines, the shape is truncated; if there are more, the excess lines have the same dimensions as the last line of the definition. The `#nindent` and `#nlength` entries specify the indentation of the line from the left margin, and the length of the line as measured from that point. The shape applies only to the current paragraph; everything is reset to normal for the next paragraph.

7.7 Summary

As you can see, the examples in this section range from the useful to the whimsical. While I don't expect that anyone will ever need the paragraph shape demonstrated in the last section, the important point is that you can do almost anything you want in LyX if you are willing to figure out how to do it in TeX and LaTeX. TeX is a fantastically powerful typesetting system and all that power is available to you since LyX uses it as its backend. Happy LyXing!