LyX-Anpassung: Möglichkeiten für fortgeschrittene Benutzer

vom LyX-Team*

Version 2.2.x

31. Juli 2016

*Übersetzung: Peter Sütterlin, Leif Albers und Hartmut Haase (HHa, bis März 2010).

Inhaltsverzeichnis

1. Einleitung

In diesem Teil der Dokumentation wird beschrieben, welche Möglichkeiten LyX bietet, um es den eigenen Wünschen anzupassen. Es werden Dinge wie Tastaturkürzel, Vorschau am Bildschirm, Optionen zum Drucken, das Senden von Befehlen an LyX durch den LyX-Server, Internationalisierung, Installation neuer LaTeX-Klassen und LyX-Layouts usw. behandelt. Es kann hier nicht alles beschrieben werden, das an LyX individuell eingestellt und verändert werden kann — die Entwickler fügen Neuerungen schneller ein, als wir sie dokumentieren können — doch werden die grundlegenden Fähigkeiten von LyX dokumentiert sowie für einige der etwas obskuren Hinweise gegeben.

Mit der Version 1.1.6 von LyX wurde eine neue Schnittstelle zu den konfigurierbaren Eigenschaften eingeführt, die Sie über den Menüpunkt Werkzeuge ▷ Einstellungen... aufrufen können. Diese macht die weiter unten aufgeführten Erläuterungen zu den Konfigurationsdateien von LyX nicht überflüssig, aber es vereinfacht doch den Prozess, LyX an *Ihre* Bedürfnisse anzupassen.

2. Die Konfigurationsdateien von LyX

Dieses Kapitel soll Ihnen dabei helfen, sich mit den Konfigurationsdateien von LyX vertraut zu machen. Bevor Sie jedoch weiterlesen, sollten Sie herausfinden, wo sich das Systemverzeichnis von LyX auf Ihrem Rechner befindet. Sie erfahren dies über den Menüpunkt Hilfe ▷ Über LyX. In diesem Verzeichnis speichert LyX alle systemweiten Konfigurationsdateien, wir werden es im weiteren LyXDir nennen.

2.1. Was befindet sich in LyXDir?

Das Verzeichnis LyXDir sowie seine Unterverzeichnisse enthalten eine Anzahl Dateien, mit denen das Verhalten von LyX beeinflusst werden kann. Diese Dateien können direkt von LyX aus über den Dialog Werkzeuge ▷ Einstellungen... gelesen und geändert werden. Fast alles, was Sie möglicherweise an LyX ändern wollen, können Sie hier einstellen. Jedoch können auch viele interne Dinge in LyX angepasst werden, indem man diverse andere Dateien in LyXDir verändert. Sie sind in verschiedene Kategorien unterteilt, die in den folgenden Unterabschnitten behandelt werden.

2.1.1. Automatisch erzeugte Dateien

Diese Dateien werden automatisch bei der Konfiguration von LyX erzeugt. Sie enthalten verschiedene Standardwerte, die durch Untersuchung des Systems ermittelt werden. Normalerweise sollte man sie nicht verändern, da sie jederzeit von LyX überschrieben werden können.

`lyxrc.defaults` enthält Standardwerte für diverse Befehle. Einstellungen, die Ihnen nicht zusagen, können einfach über Werkzeuge ▷ Einstellungen... verändert werden.

`packages.lst` enthält eine Auflistung aller LaTeX-Pakete, die von LyX erkannt wurden. Derzeit wird diese Liste von LyX selber nicht benutzt, jedoch ist die Information, zusammen mit einigem anderen, über den Menüpunkt Hilfe ▷ LaTeX Konfiguration zugänglich.

`textclass.lst` ist eine Liste mit den im Verzeichnis `layout` gefundenen Textklassen, zusammen mit den entsprechenden LaTeX-Dokumentenklassen und einer kurzen Beschreibung.

`doc/LaTeXConfig.lyx` wird bei der Konfiguration aus der Datei `LaTeXConfig.lyx.in` erzeugt. Jeder Eintrag der Form `@chk_bla@` wird dabei durch *yes* oder *no* ersetzt, je nachdem, ob das Paket `bla` gefunden wurde.

2.1.2. Verzeichnisse

`bind/` Dieses Verzeichnis enthält Dateien mit der Endung `.bind`. In ihnen werden die Tastenkombinationen festgelegt, mit denen Sie Menüs öffnen und Bearbeitungsoperationen direkt starten können. Falls eine Datei `$LANG_xxx.bind` mit einer an die internationalisierte Version von LyX angepassten Tastenbelegung existiert, wird diese bevorzugt geladen. Näheres dazu finden Sie in Kapitel 4.

`clipart/` Hier sind einige PostScript-Bilder gespeichert, die Sie in Ihre Dokumente einbinden können.

`doc/` Enthält die Dateien der LyX-Dokumentation (einschließlich der, die Sie gerade lesen). Eine kleine Ausnahmestellung hat die bereits beschriebene Datei `LaTeXConfig.lyx`. Auch hier wird eine übersetzte Version mit dem Vorsatz `$LANG_` vor dem Dateinamen zuerst gesucht und, falls vorhanden, geladen. Siehe dazu Kapitel 4.

`examples/` Enthält Beispieldateien, die erläutern, wie Sie die unterschiedlichen Möglichkeiten von LyX nutzen können. Verwenden Sie die Schaltfläche Beispiele im Dateiauswahlmenü, um in dieses Verzeichnis zu gelangen.

`images/[math/]` Enthält Bilddateien, die von der Dialogbox Aufzählungszeichen im Menü Dokument ▷ Einstellungen benötigt werden. Außerdem finden Sie hier die unterschiedlichen Icons für die Werkzeugleiste und das Bild für den Startbildschirm.

`kbd/` Hier sind die Definitionsdateien für die Tastaturbelegung gespeichert. Näheres dazu finden Sie im Kap. 4.3.

`layouts/` Hier werden die in Kapitel 5 beschriebenen Layoutdateien für die unterschiedlichen Dokumentenklassen gespeichert.

`lyx2lyx/` Enthält Dateien, die für die Konvertierung zwischen verschiedenen LyX-Versionen benötigt werden.

`scripts/` Hier sind einige Python-Skripte abgelegt, die LyX für bestimmte interne Operationen benötigt.

`templates/` Enthält die Vorlagendateien, die Ihnen bei Datei ▷ Neu von Vorlage präsentiert werden, siehe Kap. 5.2.4.

`tex/` Einige LyX-spezifische LaTeX-Textklassen- (`.cls`) und -Stildateien (`.sty`).

`ui/` Hier finden Sie Dateien mit der Endung `.ui`, die die Benutzerschnittstelle von LyX festlegen, also welche Einträge in welchen Menüs sind, und wie die Werkzeugleiste zusammengesetzt ist.

2.1.3. Dateien, die Sie nicht verändern sollten

Die folgenden Dateien werden intern von LyX verwendet. Sie sollten im Normalfall nur von den Entwicklern editiert werden.

`CREDITS` Diese Datei enthält eine Liste der Entwickler. Ihr Inhalt wird über die Menüauswahl Hilfe ▷ Über LyX angezeigt.

`chkconfig.ltx` ist ein LaTeX-Skript, das bei der Konfiguration verwendet wird. Starten Sie es nie direkt.

`configure` ist das eigentliche Skript, das zur Neukonfiguration von LyX verwendet wird. Es erzeugt die Konfigurationsdateien in dem Verzeichnis, von dem aus es aufgerufen wurde.

2.1.4. Andere Dateien

`encodings` Die hier enthaltene Tabelle beschreibt, wie die unterschiedlichen Zeichenkodierungen in Unicode dargestellt werden.

`external_templates` Diese Datei enthält die Vorlagen für das Feature Externes Material-Modul. Siehe dazu Kapitel 6

`languages` Eine Liste mit allen derzeit von LyX unterstützten Sprachen.

`latexfonts` Diese Datei enthält Informationen über die unterstützen LaTeX-Schriften.

`layouttranslations` Diese Datei enthält Übersetzungen für lokalisierbare Absatzstile (siehe Kap. 5.3.7).

`unicodesymbols` Diese Datei enthält Informationen über Unicode-kodierte Glyphen (Zeichen) und die Art und Weise, wie diese in LyX mit Hilfe von LaTeX unterstützt werden.

2.2. Das lokale Konfigurationsverzeichnis

Eventuell benutzen Sie LyX als normaler Benutzer und wollen dennoch einige Einstellungen der Konfiguration ändern. Zu diesem Zweck gibt es ein benutzereigenes Verzeichnis `UserDir`, in dem Ihre gesamte persönliche Konfiguration gespeichert wird.

Der Name dieses Verzeichnisses wird als *Benutzerverzeichnis* in Hilfe ▷ Über LyX angezeigt. Dieses Verzeichnis wird als Spiegelung des systemweiten Verzeichnisses verwendet. Das bedeutet, dass jede Datei, die Sie dort speichern, die entsprechende Datei im Systemverzeichnis ersetzt. Jede der im vorigen Abschnitt beschriebenen Konfigurationsdateien kann sich entweder im Systemverzeichnis `LyXDir` oder aber in Ihrem privaten Verzeichnis befinden. Im ersten Fall gelten die Einstellungen für alle Benutzer, im zweiten Fall nur für Sie.

Dies lässt sich an einigen Beispielen leichter erklären:

- Um eine LyX Version älter als 1.1.6 umzukonfigurieren, musste der Benutzer zunächst die Datei `LyXDir/lyxrc.example` nach `UserDir/lyxrc` kopieren und diese dann manuell editieren. Neuere Versionen von LyX lesen diese Datei zwar noch, wenn sie in `UserDir` gefunden wird, aber alle Änderungen, die über den neuen Dialog Werkzeuge ▷ Einstellungen gemacht werden, werden in der Datei `preferences` gespeichert. Danach (das heißt wenn `preferences` gefunden wird) wird die alte `lyxrc` nicht mehr eingelesen und kann gelöscht werden.

- Wenn Sie mit dem Menüpunkt Werkzeuge ▷ Neu konfigurieren eine Neukonfiguration von LyX durchführen, werden die dabei erzeugten Dateien in Ihrem privaten Konfigurationsverzeichnis `UserDir` gespeichert. Das bedeutet, dass ab sofort etwaige neue Dokumentenklassen, die Sie in Ihrem Verzeichnis `UserDir/layouts` gespeichert haben, im Feld Dokumentklasse des Dialoges Dokument ▷ Einstellungen... erscheinen.

- Falls Sie irgendwelche Dateien für neue LaTeX-Dokumentenklassen in einem Verzeichnis installiert haben, das LaTeX über die Umgebungsvariable `TEXINPUTS` findet, können auch diese in LyX verwendet werden.[1]

- Wenn Sie sich von einem LyX-FTP-Server eine aktuellere Version (oder zum Beispiel diese deutsche Version) der Dokumentation besorgt haben, sie aber nicht *offiziell* installieren können, da Sie keine Systemadministratorrechte haben, können Sie diese Dateien einfach nach `UserDir/doc` kopieren, und sie werden automatisch über das Hilfe-Menü geöffnet.

2.3. LyX mit mehreren Konfigurationen

Die hochgradige Konfigurierbarkeit von LyX durch das lokale Verzeichnis wird für diejenigen nicht ausreichend sein, die parallel mehrere unterschiedliche Konfigurationen verwenden wollen, zum Beispiel unterschiedliche Tastaturkürzel und/oder Druckerkonfigurationen. Sie können dies durch das Anlegen von mehreren Konfigurationsverzeichnissen erreichen und LyX jeweils beim Start mitteilen, welches davon verwendet werden soll.

[1] vorausgesetzt, es gibt auch eine `.layout`-Datei dafür.

Indem Sie LyX mit der Option `-userdir <verzeichnis>` starten, erreichen Sie, dass die Konfiguration aus diesem Verzeichnis anstelle des Standardverzeichnisses gelesen wird (das Standardverzeichnis ermitteln Sie, indem Sie LyX ohne diese Option starten). Falls das so angegebene Verzeichnis noch nicht existiert, fragt LyX genau wie beim ersten Start nach, ob es angelegt werden soll. Die Konfiguration in diesem Verzeichnis können Sie dann wie im Normalfall in LyX verändern, die Einstellungen im Standardverzeichnis werden aber nicht verändert – beide Verzeichnisse sind völlig unabhängig. Anstelle der Kommandozeilenoption können Sie übrigens auch die Umgebungsvariable `LYX_USERDIR_20x` auf das zu verwendende Verzeichnis setzen.

Unterschiedliche Konfigurationsverzeichnisse bedeuten aber auch zusätzlichen Aufwand: Wenn Sie etwa eine neue Layoutdatei in `UserDir/layouts` hinzufügen und diese für alle Konfigurationen sichtbar sein soll, müssen Sie sie in *allen* Verzeichnissen separat hinzufügen. Sie können das jedoch mit einem Trick umgehen: Nachdem LyX das neue UserDir angelegt hat, sind praktisch alle Unterverzeichnisse (siehe oben) leer. Sie können also all diese Verzeichnis durch einen symbolischen Link auf das entsprechende Verzeichnis im originalen `UserDir` ersetzen. Lediglich mit dem Verzeichnis `doc` müssen Sie vorsichtig sein, denn dort wird eine Datei durch das Konfigurationsskript (Werkzeuge ▷ Neu konfigurieren) abgelegt, die konfigurationsabhängig ist.

3. Der Dialog Werkzeuge ▷ Einstellungen

3.1. Formate

Als ersten Schritt müssen Sie Ihre Dateiformate definieren, wenn das nicht bereits passiert ist. Dazu öffnen Sie Werkzeuge ▷ Einstellungen ▷ Datei-Handhabung ▷ Dateiformate und klicken auf Neu. Das Format-Feld enthält den Namen, unter dem das Format im GUI identifiziert wird. Im Feld Einsortieren als steht der Name, mit dem das Format intern identifiziert wird. Außerdem muss eine Dateiendung festgelegt werden. Diese drei Felder sind erforderlich. Zusätzlich kann ein Tastenkürzel definiert werden. Zum Beispiel ruft Strg+D das Menü Dokument ▷ Ansicht ▷ DVI auf.

Ein Format kann ein Bearbeitungsprogramm und ein Anzeigeprogramm haben. Am Beispiel von JPEG-Dateien steht bei mir in beiden Feldern gimp. Was bei Ihnen steht, hängt von der Software-Ausstattung Ihres PCs ab. Zum Definieren des Befehls können auch die vier Variablen aus dem nächsten Abschnitt benutzt werden. Das Bearbeitungsprogramm wird aufgerufen, wenn Sie nach einem Rechtsklick auf ein Bild Datei extern bearbeiten auswählen.

Der MIME-Typ[1] eines Formats muss nicht zwingend angegeben werden, wenn er aber angegeben wird, dann sollte dies einheitlich über alle Formatvarianten hinweg geschehen. Der MIME-Typ wird verwendet, um ein Dateiformat über den Dateiinhalt zu erkennen. Für einige wichtige Dateiformate wurde von der zuständigen Organisation (IANA) noch kein offizielles MIME-Typ festgelegt. LyX verwendet daher die erweiterte inoffizielle Liste, die von freedesktop.org festgelegt wurde.

Wenn Dokumentformat angekreuzt ist, weiß LyX, dass das Format für den Dokumentexport geeignet ist. Wenn dann auch noch ein geeigneter Konverter existiert (siehe Kap. 3.3), wird das Format unter Datei ▷ Exportieren erscheinen. Außerdem wird es im Menü Dokument ▷ Ansicht erscheinen, wenn ein Anzeigeprogramm angegeben wurde. Reine Grafikformate wie `png` sollten diese Option nicht benutzen, dagegen aber Formate, die sowohl Vektorgrafiken als auch Dokumente repräsentieren wie `pdf`.

Die Option Vektorgrafik-Format sagt LyX, dass ein Format Vektorgrafiken enthalten kann. Diese Information wird dazu benutzt, um ein Zielformat für eingefügte Grafiken für den `pdflatex`-Export zu bestimmen. Eingefügte Grafiken müssen nach `pdf`, `png` oder `jpg` konvertiert werden, weil `pdflatex` keine anderen Grafikformate hand-

[1] MIME (*Multipurpose Internet Mail Extensions*) ist ein Kodierstandard, der ursprünglich entwickelt wurde, um die Struktur und den Aufbau von E-Mails festzulegen. Er wird mittlerweile aber auch zur generellen Bestimmung von Dateiformaten eingesetzt.

haben kann. Hat eine eingefügte Grafik bereits eines der Formate, wird sie nach pdf konvertiert, wenn Vektorgrafik-Format angekreuzt ist, sonst nach `png`.

3.2. Kopierer

Weil alle Konvertierungen im temporären Verzeichnis von LyX stattfinden, muss eine Datei manchmal geändert werden bevor sie ins temporäre Verzeichnis kopiert wird, damit die Konvertierung durchgeführt werden kann.[2] Das macht ein Kopierer: er kopiert eine Datei ins (oder vom) temporären Verzeichnis und ändert sie dabei.

Die Definitionen der Kopierer können acht Variablen benutzen:

`$$s` ist das Systemverzeichnis von LyX (zum Beispiel `/usr/local/bin/lyx`).

`$$i` ist die Eingabedatei.

`$$o` ist die Ausgabedatei.

`$$b` Der Basisname (ohne Dateinamenerweiterung), wie er im temporären LyX-Verzeichnis verwendet wird.

`$$p` ist der vollständige Dateipfad des temporären LyX-Verzeichnisses.

`$$r` ist der vollständige Dateipfad der LyX-Datei.

`$$f` ist der Dateiname der LyX-Datei (ohne Verzeichnispfad).

`$$l` ist der *LaTeX-Name*. Dies sollte der Dateiname sein, den LaTeX im `\include`-Befehl benutzt. Er ist nur dann relevant, wenn die exportierten Dateien für den Befehl geeignet sind.

Kopierer können benutzt werden, um *fast* alles mit Ausgabedateien zu machen. Wenn Sie zum Beispiel `pdf`-Dateien in ein spezielles Verzeichnis kopieren wollen, können Sie ein Shell-Skript wie folgt schreiben:

```
#!/bin/bash
FROMFILE=$1
TOFILE=‘basename $2‘
cp $FROMFILE /home/you/pdf/$TOFILE
```

Speichern Sie das Skript ausführbar in Ihrem lokalen LyX-Verzeichnis – etwa `/home/you/lyx/scrip`
Dann wählen Sie in Werkzeuge ▷ Einstellungen ▷ Datei-Handhabung ▷ Dateiformate das Format PDF (pdflatex) und tragen im Kopierer-Feld `pdfkopierer.sh $$i $$o` ein.

[2] Wenn die Datei beispielsweise auf andere Dateien mit relativen Pfaden verweist – vielleicht Bilder – und diese Pfade beim Kopieren ungültig werden.

Kopierer werden von LyX in vielen eigenen Konvertierungen benutzt. Wenn auf dem PC geeignete Programme installiert sind, wird LyX automatisch Kopierer für HTML und HTML (MS Word) installieren. Wenn diese Formate exportiert werden, *sieht* der Kopierer, dass nicht nur die Haupt-HTML-Datei, sondern auch verschiedene zugehörige Dateien (Stildateien, Bilder usw.) kopiert werden müssen. All diese Dateien werden in ein Unterverzeichnis des Verzeichnisses geschrieben, in dem die LyX-Datei steht.[3]

3.3. Konverter

Sie können eigene Konverter in Werkzeuge ▷ Einstellungen ▷ Datei-Handhabung ▷ Konverter definieren. Dazu wählen aus Von Format und In Format jeweils eins aus, schreiben den benötigten Befehl ins Feld Konverter und klicken auf Hinzufügen rechts oben. Sie können im Befehl mehrere Variablen benutzen:

`$$s` ist das Systemverzeichnis von LyX (zum Beispiel `/usr/local/bin/lyx`).

`$$i` ist die Eingabedatei.

`$$o` ist die Ausgabedatei.

`$$b` ist der Dateiname ohne Erweiterung (siehe Linux-Befehl `basename`).

`$$p` ist der Pfad zur Eingabedatei.

`$$r` ist der Pfad zur ursprünglichen Eingabedatei. Wenn eine Kette von Konvertern aufgerufen wird, weicht er von `$$p` ab.

Ins Feld Zusatz-Flag können Sie folgende, durch Kommata getrennte, Flags schreiben:

`latex` Damit ein besonderer LaTeX-Lauf gestartet, der die LaTeX-Fehlermeldungen von LyX verfügbar macht.

`needaux` benötigt die LaTeX-Datei `xyz.aux` zur Konvertierung.

`nice` benötigt eine „schöne“ Datei, also eine, die so aussieht, wie die, die man über das Menü exportiert (ohne interne Hilfsbefehle wie `input@path`).

`xml` Damit wird die Ausgabe im XML-Format gespeichert.

Die folgenden Flags sind keine richtigen, weil sie ein Argument der Form `key=value` benutzen:

[3] Kopierer können angepasst werden. Der optionale Parameter `-e` kann eine durch Kommata getrennte Liste von Erweiterungen enthalten, die mitkopiert werden sollen. Wenn es fehlt, werden alle Dateien kopiert. Der Parameter `-t` bestimmt die Namenserweiterung, die an den erzeugten Verzeichnisnamen angehängt werden soll. Standardmäßig ist es LyXconv, so dass die aus `Datei.lyx` erzeugte HTML-Datei im Unterverzeichnis `Datei.html.LyXconv` landet.

`parselog` Wenn das gesetzt ist, wird der Standardfehler des Konverters in die Datei `infile.out` umgeleitet, und das Skript wird so ausgeführt: `script < infile.out > infile.log`. Das Argument kann `$$s` enthalten.

`resultdir` ist der Name des Verzeichnisses, in dem der Konverter die erzeugten Dateien ablegen soll. LyX wird das Verzeichnis nicht anlegen und auch nichts hineinkopieren, aber dieses Verzeichnis an seinen Bestimmungsort kopieren. Das Argument darf `$$b` enthalten, was durch die Basisnamen von Ein- oder Ausgabedatei ersetzt wird, wenn das Verzeichnis kopiert wird.
Beachten Sie, das `resultdir` und `usetempdir` zusammen keinen Sinn machen. Wenn das erste definiert wurde, wird das zweite ignoriert.

`resultfile` ist der Name der Ausgabedatei und darf `$$b` enthalten. Er wird nur zusammen mit `resultdir` benutzt und ist auch da nur optional. Wenn er nicht angegeben wird, wird `index` benutzt.

Keines dieser Flags wird zur Zeit in einem Konverter benutzt, der zusammen mit LyX installiert wird.

Sie müssen nicht für alle Formate, zwischen denen Sie konvertieren wollen, Konverter definieren. Zum Beispiel gibt es keinen Konverter von LyX nach PostScript, aber LyX wird PostScript exportieren. Dies geschieht, indem zunächst eine LaTeX-Datei erzeugt wird – dafür wird auch kein Konverter benötigt –, die dann mit dem Konverter von LyX nach DVI in eine DVI-Datei konvertiert wird, die schließlich nach PostScript konvertiert wird. LyX findet solche Konverter-Ketten automatisch und wird immer die kürzeste finden.

Trotzdem können Sie Mehrfachkonversionen zwischen Dateiformaten definieren. Zum Beispiel liefert die Standardkonfiguration von LyX fünf Möglichkeiten, um von LaTeX nach PDF zu konvertieren:

1. direkt mit `pdflatex`
2. mit `ps2pdf` über DVI und PostScript
3. mit `dvipdfm` über DVI
4. direkt mit `XeTeX`
5. direkt mit `LuaTeX`

Um andere Ketten zu definieren, müssen Sie andere Ziel-*Dateiformate* definieren, wie in Kap. 3.1 beschrieben. Zum Beispiel enthält die Standardkonfiguration verschiedene Formate für pdf-Dateien, die `pdf` (für `ps2pdf`), `pdf2` (für `pdflatex`), `pdf3` (für `dvipdfm`), `pdf4` (für `XeTeX`) und `pdf5` (für `LuaTeX`) heißen.

4. Internationales LyX

Anmerkung des Übersetzers: Dieses Kapitel behandelt zwei Themenbereiche. Einmal wird in ?? *und* ?? *erklärt, wie man LyX mitteilt, dass man in einer fremdsprachlichen Umgebung arbeitet. Der überwiegende Rest des Kapitels erläutert, wie man LyX an eine neue Sprache anpasst, das heißt wie man es übersetzt. Da diese Arbeiten für eine deutsche Umgebung bereits erledigt wurden, dürfte dieser Teil des Kapitels für die Leser dieser Übersetzung weitgehend uninteressant sein. Die letzten beiden Abschnitte, Kap. 4.2 und Kap. 4.3, waren bei der Übersetzung noch nicht ganz up to date. (Leif Albers)*

LyX kann mit übersetzten Versionen seiner Benutzerschnittstelle arbeiten. Als dieser Text erstellt wurde, waren über den normalen englischen Text Anpassungen für 23 Sprachen Bestandteil der LyX-Distribution. Die von Ihnen benutze Sprache zeigt Ihnen der Befehl `locale`. (Für weitere Informationen über *locale*-Definitionen ist die Manpage *locale(5)* ein guter Startpunkt.)

Bitte beachten Sie, dass diese Übersetzungen zwar funktionieren, aber oft ein paar Einschränkungen unterliegen. Insbesondere wurde das Design der Popup-Menüs auf den englischen Text zugeschnitten. Das bedeutet, dass der übersetzte Text an einigen Stellen mehr Platz benötigt als dort zur Verfügung steht. Dies ist natürlich nur ein Darstellungsproblem und schränkt nicht die Funktionsweise von LyX ein. Sie werden auch feststellen, dass einige Übersetzungen nicht für alle Menüpunkte Tastenkürzel definieren. Manchmal stehen einfach nicht genügend freie Buchstaben zur Verfügung, manchmal hatte der Übersetzer einfach bisher keine Zeit, sich darum zu kümmern.

Wir werden versuchen, diese Dinge in einer späteren Version zu korrigieren.

4.1. LyX übersetzen

4.1.1. Die Benutzerschnittstelle übersetzen (Textmeldungen)

LyX verwendet die GNU-gettext-Bibliothek, um die Internationalisierung der Benutzerschnittstelle zu verwalten. Um LyX dazu zu bringen, in allen Menüs und Dialog-Boxen Ihre Lieblingssprache zu verwenden, müssen Sie eine `po`-Datei für diese Sprache erstellen. Anschließend müssen Sie daraus eine `mo`-Datei erzeugen und diese installieren. Eine umfassende Anleitung dazu finden Sie in der Dokumentation für GNU gettext.[1] Kurz gesagt müssen folgende Veränderungen durchgeführt werden (**`xx`** bezeichnet den Sprachencode der neuen Sprache):

[1] Natürlich nur auf englisch. Die Veränderungen, die an der `po`-Datei durchgeführt werden müssen, sind allerdings recht intuitiv.

- Laden Sie den LyX-Quellcode herunter. (Siehe die Informationen im Netz.)
- Kopieren Sie die Datei `lyx.pot` in das Verzeichnis der .po-Dateien. Benennen Sie anschließend die Datei in `xx.po` um. (Falls `lyx.pot` nicht existiert, kann sie mit dem Befehl `make lyx.pot` neu erzeugt werden.)
- Editieren Sie `xx.po`.[2] Für einige Menü- und Widgetfunktionen gibt es Tastenkürzel, die ebenfalls übersetzt werden sollten. Diese Tasten werden mit '|' markiert und sollten passend mitübersetzt werden. Sie sollten auch das Informationsfeld am Anfang der neuen po-Datei ausfüllen (mit Ihrer EMail-Adresse, usw.), damit Sie für andere Leute erreichbar sind, die Ihnen Vorschläge oder unterhaltsame Flames schicken möchten.
- Erzeugen Sie in `LYX-SOURCE-DIR/po/lyx.pot` mit dem Befehl `make update-gmo` die Datei `xx.gmo`.
- Installieren Sie die `gmo`-Datei mit dem Befehl `su -c 'make install'`.

Um eine neue `po`-Datei zu der LyX-*Distribution* hinzuzufügen, müssen eine Reihe von Dateien (Konfigurations-Skripts und mehr) verändert werden. Aber dank der gettext-Bibliothek ist ein Einbinden in den LyX-Quellcode für den Anwender unnötig.

Wenn Sie eine Übersetzung für eine Sprache erstellt haben, die LyX zur Zeit noch nicht unterstützt, sollten Sie uns ruhig ein Patch mit Ihrer Datei zusenden. Wie man ein Patch erstellt, erfahren Sie in der README-Datei im Verzeichnis `LYX-SOURCE-DIR/po/`.

4.1.2. Die Dokumentation übersetzen

[Anmerkung des Übersetzers: wenn *Sie die Dokumentation übersetzen wollen, benutzen Sie als Vorlage auf jeden Fall das englische Original. Übersetzungen – diese eingeschlossen – sind oft nicht ganz auf dem neuesten Stand. -LA]*

Auch die Online-Dokumentation (im Hilfe-Menü) kann (und sollte!) übersetzt werden. Wenn übersetzte Versionen verfügbar sind und die *locale* entsprechend gesetzt wurde, werden diese automatisch von LyX benutzt. Zur Zeit sind Übersetzungen in etwa 20 Sprachen vorhanden. LyX sucht nach übersetzten Versionen in `LyXDir/doc/xx_DocName.lyx` wobei wie immer `xx` für das entsprechende Sprachkürzel steht, das in der Umgebungsvariablen `LANG` gesetzt wird.

Falls solche Dateien nicht existieren, wird die englische Version verwendet. Auch die übersetzten Versionen müssen (bis auf das `xx_`) den gleichen (englischen) Dateinamen (im Beispiel oben `DocName`) tragen wie die englischen Originale. Wenn Sie gerne die Dokumentation übersetzen möchten (übrigens ein guter Weg, um die Originale Korrektur zu lesen!), hier ein paar Tipps, die Ihnen möglicherweise etwas Arbeit ersparen:

[2] Für diese Aufgabe gibt es spezielle Programme, wie `Poedit` (für alle Plattformen) oder `KBabel` (für KDE). `Emacs` hat ebenfalls einen Modus, der Sie bei dieser Arbeit unterstützt, siehe https://www.gnu.org/software/gettext/manual/html_node/PO-Mode.html#PO-Mode.

- Werfen Sie einen Blick auf die Seiten des Übersetzungsteams auf der Homepage des LyX-Entwickler-Teams: `http://www.lyx.org/Translation`. Dort erfahren Sie, welche Texte bereits in Ihre Sprache übersetzt sind, auch sehen Sie, ob jemand (und wenn ja, wer) die Übersetzungsaktivitäten koordiniert.

Wenn Sie dann mit der eigentlichen Übersetzungsarbeit beginnen, sind hier einige Tipps, die Ihnen vielleicht helfen, einige Schwierigkeiten zu überwinden:

- Machen Sie im Dokumentationsteam mit! Informationen dazu gibt es in Hilfe ▷ Einführung. Dies ist übrigens das erste Dokument, das Sie übersetzen sollten.
- Machen Sie sich mit den typographischen Konventionen der Sprache vertraut, in die Sie übersetzen möchten. Typographie ist eine alte Kunst, und in vielen Teilen der Welt wurden verschiedene Konventionen eingeführt. Auch sollten Sie die typographische Terminologie in Ihrem Land lernen. Eine eigene Terminologie würde nur die Leser verwirren. *(Warnung: Typographie macht süchtig!)*
- Legen Sie eine Kopie des Originaldokumentes an. Dies wird Ihre Arbeitskopie. Sie können diese als selbst-übersetzten Hilfe-Datei in LyX verwenden, indem Sie sie in den Ordner `UserDir/doc/xx/` kopieren.
  Achtung: Für komplexe Dokumente mit externem Material (Bilder usw.) werden die Dateipfade von relativ auf absolut geändert, wenn man das Dokument verschiebt. Daher ist es das Beste, LyX mittels Git zu beziehen (siehe `http://www.lyx.org/HowToUseGIT`) und das Dokument im Verzeichnis zu belassen.
- Wann immer Sie einen Fehler im Originaltext entdecken, korrigieren Sie ihn und teilen dem Rest des Dokumentationsteams Ihre Veränderungen mit. (Sie haben nicht vergessen, dem Dokumentationsteam beizutreten, oder?) *Auch die Originaldokumentation ist nicht komplett.*

4.2. Internationale Tastaturbelegung

4.2.1. Eigene Tastaturtabellen definieren: das *Keymap*-Dateiformat

Sehen wir uns einmal die *Keymap*-Datei ein wenig näher an. Es handelt sich um eine ASCII-Datei, in der folgendes definiert wird:

- Taste-Taste- oder Taste-String-Transformationen
- *Tote* Tasten – sogenannte *dead keys*
- Ausnahmen für tote Tasten.

Zur Definition einer Taste-Taste- oder Taste-String-Transformation dient folgender Befehl:

```
\kmap Taste Ausgabe
```

wobei `Taste` die zu übersetzende Taste bezeichnet und `Ausgabe` die Taste oder den String, der dafür in das Dokument eingefügt werden soll. Eine tote Taste definiert man mit:

```
\kmod Taste Tote-Taste
```

wobei `Taste` wieder eine Taste auf der Tastatur bezeichnet und `Tote-Taste` der Name einer toten Taste ist. LyX unterstützt folgende toten Tasten (Abkürzungen in Klammern):

Name	*Beispiel*
acute (acu)	áéíóú
grave (gra)	àèìòù
macron (mac)	ō
tilde (til)	ñÑ
underbar (underb)	o̲
cedilla (ced)	çÇ
underdot (underd)	ọ
circumflex (circu)	âêîôû
circle (circl)	ÅůŮ
tie (tie)	o͡o
breve (bre)	ăŏ
caron (car)	čšž
hungarian umlaut (hug)	őű
umlaut (uml)	äöü
dot (dot)	żṡ

Da es auf vielen internationalen Tastaturen Ausnahmen dafür gibt, wie eine bestimmte tote Taste das folgende Zeichen verändern soll, können diese definiert werden, und zwar mit:

```
\kxmod Tote-Taste Taste Ausgabe
```

Zum Beispiel soll caron-o auf einer slowakischen Tastatur ein circumflex-o erzeugen. Dies erreicht man mit:

```
\kxmod caron o "\^o"
```

Auch müssen für die Buchstaben i und j Ausnahmen definiert werden, um den Punkt zu löschen, bevor ein Akzent eingefügt wird. Ich werde dies beizeiten ändern, hatte aber bisher noch keine Zeit dazu.

Ach so, zur Definition der Ausgabe: Der Backslash „\“ ist ein Sonderzeichen. Um ihn einzugeben, muss man einen doppelten Backslash „\\“ eingeben. Auch das Anführungszeichen „"“ und das Doppelkreuz „#“ haben eine andere Bedeutung. # bezeichnet einen Kommentar, Anführungszeichen markieren den Anfang und das Ende

eines Strings (das heißt einer LaTeX-Befehlssequenz). Um diese Zeichen einzugeben, muss ein Backslash vorangestellt werden (also: `\"` und `\#`). Wenn Sie eine funktionierende Keymap-Datei für eine neue Sprache angefertigt haben, mailen Sie diese bitte an das Entwickler-Team, damit sie in die nächste Distribution integriert werden kann.

In Zukunft werden auch noch folgende Befehle unterstützt:

- `\kinclude Dateiname` bindet eine andere Datei ein
- `\kprog Programm` definiert ein externes Keymap-Programm

Auch sollte es die `lyxrc`-Datei nach Voreinstellungen durchsuchen (zum Beispiel nach einer Option `\kinclude`, um eine Standardtastatur zu verwenden).

4.3. Internationale Tastaturtabellen: *Keymaps*

Die nächsten beiden Abschnitte beschreiben detailliert die Syntax der `.kmap`- und `.cdef`-Dateien. Diese Abschnitte sollten Ihnen dabei helfen, Ihre eigene Tastaturtabelle zu entwerfen, wenn die vorhandenen nicht ganz Ihren Bedürfnissen entsprechen.

4.3.1. Die `.kmap`-Datei

Eine .`kmap`-Datei transformiert gedrückte Tasten zu Buchstaben oder Strings (Zeichenketten) – es definiert ein *keyboard mapping*. Im Folgenden werden die Schlüsselwörter `kmap`, `kmod`, `kxmod` und `kcomb` beschrieben.

`kmap` Transformiere einen Buchstaben zu einem String

`\kmap` *Zeichen Ausgabe*

Dieser Ausdruck definiert, dass *Zeichen* zu *Ausgabe* transformiert werden soll. Dabei müssen in *Ausgabe* die Zeichen Backslash „\“ und Anführungszeichen „"“ mit einem vorangehenden Backslash versehen werden.

Als Beispiel ein Ausdruck, der das Zeichen „`/`“ ausgibt, wenn die Taste „`&`“ gedrückt wurde:

```
\kmap & /
```

`kmod` Spezifiziere ein Akzentzeichen

`\kmod` *Zeichen Akzent erlaubt*

Dieser Ausdruck wird dafür sorgen, dass *Zeichen* als ein bestimmter *Akzent* interpretiert wird, und zwar bei allen Zeichen, die in *erlaubt* aufgeführt sind. Dies ist der

Mechanismus toter Tasten *(dead keys)*.[3] Wenn Sie die Taste *Zeichen* drücken, gefolgt von einem Zeichen, das *nicht* in *erlaubt* aufgeführt wurde, werden einfach beide Zeichen einzeln ausgegeben.

Der folgende Ausdruck definiert, dass die Taste „^“ der circumflex-Akzent wird, wenn er von einem der Buchstaben a, e, i, o, u, A, E, I ,O oder U gefolgt wird:

```
\kmod ^ circumflex aeiouAEIOU
```

`kxmod` Definiere eine Ausnahme zu einem Akzentzeichen

`\kxmod` *Akzent Zeichen Ausgabe*

Dieser Ausdruck definiert eine Ausnahme für die Wirkung, die *Akzent* in Verbindung mit *Zeichen* haben soll. Dabei muss *Akzent* vorher mit Hilfe einer `\kmod`-Zeile einer Taste zugewiesen worden sein. Wenn Sie die Sequenz *Akzent, Zeichen* drücken, wird *Ausgabe* produziert. Falls solch eine Definition *nicht* existiert, und Sie *Akzent, Zeichen* eingeben, erhalten Sie das *Zeichen* – akzentuiert.

Der folgende Ausdruck sorgt dafür, dass LaTeX bei einem „i“ mit circumflex den I-Punkt entfernt, bevor das Akzentsymbol eingefügt wird:

```
\kxmod circumflex i "\\^{\\i}"
```

`kcomb` Kombiniere zwei Akzentsymbole

`\kcomb` *Akzent1 Akzent2 erlaubt*

Hier wird es ziemlich esoterisch. Dieser Ausdruck erlaubt die Kombination der Effekte von *Akzent1* und *Akzent2* (in dieser Reihenfolge!) bei allen *erlaubten* Zeichen. Die Bedeutungen von *Akzent1* und *Akzent2* müssen zuvor mit Hilfe von `\kmod` definiert worden sein.

Folgendes Beispiel aus der Datei `greek.kmap`:

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

Diese Zeilen erlauben es, „`;:i`“ einzugeben und auf diese Weise „`\'{\"{i}}`“ zu erzeugen (ΐ). In diesem Fall löscht die Backspace-Taste das letzte gedrückte Zeichen. Wenn Sie also `;:` Backspace `i` eingeben, erhalten Sie „`\'(i)`“ (ί).

[3] Der Ausdruck *tote Taste* kommt daher, dass diese Taste allein kein Zeichen erzeugt, aber, gefolgt von einer anderen Taste, akzentuierte Zeichen erzeugt. Zum Beispiel kann auf diese Weise é“ erzeugt werden.

4.3.2. Die `.cdef`-Datei

Nachdem LyX die `.kmap`-Datei verarbeitet hat, erklärt eine `.cdef`-Datei, wie die einzelnen Symbole im gegenwärtigen Zeichensatz dargestellt werden sollen. Die LyX-Distribution enthält wenigstens die Dateien `iso8859-1.cdef` und `iso8859-2.cdef`.

Generell besteht eine `.cdef`-Datei aus einer Reihe von Deklarationen der folgenden Form:

Position_im_Zeichensatz String

Um beispielsweise dem String (*Ausgabe* im vorigen Abschnitt) „`'{e}`“ das entsprechende Zeichen im ISO-8859-1 Zeichensatz (233) zuzuweisen, benutzt man folgenden Ausdruck:

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

Wieder müssen den Zeichen „\“ und „"“ ein Backslash vorangestellt werden. Beachten Sie, dass es durchaus möglich ist, dass dasselbe Zeichen (sinnvoll) zwei verschiedene Strings repräsentieren kann. Zum Beispiel in `iso-8859-7.cdef` gibt es die Zeilen:

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

Wenn LyX kein passendes Zeichen für einen String finden kann, der durch eine Tastensequenz erzeugt wurde, wird es versuchen, falls der String wie ein akzentuierter Buchstabe aussieht, auf dem Bildschirm den Buchstaben mit Akzent selbst zu zeichnen.

4.3.3. Tote Tasten definieren

Anmerkung des Übersetzers: An dieser Stelle weiche ich krass vom Original ab. Der englische Text ist hier viel zu weitschweifig und ein bisschen konfus. -LA

Es gibt noch eine zweite Möglichkeit, internationale Buchstaben mit Hilfe von *toten Tasten (dead keys)* zu erzeugen – nämlich direkt in der `.bind`-Datei. Dazu ein einfaches Beispiel:

Nehmen wir an, Sie benötigen eine Zirkumflex-Taste. Diese können Sie definieren, indem Sie in der `lyxrc`-Datei folgende Zeile einfügen:

```
\bind "asciicircum" "accent-circumflex"
```

Dabei ist `asciicircum` die Bezeichnung, die das X11-System für die „^“-Taste verwendet.[4] `accent-circumflex` ist ein LyX-Befehl, der den Zirkumflex-Akzent erzeugt.

Leider unterscheidet sich die Wirkungsweise der toten Tasten, die in `.lyxrc` definiert wurden, merklich von der, die in **??**subsec:Die-.kmap-Datei auf Seite 17 beschrieben wurde.

[4] Die Bezeichnungen anderer Symbole lassen sich ganz gut aus einer c-Include-Datei namens `keysymdef.h` ablesen. Meist findet man sie in `/usr/X11/include/X11/`.

4.3.4. Ihre Sprachkonfiguration einstellen

Sie können Ihre `lyxrc-`Datei so verändern, dass Ihre gewünschte Sprachumgebung automatisch geladen wird, wenn LyX gestartet wird. Dieser Abschnitt beschreibt Befehle, die folgendes spezifizieren:

- Standard-, erste und zweite Tastaturbelegung
- Zeichensatzkodierung

In Ihrer `lyxrc-`Datei finden Sie verschiedene Beispiele, wie man sie konfigurieren kann. Zum Beispiel können Sie einen Eintrag für eine Tastaturbelegung folgendermaßen vornehmen:

```
\bind "american" "keymap-primary"
```

eine tote Taste *(dead key)* definieren:

```
\bind "Alt+,"  "accent-cedilla"
```

oder eine Zeichensatzkodierung festlegen:

```
# Die Norm für die Bildschirmzeichensätze
# Voreinstellung ist iso8859-1.
\font_norm iso8859-2
```

5. Installieren neuer Textklassen, Layouts und Vorlagen

In diesem Abschnitt wird beschrieben, wie Sie beim Installieren neuer Layout- und Vorlagedateien vorgehen müssen, außerdem auch eine kleine Auffrischung, wie man neue Dokumentenklassen für LaTeX korrekt installiert. Zunächst ein paar Definitionen:

Eine *Dokumentenklasse* ist eine LaTeX-Datei (normalerweise mit der Endung `.cls` oder `.sty`), die das Format einer speziellen Art von Dokument beschreibt, etwa Artikel, Brief usw., und auch alle dazu notwendigen Befehle definiert.

Eine *Layout-Datei* ist eine LyX-Datei, die einer LaTeX-Dokumentenklasse entspricht und LyX mitteilt, wie die diversen Formatelemente am Bildschirm dargestellt werden sollen, damit der Eindruck möglichst gut dem späteren Druckbild entspricht. Genauer gesagt beschreibt eine Layoutdatei eine *Textklasse*, das interne Konstrukt, welches LyX verwendet, um den Text am Bildschirm darzustellen.

Layout und *Textklasse* sind somit in gewisser Weise äquivalent, aber es ist besser, die Datei als Layout zu bezeichnen und die interne Realisation im Speicher von LyX als Textklasse. Eine Vorlage ist einfach ein LyX-Dokument, welches bereits einige vordefinierte Einträge für eine bestimmte Textklasse enthält. Derartige Vorlagen sind beispielsweise für Briefe oder Artikel für Zeitschriften sehr hilfreich.

5.1. Installation eines neuen LaTeX-Paketes

Bei manchen TeX-Installationen fehlt möglicherweise das eine oder andere Paket, das Sie gerne mit LyX verwenden würden. Zum Beispiel wollen Sie FoilTeX verwenden, ein Paket zur Erstellung von Dias und Folien für Overheadprojektoren. Moderne LaTeX-Distributionen wie TeXLive (2008 oder neuer) oder MiKTeX besitzen ein grafisches Programm um solche Pakete zu installieren. Z. B. bei MiKTeX starten Sie das Programm „Package Manager“ um eine Liste mit den verfügbaren Paketen zu bekommen. Um eines davon zu installieren, rechts-klicken Sie oder benutzen den entsprechenden Werkzeugleistenknopf.

Falls Ihre LaTeX-Distribution keinen Paketmanager besitzt, oder falls das Paket nicht direkt über Ihre Distribution verfügbar ist, folgen Sie diesen Schritten um es manuell zu installieren:

1. Besorgen Sie sich das Paket von CTAN oder einer anderen Quelle.

2. Falls das Paket eine Datei mit der Endung „`.ins`“ enthält (was bei FoilTEX der Fall ist), dann öffnen sie eine Kommandozeile wechseln in das Verzeichnis der Datei und führen den Befehl
   latex foiltex.ins
   aus. Sie haben damit dass Paket entpackt und haben alle Dateien um es zu installieren. Die meisten LATEX-Pakete sind nicht gepackt und man kann direkt mit der Installation beginnen:

3. Nun müssen Sie entscheiden, ob das Paket für alle Nutzer oder nur für Sie verfügbar sein soll.

   a) Bei *nix Systemen (Linux, OSX, etc.), wenn Sie das Paket für alle Nutzer installieren möchten, installieren Sie es in den lokalen TEX Ordner; anderenfalls installieren Sie es in den eigenen „Benutzer“-TEX-Ordner. Wo man diese Ordner anlegt, sofern sie nicht schon existieren, hängt von Ihrem System ab. Dazu schauen Sie in die Datei `texmf.cnf`.[1] Der Ort des lokalen TEX-Ordners ist in der Variable `TEXMFLOCAL` definiert; es ist üblicherweise der Pfad /usr/local/share/texmf/ oder /usr/local/texlive/XXXX, wobei XXXX das Jahr der installierten TEXLive-Distribution ist. Der Ort des Benutzer-TEX-Ordners ist in der Variable `TEXMFHOME` definiert und ist üblicherweise der Pfad $HOME/texmf/ oder $HOME/.texliveXXXX. (Wenn diese Variablen nicht vordefiniert sind, müssen Sie diese selbst definieren.) Sie brauchen wahrscheinlich Root-Rechte um in den lokalen TEX-Ordner zu schreiben, beim Benutzer-TEX-Ordner ist die nicht nötig.
   Allgemein empfiehlt es sich, Pakete in den Benutzer-TEX-Ordner zu installieren, da dieser nicht verändert oder gar überschrieben wird, wenn Sie ihr System aktualisieren. Des Weiteren wird er zusammen mit Ihren Nutzerdaten gesichert, wenn Sie ein Backup machen (was Sie natürlich regelmäßig tun).

   b) Bei Windows, wenn Sie das Paket für alle Nutzer installieren möchten, gehen Sie in den Ordner, in dem LATEX installiert ist und wechseln dort in das Verzeichnis ~tex\latex. (Verwendet man MiKTEX, wäre es standardmäßig der Ordner ~:\Programme\MiKTeX\tex\latex .) Legen Sie dort einen neuen Ordner mit dem Namen „foiltex“ an und kopieren Sie alle Dateien das Pakets hinein. Wenn das Paket nur für den aktuellen Benutzer verfügbar sein soll bzw. Sie keine Administrator-Rechte haben, tun Sie dasselbe, aber im lokalen LATEX-Ordner. Z. B. bei MiKTEX 2.9 wäre das unter WinXP der Ordner
   ~:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten\
   MiKTeX\2.9\tex\latex
   , unter WinVista wäre es der Ordner
   ~:\Users\<Benutzername>\AppData\Roaming\2.9\MiKTeX\tex\latex .

[1] Diese befindet sich normalerweise im Ordner $TEXMF/web2c. Falls nicht, führen Sie den Befehl `kpsewhich texmf.cnf` aus, um sie zu lokalisieren.

4. Jetzt muss man LaTeX nur noch mitteilen, dass es neue Dateien gibt. Die ist je nach LaTeX-Distribution anders:

   a) Bei TeXLive führen Sie von einer Kommandozeile den Befehl `texhash` aus. Wenn Sie das Paket für alle Nutzer installiert haben, brauchen sie dazu wahrscheinlich Root-Rechte.

   b) Bei MiKTeX, wenn Sie das Paket für alle Nutzer installiert haben, starten Sie das Programm „Settings (Admin)“ und drücken dann auf den Kopf „Refresh FNDB“. Anderenfalls starten Sie das Programm „Settings“ und machen dasselbe.

5. Nun muss man LyX noch mitteilen, dass es neue Pakete gibt. Verwenden Sie dazu in LyX das Menü Werkzeuge ▷ Neu konfigurieren und starten LyX danach neu.

Nun ist das Paket installiert. In unserem Beispiel wird nun die Dokumentklasse `Slides (FoilTeX)` im Menü Dokument ▷ Einstellungen ▷ Dokumentklasse verfügbar sein.

Möchten sie eine LaTeX-Dokumentklasse verwenden, die generell nicht im Menü Dokument ▷ Einstellungen ▷ Dokumentklasse gelistet ist, müssen Sie dafür selbst ein „Layout“ erstellen. Dies ist das Thema des nächsten Abschnitts.

5.2. Layout-Dateitypen

This section describes the various sorts of LyX files that contain layout information. These files describe various paragraph and character styles, determining how LyX should display them and how they should be translated into LaTeX, DocBook, XHTML, or whatever output format is being used.

We shall try to provide a thorough description of the process of writing layout files here. However, there are so many different types of documents supported even by just LaTeX that we can't hope to cover every different possibility or problem you might encounter. The LyX users' list is frequented by people with lots of experience with layout design who are willing to share what they've learned, so please feel free to ask questions there.

As you prepare to write a new layout, it is extremely helpful to look at the layouts distributed with LyX. If you write a LyX layout for a LaTeX document class that might also be used by others, or write a module that might be useful to others, then you should consider posting your layout to the layout section on the LyX wiki or even to the LyX developers' list, so that it might be included in LyX itself.[2]

[2]Note that LyX is licensed under the General Public License, so any material that is contributed to LyX must be similarly licensed.

5.2.1. Layout Module

We have spoken to this point about 'layout files'. But there are different sorts of files that contain layout information. Layout files, strictly so called, have the `.layout` extension and provide LyX with information about document classes. As of LyX 1.6, however, layout information can also be contained in layout *modules*, which have the `.module` extension. Modules are to LaTeX packages much as layouts are to LaTeX classes, and some modules—such as the `endnotes` module—specifically provide support for one package. In a sense, layout modules are similar to included[3] files—files like `stdsections.inc`—in that modules are not specific to a given document class but may be used with many different classes. The difference is that using an included file with `article.cls` requires editing that file. Modules, by contrast, are selected in the Document ▷ Settings dialog.

Building modules is the easiest way to get started with layout editing, since it can be as simple as adding a single new paragraph style or flex inset. But modules may, in principle, contain anything a layout file can contain.

After creating a new module and copying it to the `layouts/` folder, you will need to reconfigure and then restart LyX for the module to appear in the menu. However, changes you make to the module will be seen immediately, if you open Document ▷ Settings, highlight something, and then hit "OK". *It is strongly recommended that you save your work before doing this.* In fact, *it is strongly recommended that you not attempt to edit modules while simultaneously working on actual documents.* Though of course the developers strive to keep LyX stable in such situations, syntax errors and the like in your module file could cause strange behavior.

5.2.1.1. Lokales Format

Modules are to LyX as packages are to LaTeX. Sometimes, however, you find yourself wanting a specific inset or character style just for one document and writing a module that will also be available to other documents makes little sense. What you need is LyX's "Local Layout".

You will find it under Document ▷ Settings ▷ Local Layout. The large text box allows you to enter anything that you might enter in a layout file or module. You can think of a document's local layout, in fact, as a module that belongs just to it. So, in particular, you must enter a `Format` tag. Any format is acceptable, but one would normally use the format current at the time. (In LyX 2.2, the current layout format is 60.)

When you have entered something in the `Local Layout` pane, LyX will enable the "Validate" button at the bottom. Clicking this button will cause LyX to determine whether what you have entered is valid layout information for the chosen format. LyX will report the result but, unfortunately, will not tell you what errors there might have been. These will be written to the terminal, however, if LyX is started from a

[3] These can have any extension, but by convention have the `.inc` extension.

terminal. You will not be permitted to save your local layout until you have entered something valid.

The warnings at the end of the previous section apply here, too. Do not play with local layout while you are actually working, especially if you have not saved your document. That said, using local layout with a test document can be a very convenient way to try out layout ideas, or even to start developing a module.

5.2.2. Layout für .sty-Dateien

There are two situations you are likely to encounter when wanting to support a new LaTeX document class, involving style (`.sty`) files and LaTeX 2_ε class (`.cls`) . Supporting a style file is usually fairly easy. Supporting a new class file is a bit harder. We'll discuss the former in this section and the latter in the next. Similar remarks apply, of course, if you want to support a new DocBook DTD.

The easier case is the one in which your new document class is provided as a style file that is to be used in conjunction with an already supported document class. For the sake of the example, we'll assume that the style file is called MYCLASS.STY and that it is meant to be used with REPORT.CLS, which is a standard class.

Start by copying the existing class's layout file into your local directory:[4]

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

Then edit `myclass.layout` and change the line:

```
\DeclareLaTeXClass{report}
```

to read

```
\DeclareLaTeXClass[report, myclass.sty]{report (myclass)}
```

Then add:

```
Preamble
    \usepackage{myclass}
EndPreamble
```

near the top of the file.

Start LyX and select TOOLS ▷ RECONFIGURE. Then restart LyX and try creating a new document. You should see "REPORT (MYCLASS)" as a document class option in the DOCUMENT ▷ SETTINGS dialog. It is likely that some of the sectioning commands and such in your new class will work differently from how they worked in the base class—`report` in this example—so you can fiddle around with the settings for the different sections if you wish. The layout information for sections is contained in `stdsections.inc`, but you do not need to copy and change this file. Instead, you can simply add your changes to your layout file, after the line `Input stdclass.inc`, which itself includes `stdsections.inc`. For example, you might add these lines:

[4] Of course, which directory is your local directory will vary by platform, and LyX allows you to specify your local directory on startup, too, using the `-userdir` option.

```
Style Chapter
    Font
        Family Sans
    EndFont
End
```

to change the font for chapter headings to sans-serif. This will override (or, in this case, add to) the existing declaration for the Chapter style.

Your new package may also provide commands or environments not present in the base class. In this case, you will want to add these to the layout file. See section 5.3 for information on how to do so.

If MYCLASS.STY can be used with several different document classes, and even if it cannot, you might find it easiest just to write a module that you can load with the base class. The simplest possible such module would be:

```
#\DeclareLyXModule{My Package}
#DescriptionBegin
#Support for mypkg.sty.
#DescriptionEnd
Format 60
Preamble
    \usepackage{mypkg}
EndPreamble
```

A more complex module might modify the behavior of some existing constructs or define some new ones. Again, see section 5.3 for discussion.

5.2.3. Layout für .cls-Dateien

There are two possibilities here. One is that the class file is itself based upon an existing document class. For example, many thesis classes are based upon BOOK.CLS. To see whether yours is, look for a line like

```
\LoadClass{book}
```

in the file. If so, then you may proceed largely as in the previous section, though the `DeclareLaTeXClass` line will be different. If your new class is `thesis` and it is based upon `book`, then the line should read:[5]

```
\DeclareLaTeXClass[thesis,book]{thesis}
```

If, on the other hand, the new class is not based upon an existing class, you will probably have to "roll your own" layout. We strongly suggest copying an existing layout file which uses a similar LaTeX class and then modifying it, if you can do so. At least use an existing file as a starting point so you can find out what items you need to worry about. Again, the specifics are covered below.

[5] And it will be easiest if you save the file to `thesis.layout`: LyX assumes that the document class has the same name as the layout file.

5.2.4. Vorlagen erstellen

Once you have written a layout file for a new document class, you might want to consider writing a *template* for it, too. A template acts as a kind of tutorial for your layout, showing how it might be used, though containing dummy content. You can of course look at the various templates included with LyX for ideas.

Templates are created just like usual documents: using LyX. The only difference is that usual documents contain all possible settings, including the font scheme and the paper size. Usually a user doesn't want a template to overwrite his preferred settings for such parameters. For that reason, the designer of a template should remove the corresponding commands like `\font_roman` or `\papersize` from the template LyX file. This can be done with any simple text-editor, for example `vi` or `notepad`.

Put the edited template files you create in `UserDir/templates/`, copy the ones you use from the global template directory in `LyXDir/templates/` to the same place, and redefine the template path in the Tools ▷ Preferences ▷ Paths dialog.

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

5.2.5. Alte Layout-Dateien auf den neuesten Stand bringen

Das Format der Layout-Dateien ändert sich mit jeder LyX-Version. Daher müssen die Layout-Dateien in das neue Format konvertiert werden. Wenn LyX eine Layout-Datei eines älteren Formats liest, ruft es automatisch das Skript `layout2layout.py` auf um es in eine temporäre Datei im aktuellen Format zu konvertieren. Die Originaldatei wird nicht verändert. Wenn Sie die Layout-Datei öfter verwenden, dann können Sie sie permanent in das neue Format konvertieren, so dass LyX dies nicht jedes Mal tun muss. Um das zu tun, machen sie Folgendes:

1. Benennen Sie `MeineKlasse.layout` in `MeineKlasse.alt` um.
2. Rufen Sie den Befehl
   `python LyXDir/scripts/layout2layout.py myclass.old myclass.layout`
   auf. Wobei `LyXDir` der Name Ihres LyX-Systemverzeichnisses ist.

Beachten Sie, das manuelle Konvertierungen keine eingefügten Dateien mit konvertieren. Diese müssen separat konvertiert werden.

5.3. Das Layout-Dateiformat

Die folgenden Abschnitte beschreiben wie Layout-Dateien aufgebaut sind und erstellt werden. Wir empfehlen bei der Erstellung von Layouts langsam zu beginnen und sich Stück für Stück vorzuarbeiten. Es ist nicht wirklich schwer, jedoch sind die

möglichen Optionen manchmal etwas erschlagend, besonders wenn man zu viele davon auf einmal ausprobiert. Am einfachsten ist es, wenn man bestehende Layout-Dateien von LyX als Beispiel nimmt oder diese umgestaltet.

Beachten Sie dass alle Tags in Layout-Dateien nicht durch Groß- und Kleinschreibung zu unterscheiden sind. Das bedeutet dass `Style`, `style` und `StYlE` derselbe Tag sind. Die möglichen Argumente für die Tags sind hinter ihnen in eckigen Klammern angegeben. Das voreingestellte Argument ist ***hervorgehoben***. Wenn das Argument einen Datentyp hat wie „string“ oder „float“, wird die Voreinstellung so angezeigt: `float=`***`default`***.

5.3.1. Deklaration einer neuen Textklasse und Klassifikation

Zeilen, die mit einem `#` beginnen, sind Kommentare. Mit einer Ausnahme: alle Textklassen sollten mit Zeilen ähnlich wie den folgenden beginnen:

```
#% Do not delete the line below; configure depends on this[6]
#  \DeclareLaTeXClass{Article (Standard Class)}
#  \DeclareCategory{Articles}
```

Die zweite und dritte Zeile wird benötigt, wenn Sie LyX konfigurieren. Die Textklassen-Datei wird von dem LaTeX-Skript chkconfig.ltx gelesen, und zwar in einem speziellen Modus, in dem `#`-Zeichen ignoriert werden. Die erste Zeile ist einfach ein LaTeX-Kommentar, in der zweiten muss die Textklasse deklariert werden und die dritte Zeile enthält die optionale Klassifikation der Klasse. Eine Datei namens `article.layout`, die mit diesen beiden Zeilen beginnt, definiert eine Textklasse mit dem Namen article (der Name der Layout-Datei) und benutzt die LaTeX-Dokumentenklasse `article.cls` (Standard ist denselben Namen wie das Layout zu verwenden). Die Zeichenkette „Article (Standard Class)“, die oben erscheint, ist auch die Beschreibung, die später im Popup-Menü Dokument ▷ Einstellungen auftaucht. Die Kategorie („Articles“ im Beispiel) wird auch im Dialog Dokument ▷ Einstellungen verwendet: Die Textklassen weden nach diesen Kategorien gruppiert (was üblicherweise Genres sind, typische Kategorien sind also „Artikel“, „Bücher“, „Berichte“, „Briefe“, „Präsentationen“, „Lebensläufe“ usw.). Wenn keine Kategorie deklariert wurde, wird die Klasse in die Gruppe „Nicht kategorisiert“ getan.

Angenommen, Sie möchten Ihre eigene Textklasse schreiben, welche die LaTeX-Dokumentenklasse `article` benutzt, in der Sie aber das Aussehen der Kopfzeile verändert haben. Wenn Sie dann Ihre Textklasse in eine Datei namens `myarticle.layout` schreiben, sollten die ersten beiden Zeilen der Datei etwa so aussehen:

```
#% Do not delete the line below; configure depends on this
#  \DeclareLaTeXClass[article]{Article (with My Own Headings)}
#  \DeclareCategory{Articles}
```

[6] zu Deutsch: Löschen Sie die folgenden Zeilen nicht, da die Konfiguration davon abhängt

Auf diese Weise deklarieren Sie eine Textklasse `myarticle`, die die LaTeX-Dokumentklasse `article.cls` verwendet und (im Popup-Menü) beschrieben wird mit: Article (with My Own Headings). Falls Ihre Textklasse auch noch von weiteren Paketen abhängt, können Sie das so angeben:

```
#% Do not delete the line below; configure depends on this
#  \DeclareLaTeXClass[article,foo.sty]{Article (with My Own Head-
ings)}
#  \DeclareCategory{Articles}
```

Dadurch wird angezeigt, dass Ihre Klasse auch das Paket `foo.sty` verwendet. Schließlich können Sie auch Klassen für DocBook festlegen. Eine typische Deklaration sieht so aus:

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

Diesen Deklarationen kann außerdem ein optionaler Parameter zugefügt werden, der den Namen der Dokumentenklasse festlegt (hier ist aber keine Liste erlaubt).

Eine Layout-Deklaration hat demnach die Form

```
#  \DeclareLaTeXClass[class,package.sty]{Layout-Beschreibung}
#  \DeclareCategory{Kategorie}
```

Wenn Sie eine Textklasse nach Ihrem Geschmack erstellt haben, müssen Sie die Datei nur noch in das Verzeichnis LyXDir/layouts/ oder nach `UserDir/layouts` kopieren und unter LyX den Menüpunkt Werkzeuge ▷ Neu konfigurieren auswählen. Nach dem Neustart von LyX sollte Ihre Textklasse im Popup-Menü Dokument ▷ Einstellungen auswählbar sein.

5.3.2. Die Modul-Deklaration

Ein Modul muss mit einer Zeile wie die folgende beginnen:

```
#\DeclareLyXModule[endnotes.sty]{Endnotes}
```

Das benötigte Argument in geschweiften Klammern ist der Name des Moduls, wie es in Dokument ▷ Einstellungen ▷ Module erscheinen soll. Das Argument in eckigen Klammern ist optional: Es deklariert alle LaTeX-Pakete, die das Modul benötigt. Es ist außerdem möglich die Form von->zu als optionales Argument zu verwenden, das angibt, dass das Modul nur verwendet werden kann, wenn es eine Konvertierungsmöglichkeit zwischen den Formaten „`von`“ und „`zu`“ gibt.

Die Modul-Deklaration sollte gefolgt werden von Zeilen zur Beschreibung wie den folgenden:[7]

[7] Vorzugsweise in Englisch wenn das Modul als Teil von LyX veröffentlicht werden soll. Diese Beschreibung wird dann in LyXs Liste der zu übersetzenden Zeichenketten erscheinen und übersetzt werden.

```
#DescriptionBegin
#Adds an endnote command, in addition to footnotes.
#You will need to add  \theendnotes in TeX code where you
#want the endnotes to appear.
#DescriptionEnd
#Requires: somemodule | othermodule
#Excludes: badmodule
```

Die Beschreibung wird in DOKUMENT ▷ EINSTELLUNGEN ▷ MODULE verwendet um dem Nutzer zu beschreiben was das Modul macht. Die Zeile mit `Requires` wird verwendet, um andere Module anzugeben, die dieses Modul verwenden muss; die Zeile mit `Excludes` wird verwendet, um Module anzugeben, die mit diesem Modul nicht verwendet werden dürfen. Beide Zeilen sind optional und, wie gezeigt, müssen mehrere Module mit einem „|“ getrennt werden. Beachten Sie dass die benötigten Module disjunktiv behandelt werden: *mindestens eins* der benötigten Module muss verwendet werden. Dementsprechend darf *keines* der ausgeschlossenen Modul verwendet werden. Beachten Sie auch, dass Module durch ihren Dateinamen ohne die Dateiendung `.module` angegeben werden. Daher ist `EinModul` ist in Wirklichkeit `EinModul.module`.

5.3.3. Dateiformat

Die erste Zeile, die kein Kommentar ist, muss die Dateiformatnummer enthalten:

`Format` [`int`] Die Nummer des Formats der Layout-Datei.

Dieser Tag wurde mit LyX 1.4.0 eingeführt. Layout-Dateien älteren LyX-Versionen haben kein explizites Format und werden als `Format 1` behandelt. Das Format dieser LyX-Version ist 60. Aber jede LyX-Version kann ältere Versionen von Layout-Dateien lesen, so wie es ältere LyX-Dateien lesen kann. Es gibt jedoch keine Unterstützung in ältere Formate zu konvertieren.

5.3.4. Allgemeine Parameter für Textklassen

Nachfolgend allgemeine Parameter, die die Form der gesamten Dokumentklasse beschreiben. (Dies bedeutet *nicht* dass sie nur in `.layout`-Dateien und nicht in Modulen erscheinen müssen. Ein Modul kann jeden Layout-Tag enthalten.)

`AddToHTMLPreamble` fügt Informationen hinzu, die im `<head>`-Block ausgegeben werden, wenn das Dokument als XHTML ausgegeben wird. Typischerweise wird dies verwendet werden, um CSS-Stilinformationen auszugeben, aber es kann auch für alles Andere verwendet werden, dass in `<head>` zulässig ist. Muss mit „`EndPreamble`“ beendet werden.

`AddToPreamble` fügt Informationen zum LaTeX-Vorspann hinzu. Muss mit „`EndPreamble`“ beendet werden.

CiteFormat Definiert Formate die in der Anzeige von Bibliographie-Informationen verwendet werden. Siehe Kap. 5.3.12 für Details. Muss mit „**End**“ beendet werden.

ClassOptions Dieser Abschnitt beschreibt verschiedene globale Optionen, die von der Dokumentenklasse unterstützt werden. Eine detaillierte Beschreibung finden Sie in Kap. 5.3.5. Muss mit „**End**“ beendet werden.

Columns [*1*, 2] Gibt an, ob die Textklasse standardmäßig ein- oder zweispaltig gesetzt wird. Kann im Menü DOKUMENT ▷ EINSTELLUNGEN geändert werden.

Counter [string] definiert die Eigenschaften für einen Zähler. Wenn der Zähler noch nicht existiert, wird er erstellt; wenn er bereits existiert, wird er modifiziert. Muss mit „**End**“ beendet werden.
Siehe Kap. 5.3.10 für Details zu Zählern.

DefaultFont Definiert den Standardzeichensatz für die Anzeige des Dokuments. Eine genauere Beschreibung finden Sie in Kap. 5.3.11. Muss mit „**EndFont**“ beendet werden.

DefaultModule [<Modul>] spezifiziert ein Modul, das standardmäßig zu dieser Dokumentklasse hinzugefügt wird. <Modul> ist der Dateiname ohne die Dateiendung `.module`. Der Nutzer kann das Modul zwar immer noch entfernen, aber es bleibt von Beginn an aktiv. (Dies gilt nur für neue Dateien oder wenn diese Klasse für ein existierendes Dokument gewählt wird.)

DefaultStyle [<Stil>] Dies ist das Layout bzw. der Stil, der für neu angelegte Absätze verwendet wird. Normalerweise ist das STANDARD. Fehlt dieser Eintrag, wird das erste definierte Layout verwendet; dennoch ist es ratsam `DefaultStyle` anzugeben.

ExcludesModule [<Modul>] zeigt an, dass das genannte Modul (das durch den Dateinamen ohne die Endung `.module` angegeben wird) in dieser Dokumentklasse nicht benutzt werden kann. Dies könnte in einem Journal-spezifischen Layout benutzt werden, um zum Beispiel die Verwendung des Moduls `theorems-sec` zu verhindern, das Theoreme abschnittsweise nummeriert. Diese Marke darf *nicht* in einem Modul benutzt werden. Module haben ihre eigene Methode andere Module auszuschließen (siehe Kap. 5.2.1).

Float definiert ein neues Gleitobjekt. Siehe Kap. 5.3.8 für Details. Muss mit „**End**“ beendet werden.

HTMLPreamble Informationen, die im `<head>`-Block ausgegeben werden, wenn das Dokument als XHTML ausgegeben wird. Beachten Sie, dass dies jede vorhergehende `HTMLPreamble` oder `AddToHTMLPreamble`-Deklaration überschreibt. (Verwenden Sie `AddToHTMLPreamble` wenn Sie Material zum Vorspann hinzufügen wollen.) Muss mit „**EndPreamble**“ beendet werden.

HTMLTOCSection [<Stil>] Das Layout bzw. der Stil, der für das Inhaltsverzeichnis, das Literaturverzeichnis etc. verwendet werden soll, wenn das Dokument als HTML exportiert wird. Für Artikel sollte dies normalerweise `Section` sein und für Bücher `Chapter`. Wenn es nicht angegeben wird, wird LyX versuchen herauszufinden, welches Layout zu benutzen ist.

IfCounter [<Zähler>] Ändert die Eigenschaften des angegebenen Zählers. Wenn dieser nicht existiert, wird die Anweisung ignoriert. Muss mit „`End`“ beendet werden.
Siehe Kap. 5.3.10 für Details zu Zählern.

Input [<Dateiname>] Hiermit können Sie andere Dateien einbinden, die Definitionen für Textklassen enthalten. Damit können Sie unnötige Mehrfachdefinitionen vermeiden. Beispiele sind die Standard-Layout-Dateien, z. B. `stdclass.inc`, die ein Großteil der Standardlayouts enthalten.

InsetLayout [<Typ>] Dieser Abschnitt definiert das Layout einer Einfügung (neu). Es kann auf eine vorhandene Einfügung angewendet werden oder eine neue, benutzerdefinierte, zum Beispiel einen neuen Zeichenstil. Muss mit „`End`“ beendet werden.
Kap. 5.3.9 enthält weitere Einzelheiten.

LeftMargin [string] ist ein String dessen Länge die Breite des linken Randes festlegt, zum Beispiel „`MMMMM`“.

ModifyStyle [<Stil>] Ändert die Eigenschaften des angegebenen Paragraphstils. Wenn dieser nicht existiert, wird die Anweisung ignoriert. Muss mit „`End`“ beendet werden.

NoCounter [<Zähler>] Löscht einen existierenden Zähler; üblicherweise einen, der in einer eingefügten Datei definiert wurde.

NoFloat [<Gleitobjekt>] Löscht ein vorhandenes Gleitobjekt. Dies ist dann nützlich, wenn Sie ein Gleitobjekt, das in einer eingefügten Datei definiert wurde, nicht verwenden wollen.

NoStyle [<Stil>] Löscht ein existierendes Layout bzw. Stil.

OutputFormat [<Format>] Das Dateiformat (wie es in den LyX-Voreinstellungen definiert ist) das von dieser Dokumentklasse erzeugt wird. Es ist hauptsächlich nützlich wenn `OutputType` auf `literate` gesetzt ist und man einen neuen Typ eines „literate“-Dokuments definieren will. Das Format wird auf „`docbook`“ oder „`latex`“ zurückgesetzt wenn der entsprechende `OutputType`-Parameter gefunden wird.

OutputType [*latex*, `docbook`, `literate`] Gibt an welche Dokumentart diese Klasse erzeugt.

PackageOptions [string string] Der zweite String gibt Optionen für das Paket im ersten String an. Zum Beispiel lädt „PackageOptions natbib square“ natbib mit der Option square. (Für TEXperten: Dies bewirkt, dass LyX \PassOptionsToPackage{natbib
vor dem Laden von natbib ausgibt.)

PageStyle [*plain*, empty, headings] Der Standard-Seitenstil. Kann im Menü DOKUMENT ▷ EINSTELLUNGEN geändert werden.

Preamble Definiert den Vorspann für das LATEX-Dokument. Beachten Sie, dass dies jede vorhergehende Preamble oder AddToPreamble-Deklaration überschreibt. (Verwenden Sie AddToPreamble wenn Sie Material zum Vorspann hinzufügen wollen.) Muss mit „EndPreamble“ beendet werden.

Provides [string] [*0*, 1] zeigt an, ob die Klasse bereits die Funktion string liefert. Eine Funktion ist im Allgemeinen der Name eines Paketes (z. B. amsmath oder makeidx) oder ein Makro (z. B. url oder boldsymbol). Siehe Anhang A für eine Liste der Funktionen.

ProvidesModule [string] zeigt an, dass dieses Layout die Funktionalität des Moduls string anbietet, das als Dateiname ohne die Erweiterung .module angegeben wird. Dies wird typischerweise benutzt, wenn das Layout das Modul direkt benutzt statt den Tag DefaultModule zu benutzen. Es könnte auch in einem Modul benutzt werden, das eine andere Implementation derselben Funktion liefert.

ProvideStyle [<Stil>] Erstellt einen neuen Paragraphstil, falls er noch nicht existiert. Existiert er bereits, wird ProvideStyle ignoriert. Muss mit „End“ beendet werden.

Requires [string] zeigt an, ob die Klasse die Funktion string benötigt. Mehrfache Funktionen müssen durch Komma getrennt werden Beachten Sie, dass Sie nur unterstützte Funktionen anfordern können. (Siehe Anhang A für eine Liste der Funktionen.) Wenn Sie ein Paket mit bestimmten Optionen anfordern müssen, können Sie zusätzlich PackageOptionsverwenden.

RightMargin [string] ist ein String dessen Länge die Breite des rechten Randes festlegt, zum Beispiel „MMMMM“.

SecNumDepth [int=3] legt die Nummerierungstiefe fest; korrespondiert mit dem LATEX-Zähler secnumdepth.

Sides [*1*, 2] Gibt an, ob der Text standardmäßig für ein- oder für zweiseitigen Druck gesetzt wird. Kann im Dialog DOKUMENT ▷ EINSTELLUNGEN geändert werden.

Style [<Name>] definiert einen neuen Absatzstil. Wenn er bereits existiert, werden stattdessen einige seiner Parameter neu definiert. Muss mit „End“ beendet werden.
Siehe Kap. 5.3.6 für mehr über Absatzstile.

TitleLatexName [string="maketitle"] ist der Name des Befehls oder der Umgebung, der für TitleLatexType benutzt werden soll.

TitleLatexType [*CommandAfter*, Environment] gibt an, wie der Dokumenttitel aussehen soll. CommandAfter bedeutet, dass das Makro namens TitleLatexName nach dem letzten Layout mit „InTitle 1“ gesetzt werden soll. Environment ist für den Fall, dass alle Layouts mit „InTitle 1“ in die TitleLatexName-Umgebung gesetzt werden sollen.

TocDepth [int=3] legt fest, bis zu welcher Tiefe das Inhaltsverzeichnis gehen soll; korrespondiert mit dem LaTeX-Zähler tocdepth.

5.3.5. Der Abschnitt ClassOptions

Der Abschnitt ClassOptions kann folgende Einträge enthalten:

FontSize [string="10|11|12"] Eine Liste verfügbarer Größen für den Hauptzeichensatz; die Einträge werden mit „|“ getrennt.

Header wird benutzt, um die DTD-Zeile mit XML-basierten Klassen zu setzen. Zum Beispiel PUBLIC „-//OASIS//DTD DocBook V4.2//EN“.

Other [string=""] Sonstige Optionen für die Dokumentenklasse, die durch Komma getrennt werden. Sie werden in dem \documentclass-Befehl als optionales Argument übergeben.

PageStyle [string="empty|plain|headings|fancy"] Eine Liste verfügbarer Seitenstile; die Einträge werden mit „|“ getrennt.

Der Abschnitt ClassOptions muss mit „End“ beendet werden.

5.3.6. Einzelne Absatz-Layouts

Eine Layoutbeschreibung für einen Absatz sieht wie folgt aus[8]:

```
Style Name
  ...
End
```

Innerhalb des Blocks sind folgende Befehle erlaubt:

Align [*block*, left, right, center] Gibt an, ob der Text im Blocksatz linksbündig, rechtsbündig oder zentriert gesetzt wird.

[8] Sie können mit diesem Ausdruck entweder ein neues Layout definieren oder aber ein bereits definiertes umdefinieren.

AlignPossible [*block*, left, right, center] Eine Liste von möglichen Textausrichtungen, die durch Kommata voneinander getrennt werden. (Einige LaTeX-Stile verbieten bestimmte Ausrichtungen, weil sie keinen Sinn machen. Beispielsweise sollte in einer nummerierten Aufzählung der Text nicht rechtsbündig oder zentriert gesetzt werden.)

Argument [int] Definiert Argument Nummer <int> eines Befehls/einer Umgebung, der/die im aktuellen Stil definiert ist. Dies ist nützlich für Dinge wie Abschnittsüberschriften. Jedes Argument (optional oder erforderlich) eines Befehls oder einer Umgebung hat eine eigene Definition (ausgenommen das erforderliche Haupt-Argument des Absatzes). Die Nummer gibt die Reihenfolge des Arguments an. Die Definition muss mit `EndArgument` enden. Ein Befehl mit 2 optionalen Argumenten hat somit diese Struktur:

```
Argument 1

...

EndArgument

Argument 2

...

EndArgument
```

Innerhalb einer `Argument`-Definition sind die folgenden Spezifikationen möglich:

- `LabelString [string]` The string that will appear both in the menu (to insert this argument) and on the argument inset button (unless you also specify a separate `MenuString`). For the menu, you can define an accelerator by appending the respective character to the string, divided by „|“ (z. B. „`Short Title|S`“).
- `MenuString [string]` A separate string for the menu. You can define an accelerator by appending the respective character to the string, divided by „|“ (z. B. „`Short Title|S`“). This specification is optional. If it is not given the `LabelString` will be used instead for the menu.
- `Tooltip [string]` A longer explanatory text that appears in the tooltip when hovering over the argument inset.
- `Mandatory` [*0*, 1] Declare if this is a mandatory (1) or an optional (0) argument. Mandatory arguments will be output empty if not given, while optional arguments will not be output at all. By default, mandatory arguments are delimited by `{...}`, while optional arguments are delimited by `[...]`
- `Requires [int=0]` Defines another argument (by its number) which this argument requires to be output if it is itself output. E. g., in LaTeX commands, optional arguments often require previous optional arguments to

be output (at least empty), as in `\command[][Argument]{Text}`. This can be achieved by the statement `Requires 1` innerhalb von `Argument 2`.

- `LeftDelim [string]` Definiert ein eigenes linkes Begrenzungszeichen (satt `{` oder `[`). Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.
- `RightDelim [string]` Definiert ein eigenes rechtes Begrenzungszeichen (statt `}` oder `]`). Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.
- `DefaultArg [string]` Definiert ein Argument, das nur eingefügt wird, wenn der Nutzer kein Argument eingefügt hat. Das heißt, wenn keine Argument-Einfügung eingefügt wurde oder sie eingefügt wurde aber leer ist. Mehrere Argumente werden durch Kommas getrennt.
- `PresetArg [string]` Definiert ein Argument, das in jedem Fall eingefügt wird (allein oder zusätzlich zu benutzerdefinierten Argumenten). Mehrere Argumente werden durch Kommas getrennt.
- `Font` Die Schrift, die für den Argumentinhalt verwendet wird; siehe Kap. 5.3.11.
- `LabelFont` Die Schrift, die für die Marke verwendet wird; siehe Kap. 5.3.11.
- `Decoration` [*`Classic`*, `Minimalistic, Conglomerate`] legt den Anzeigestil für den Rahmen und Knopf der Einfügung fest.
- `AutoInsert [int=0]` If this is set to `1`, this argument is automatically inserted when the respective style is selected. Currently, only one argument per style/layout can be automatically inserted.
- `InsertCotext [int=0]` If this is set to `1`, this argument will be inserted with a copy of the co-text (either selected text or the whole paragraph) as content.
- `PassThruChars` [`string of characters`] Defines individual characters that should be output in raw form, meaning without special translations that LaTeX would require. Note that, contrary to PassThru, this needs to be explicitly defined for arguments. That is, arguments do not inherit PassThruChars from their parent inset or layout.

By default, the text entered in the LyX workarea in the respective layout is the last (mandatory) argument of a command if the `LatexType` is `Command`. However, arguments with the prefix `post:` are output after this workarea argument. Note that post-argument numbering restarts at 1, so the first argument following the workarea argument is `post:1`. Post-Argumente werden in allen anderen `LatexType` außer `Command` ignoriert.

Argumente für Listen-`\item`s (wie in `\item[foo]`) haben den Präfix `item:` gefolgt von der Nummer (z. B. `Argument item:1`)

`BabelPreamble` Beachten Sie, dass dies alle vorhergehenden `BabelPreamble`-Deklaration für diesen Stil überschreibt. Muss mit „`EndBabelPreamble`“ beendet werden. Siehe Kap. 5.3.7 für Details zur Verwendung.

BottomSep [float=0][9] Der vertikale Abstand, der die letzte Serie von Absätzen vom folgenden Text trennt. Wenn der nächste Paragraph einen anderen Stil hat, werden die Abstände nicht einfach addiert, sondern das Maximum wird verwendet.

Category [string] ist die Kategorie für diesen Stil. Sie wird benutzt, um zugehörige Stile in der Layout-Kombobox der Werkzeugleiste zu gruppieren. Jeder beliebige String kann benutzt werden, aber es ist sinnvoll vorhandene Kategorien zusammen mit Ihren eigenen Stilen zu benutzen.

CommandDepth ist die Tiefe des XML-Befehls und wird nur für XML-Formate benutzt.

CopyStyle [string] Kopiert alle Eigenschaften eines bereits definierten Layouts in das aktuelle.

DependsOn [<Name>] ist der Name eines Stils, dessen Vorspann *vor* diesem ausgegeben werden soll. Dadurch wird eine Reihenfolge von Vorspannteilen bewirkt, wenn Makro-Definitionen voneinander abhängen.[10]

EndLabeltype [*No_Label*, Box, Filled_Box, Static] ist der Markentyp, der am Ende eines Absatzes steht (oder mehrerer Absätze, wenn LatexType auf Environment, Item_Environment oder List_Environment gesetzt ist). No_Label bedeutet „nichts“, Box oder Filled_Box ist ein weißes oder schwarzes Quadrat, das für das Markieren eines Beweisendes geeignet ist. Static ist eine explizite Zeichenkette.

EndLabelString [string=""] ist eine Zeichenkette, die für einen Static EndLabelType benutzt wird.

Font Der Zeichensatz, der für den Textkörper *und* die Marke verwendet wird, siehe Kap. 5.3.11. Wird Font gesetzt, dann erhält LabelFont automatisch denselben Wert. Daher sollte Font zuerst definiert werden.

ForceLocal [int=0] Wird benutzt um neue Stile für stabile LyX-Versionen zu konvertieren. Die erste stabile Version, die das unterstützt ist LyX 2.1.0. Das Argument ist eine Nummer, die entweder 0, -1 oder irgend eine Zahl größer Null sein kann. Wenn ForceLocal eines Stils größer als Null ist, wird er immer in den Dokumentkopf geschrieben. Wenn eine .lyx-Datei gelesen wird, werden die Stil-Definitionen aus dem Dokumentkopf zur Dokumentklasse hinzugefügt. Dadurch können sogar ältere LyX-Versionen den Stil handhaben. Das Argument von ForceLocal ist eine Versionsnummer: Wenn der Stil gelesen wird, und die

[9] „float“ ist eine Gleitkommazahl, wie „1,5“.

[10] Beachten Sie, dass es außer dieser Funktionalität keine andere Möglichkeit gibt, Vorspanne zu ordnen. Die Reihenfolge, die Sie in einer LyX-Version sehen, kann sich in späteren Versionen ohne Warnung ändern.

Versionsnummer ist kleiner als die Versionsnummer des bereits existierenden Stils der Dokumentklasse, wird der neue Stil ignoriert. Wenn die Versionsnummer größer ist, ersetzt der neue Stil den bestehenden. Der Wert -1 steht für eine unendliche Versionsnummer, das heißt der Stil wird immer benutzt.

FreeSpacing [*0*, 1] Normalerweise erlaubt es LyX nicht, mehr als ein Leerzeichen zwischen Wörtern einzufügen. Diese Eigenschaft kann in bestimmten Fällen umständlich sein, zum Beispiel, wenn ein Programmcode eingegeben werden soll. In solchen Fällen kann `FreeSpacing` auf `1` gesetzt werden. LyX erzeugt in diesem Falls sich LyX nicht im LaTeX-Modus befindet, erzeugt es für jedes zusätzliche Leerzeichen ein geschütztes Leerzeichen.

HTML* Diese Tags kontrollieren die XHTML-Ausgabe. Siehe Kap. 5.4.

InnerTag [FIXME] (Wird nur für XML-Formate benutzt.)

InPreamble [*0*, 1]Wenn auf `1`gesetzt, wird der Stil in den LaTeX-Vorspann gesetzt und nicht in den eigentlichen Dokumenttext. Dies ist nützlich für Dokumentklassen, die Informationen wie den Titel und Autor im Vorspann erwarten. Beachten Sie, dass dies nur für Stile funktioniert, deren `LatexType Command` oder `Paragraph`ist.

InTitle [*0*, 1] Wenn auf `1`gesetzt, wird der Stil als Teil des Titel-Abschnitts behandelt (siehe auch die allgemeinen Textklassen-Parameter `TitleLatexType` und `TitleLatexName`).

ItemCommand [`string="item"`] Der LaTeX-Befehl, der ein Item in einer Liste definiert. Dieser Befehl muss ohne den Backslash am Anfang angegeben werden (die Voreinstellung ist "item", was in der LaTeX-Ausgabe `\item` zur Folge hat).

ItemSep [`float=0`] Ein zusätzlicher Abstand zwischen Absätzen desselben Layouts. Wenn in einer Umgebung andere Layouts integriert werden, so werden diese mit dem `ParSep` der Umgebung getrennt. Die kompletten Unterpunkte der Umgebung werden jedoch *zusätzlich* mit `ItemSep` getrennt. Man beachte, dass `ItemSep`ein *Multiplikator* ist.

ItemTag [FIXME] (Wird nur für XML-Formate benutzt.)

KeepEmpty [*0*, 1] Normalerweise ist es in LyX nicht möglich, einen Absatz leer zu lassen, da das zu einer leeren LaTeX-Ausgabe führen würde. In manchen Fällen ist das aber durchaus gewünscht: So können beispielsweise in einer Briefvorlage die benötigten Felder leer voreingestellt werden, damit keiner vergisst, sie anzugeben; in speziellen Klassen wird ein Absatz als Unterbrechung verwendet, der keinen Text enthält.

LabelBottomsep [`float=0`] Der vertikale Abstand zwischen der Marke und dem folgenden Text. Wird nur für Marken benutzt, die über dem folgenden Text stehen (`Top_Environment` und `Centered_Top_Environment`).

LabelCounter [string=""] ist der Name des Zählers zur automatischen Nummerierung. Um den Zähler einer Marke zuzuordnen, muss er im LabelString referenziert werden. Dies funktioniert zumindest mit LabelTypeen, Static, Above und Centered.
Er *kann* angegeben werden, wenn LabelType Enumerate ist. In diesem Fall ist es etwas kompliziert: Angenommen Sie haben "LabelCounter MeinZaehler" angegeben, dann lauten die eigentlichen Zähler MeinZaehleri, MeinZaehlerii, MeinZaehleriii und MeinZaehleriv; so wie in LaTeX. Diese Zähler müssen alle separat deklariert werden.
Siehe Kap. 5.3.10 für Einzelheiten zu Zählern.

LabelFont Der Zeichensatz, der für die Marke verwendet wird. Siehe Kap. 5.3.11.

LabelIndent [string=""] Text der angibt, wie weit die Marke eingerückt werden soll.

Labelsep [string=""] Text der den horizontalen Abstand zwischen der Marke und dem folgenden Text angibt. Wird nur für Marken benutzt, die nicht über dem folgenden Text stehen.

LabelString [string=""] Der String, der für den LabelType Static verwendet wird. Wenn LabelCounter gesetzt wurde, kann der String spezielle Formatierungsbefehle enthalten, wie sie in Kap. 5.3.10 beschrieben sind.

LabelStringAppendix [string=""] wird im Anhang anstatt LabelString benutzt. Beachten Sie, dass jede Definition von LabelString auch LabelStringAppendix zurücksetzt.

LabelTag [FIXME] (Wird nur für XML-Formate benutzt.)

LabelType [*No_Label*, Manual, Static, Above, Centered, Sensitive, Enumerate, Itemize, Bibliography]

Manual bedeutet: die Marke ist das erste Wort (bis zum ersten echten Leerzeichen). Verwenden Sie geschützte Leerzeichen wenn Sie mehr als ein Wort als Marke haben wollen.

Static bedeutet: die Marke ist was als LabelString definiert wurde. Die Marke wird „inline" zu Beginn des Absatzes angezeigt. Wenn der LatexType Environment ist, wird sie nur im ersten Absatz von aufeinanderfolgenden Absätzen mit demselben Style. angezeigt.

Above und Centered sind Spezialfälle von Static. Die Marke erscheint über dem Absatz, entweder am Anfang der Zeile oder zentriert.

Sensitive ist ein Spezialfall für Beschriftungsmarken für Abbildungen und Tabellen-Gleitobjekte. Sensitive bedeutet, dass der gedruckte Text von

der Art des Gleitobjekts abhängt: Er ist fest einprogrammiert als 'GleitobjektTyp N', wobei N der Wert des Zählers des Gleitobjekttyps ist. Für den Fall, dass die Beschriftungsmarke außerhalb eines Gleitobjekts eingefügt wird, erscheint der `LabelString` als „Sinnlos!“.

Enumerate erzeugt die üblichen Marken für Nummerierungen. Momentan sind diese fest auf arabische Zahlen, Kleinbuchstaben, kleine römische Zahlen und Großbuchstaben (für die 4 möglichen Schachtelungstiefen) programmiert.

Itemize erzeugt je nach Schachtelungstiefe verschiedene Auflistungszeichen, Diese sind ebenfalls fest programmiert.

Bibliography sollte nur zusammen mit `LatexType BibEnvironment` verwendet werden.

LangPreamble Beachten Sie, dass dies alle vorhergehenden `LangPreamble`-Deklaration für diesen Stil überschreibt. Muss mit „`EndLangPreamble`“ beendet werden. Siehe Kap. 5.3.7 für Details zur Verwendung.

LatexName [`<Name>`] Der LaTeX-Name für dieses Layout. Das bedeutet entweder der Name eines LaTeX-Befehls oder der einer LaTeX-Umgebung.

LatexParam [`<Parameter>`] Ein optionaler Parameter für den entsprechenden `LatexName`. Dieser Parameter kann innerhalb von LyX nicht mehr geändert werden (man verwendet `Argument` für anpassbare Parameters). Dieser wird nach allen anderen LaTeX-`Argument`en ausgegeben.

LatexType [*Paragraph*, `Command, Environment, Item_Environment, List_Environment, Bib_Environment`] Legt fest, wie das Layout in LaTeX übersetzt wird.[11]

Paragraph bewirkt nichts besonderes – der Text wird als *normaler Absatz* übernommen.

Command behandelt den Text als Argument eines LaTeX-Befehls (`\`*LatexName*`{...}`).

Environment behandelt den Text als Kern einer LaTeX-Umgebung (`\begin{`*LatexName*`}...\end{`*LatexName*`}`).

Item_Environment bewirkt dasselbe wie `Environment`, nur dass vor jedem Absatz ein `\item` eingefügt wird.

List_Environment funktioniert wie `Item_Environment`, nur dass `LabelWidthString` als Argument an die Umgebung übergeben wird. `LabelWidthString` kann im Menü Bearbeiten ▷ Absatz-Einstellungen definiert werden.

[11] `LatexType` mag irreführend sein, denn dessen Regeln gelten auch für DocBook-Klassen. Siehe die DocBook Klassendateien (Dateinames `db_*.inc`) für spezielle Beispiele.

Bib_Environment ist wie Environment aber fügt zusätzlich das notwendige Argument (die längste Marke) zum Begin-Befehl der Bibliografie-Umgebung ein:
`\begin{thebibliography}{99}` Es ist daher nur für die Bibliografie-Umgebung nützlich. Die voreingestellte längste Marke „99“ kann vom Nutzer in den Absatzeinstellungen eines Bibliografie-Eintrags geändert werden.

Fasst man die letzten Sachen zusammen, wird die LaTeX-Ausgabe entweder so:

```
\LatexName[LatexParam]{...}
```

oder so:

```
\begin{LatexName}[LatexParam] ... \end{LatexName}.
```

aussehen, abhängig vom LaTeX-Typ.

LeftDelim [string] Eine Zeichenkette, die zu Beginn des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

LeftMargin [string=""] Wenn ein Layout in ein anderes Layout für Umgebungen eingefügt wird, werden die Breiten der verschiedenen LeftMargin nicht einfach addiert, sondern vorher in Abhängigkeit zur Schachtelungstiefe mit dem Faktor $\frac{4}{\text{Tiefe}+4}$ multipliziert. Dieser Parameter wird auch dann benutzt, wenn Margin als Manual oder Dynamic definiert wurde. In diesem Fall wird der Wert zu den gegebenen manuellen oder dynamischen Rändern hinzugefügt.
Zum Beispiel bedeutet „MM“, dass der Absatz mit der Breite eingerückt wird, die die Buchstaben „MM“ in der normalen Schriftart haben. man kann negative Breite erzeugen, indem man den String mit „-“ beginnt. Diese Art der Angabe wurde gewählt, damit der Text unabhängig von der verwendeten Bildschirmschriftart wie vorgesehen aussieht.

Margin [*Static*, Manual, Dynamic, First_Dynamic, Right_Address_Box] legt fest, wie der linke Rand des Textes bestimmt wird.

Static wählt feste Randbreiten.

Manual bedeutet, dass der Rand von der Einstellung der Ausrichtung im Menü Bearbeiten ▷ Absatz-Einstellungen abhängt. Dies wird für hübsche Listen ohne Tabulatoren benutzt.

Dynamic bedeutet, der linke Rand hängt von der Größe der verwendeten Markierung ab. Dies wird zum Beispiel bei automatisch nummerierten Überschriften verwendet. Es leuchtet ein, dass die Überschrift „5.4.3.2.1 Sehr lange ... Überschrift“ einen größeren linken Rand benötigt, als „3.2 Sehr lange ... Überschrift“.

First_Dynamic arbeitet ähnlich wie Dynamic, aber nur die erste Zeile wird dynamisch gesetzt, die anderen statisch. Dies wird für die LaTeX-Umgebung description benutzt.

`Right_Address_Box` bedeutet, dass der Rand so gewählt wird, dass die längste Zeile des Absatzes gerade den rechten Rand berührt. Dies wird zum Setzen einer Adresse am rechten Rand der Seite eingesetzt.

`NeedProtect` [*0*, 1] Gibt an, ob „zerbrechliche“ LaTeX-Befehle innerhalb dieses Layouts durch `\protect` geschützt werden müssen. (Achtung: Diese Einstellung sagt nichts darüber aus, ob der Befehl an sich geschützt werden soll.)

`Newline` [0, *1*] Gibt an, ob Zeilenumbrüche in LaTeX als „\\“ dargestellt werden, oder nicht. Man kann dies ausschalten (Wert: `0`), um TeX-Code in LyX komfortabler editieren zu können.

`NextNoIndent` [*0*, 1] Gibt an, ob der nachfolgende Absatz einen linken Einzug haben darf oder nicht. `1` heißt, der Absatz erhält auf keinen Fall einen Einzug (z. B. nach einer Überschrift), wenn `DefaultStyle`- (normalerweise `Standard`-) Paragraphen einen Einzug haben. (Daher beeinflusst die Einstellung nur `Standard`-Paragraphen.)

`ObsoletedBy` [`<Name>`] Der Name eines Layouts, das durch dieses ersetzt wurde. So können Sie ein Layout umbenennen und die Rückwärtskompatibilität erhalten.

`ParagraphGroup` [0, 1] Legt fest ob aufeinanderfolgende Absätze desselben Typs als zusammengehörend behandelt werden. Das hat den Effekt, dass `GuiLabel` nur einmalig vor einer solchen Gruppe ausgegeben wird. Dies ist standardmäßig der Fall für `LaTeXType Environment` und `Bib_Environment` und nicht der Fall für alle anderen Typen.

`ParbreakIsNewline` [*0*, 1] Gibt an, dass ein Paragraph nicht durch eine leere Zeile in der LaTeX-Ausgabe abgesetzt wird, sondern nur durch einen Zeilenumbruch. Zusammen mit `PassThru 1` erlaubt dies die Emulation eines reinen Texteditors (so wie die TeX-Code Einfügung).

`ParIndent` [`string=""`] Der Einzug der ersten Zeile eines Absatzes. `Parindent` bleibt für ein bestimmtes Layout fest. Eine Ausnahme ist das Standard-Layout, denn dort kann der Einzug vom vorherigen Layout mit `NextNoIndent` verboten werden. Außerdem benutzt das `Standard`-Layout innerhalb von Umgebungen den `Parindent` der Umgebung und nicht den eigenen. Zum Beispiel haben `Standard`-Absätze innerhalb einer Aufzählung keinen Einzug.

`ParSep` [`float=0`] Der vertikale Anstand zwischen den Absätzen dieses Layouts.

`Parskip` [`float=0`] Der Benutzer kann in LyX wählen ob Absätze durch Einrückung oder Vertikaler Abstand getrennt werden. Wenn Einrückung gewählt ist, wird `Parskip` ignoriert. Ist Vertikaler Abstand gewählt wird `ParIndent` ignoriert und alle Absätze durch den vertikalen Abstand von `Parskip` getrennt. Die Länge dieses Abstands berechnet sich mit `Parskip * DefaultHeight` wobei

`DefaultHeight` die Höhe einer Zeile in der normalen Schrift ist. Dadurch bleibt das Aussehen mit verschiedenen Schriften gleich.

PassThru [*0*, 1] Legt fest, ob der Absatzinhalt unverändert ausgegeben werden soll, also ohne diverse von LaTeX benötigte Ersetzungen durchzuführen.

PassThruChars [string] Definiert Zeichen, die unverändert ausgegeben werden sollen. Das bedeutet, dass sie nicht in einen LaTeX-Befehl übersetzt werden, falls das normalerweise der Fall wäre.

Preamble Befehle und Definitionen, die in die Präambel (vor `\begin{document}`) eingefügt werden, wenn dieses Layout benutzt wird. Kann verwendet werden um Pakete zu laden, Makros zu definieren usw.. Muss mit „`EndPreamble`“ beendet werden.

RefPrefix [string] Der Präfix, der verwendet werden soll, wenn auf Marken dieses Absatzes verwiesen wird. Dies erlaubt die Verwendung von Formatierten Querverweisen.

Requires [string] legt fest, dass das Layout die Funktion `string` benötigt (siehe Anhang A für eine List der Funktionen). Wenn Sie ein Paket mit bestimmten Optionen anfordern müssen, können Sie zusätzlich `PackageOptions`als allgemeiner Textklassen-Parameter verwenden (siehe Kap. 5.3.4).

ResetArgs [*0*,1] Setzt die LaTeX-Argumente dieses Stils zurück (der via `Argument` definiert wurde). Dies ist nützlich, wenn man einen Stil mit `CopyStyle` kopiert hat, aber nicht dessen (benötigten und optionalen) Argumente übernehmen will.

RightDelim [string] Eine Zeichenkette, die am Ende des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

RightMargin [string=""] Ähnlich wie `LeftMargin`.

Spacing [*single*, onehalf, double, other <Wert>] Dies definiert die Voreinstellung für den Zeilenabstand des Layouts. Die Argumente *single*, `onehalf` und `double` entsprechen den Multiplikatoren 1, 1.25 und 1.667. Wenn Sie als Argument `other` angeben, müssen Sie als *Wert* einen konkreten Multiplikator angeben. Im Gegensatz zu anderen Parametern erzeugt LyX, wenn `Spacing` gesetzt wird, spezifischen LaTeX-Code, der das LaTeX-Paket `setspace` verwendet.

Spellcheck [0, *1*] Erlaubt es, den Inhalt des Absatzes auf Rechtschreibung zu überprüfen.

TextFont Der Zeichensatz, der für den Textkörper verwendet wird.
Siehe Kap. 5.3.11.

`TocLevel` [`int=3`] ist die Stufe des Stils im Inhaltsverzeichnis und wird zur automatischen Nummerierung von Abschnittsüberschriften benutzt.

`ToggleIndent` [*`default`*, `always`, `never`] Dieser Befehl legt fest, ob die Einrückung der ersten Absatzzeile über den Absatz-Dialog ein/ausgeschaltet werden kann. Wenn *`default`* gesetzt ist, kann umgeschaltet werden, wenn in den Dokumenteinstellungen für die Absatztrennung „Einrückung“ gewählt ist, Mit `always` kann immer umgeschaltet werden, unabhängig von den Dokumenteinstellungen. Mit `never`kann nie umgeschaltet werden.

`TopSep` [`float=0`] Der vertikale Abstand, der die erste Serie von Absätzen vom vorangehenden Text trennt.

5.3.7. Internationalisierung von Absatz-Stilen

If a `Style` defines text that is to appear in the typeset document, it may use `LangPreamble` and `BabelPreamble` to support non-English and even multi-language documents correctly. The following excerpt (from the `theorems-ams.inc` file) shows how this works:

```
Preamble
  \theoremstyle{remark}
  \newtheorem{claim}[thm]{\protect\claimname}
EndPreamble
LangPreamble
  \providecommand{\claimname}{_(Claim)}
EndLangPreamble
BabelPreamble
    \addto\captions$$lang{\renewcommand{\claimname}{_(Claim)}}
EndBabelPreamble
```

In principle, any legal LaTeX may appear in the `LangPreamble` and `BabelPreamble` tags, but in practice they will typically look as they do here. The key to correct translation of the typeset text is the definition of the LaTeX command `\claimname` and its use in `\newtheorem`.

The `LangPreamble` tag provides for internationalization based upon the overall language of the document. The contents of the tag will be included in the preamble, just as with the `Preamble` tag. What makes it special is the use of the “function” `_()`, which will be replaced, when LyX produces LaTeX output, with the translation of its argument into the document language.

The `BabelPreamble` tag is more complex, since it is meant to provide support for multi-language documents and so offers an interface to the `babel` package. Its contents will be added to the preamble once for each language that appears in the document. In this case, the argument to `_()` will be replaced with its translation into

the language in question; the expression `$$lang` is replaced by the language name (as used by the babel package).

A German document that also included a French section would thus have the following in the preamble:

```
\addto\captionsfrench{\renewcommand{\claimname}{Affirmation}}
\addto\captionsngerman{\renewcommand{\claimname}{Behauptung}}
\providecommand{\claimname}{Behauptung}
```

LaTeX and `babel` will then conspire to produce the correct text in the output.

One important point to note here is that the translations are provided by LyX itself, through the file `layouttranslations`. This means, in effect, that `LangPreamble` and `BabelPreamble` are really only of use in layout files that are provided with LyX, since text entered in user-created layout files will not be seen by LyX's internationalization routines unless the `layouttranslations` file is modified accordingly. That said, however, any layout created with the intention that it will be included with LyX should use these tags where appropriate. Please note that the paragraph style translations provided by LyX will never change with a minor update (e. g. from version 2.1.x to 2.1.y). It is however quite likely that a major update (e. g. from 2.0.x to 2.1.y) will introduce new translations or corrections.

5.3.8. Gleitobjekte

Es ist nötig Gleitobjekte (Abbildung, Tabelle,...) in der Textklasse selber zu definieren. Standardgleitobjekte sind in der Datei `stdfloats.inc` enthalten, so dass Sie sie nur noch

```
Input stdfloats.inc
```

zu Ihrer Layoutdatei hinzufügen müssen. Wenn Sie eine Textklasse implementieren wollen, die andere Gleitobjekttypen enthält (wie zum Beispiel die AGU-Klasse), werden Ihnen die folgenden Informationen helfen:

`AllowedPlacement` [`string`=!htbpH] Erlaubte Platzierungsoptionen für den Gleitobjekttyp. Der Wert ist eine Zeichenkette aus Platzierungszeichen. Mögliche Zeichen sind: *h* („hier wenn möglich“), *t* („oben auf Seite“), *b* („unten auf Seite“), *p* („auf Seite nur mit Gleitobjekten“), *H* („hier, unbedingt“) und *!* („ignoriere LaTeX-Regeln”). Die Reihenfolge der Zeichen in der Zeichenkette ist egal. Wenn keine Platzierungsoptionen erlaubt sind, verwendet man stattdessen die Zeichenkette *none*.

`AllowsSideways` [`0`, *`1`*] Definiert ob das Gleitobjekt mit Hilfe des LaTeX-Pakets `rotfloat` (sidewaysfloat) rotiert werden kann. Falls das nicht der Fall ist, setzt man es auf `0`.

AllowsWide [0, *1*] Definiert ob das Gleitobjekt eine „gesternte“ Version hat, die in einem zweispaltigen Dokument die komplette Seitenbreite einnimmt. Falls das nicht der Fall ist, setzt man es auf `0`.

Extension [string=""] Die Dateinamenserweiterung einer zusätzlichen Datei für das Abbildungsverzeichnis (oder andere). LaTeX schreibt die Beschriftungen in diese Datei.

GuiName [string=""] Die Zeichenkette, die in den Menüs und für die Beschriftung benutzt wird. Dies wird in die aktuelle Sprache übersetzt, wenn babel verwendet wird.

HTML* Diese Tags kontrollieren die XHTML-Ausgabe. Siehe Kap. 5.4.

IsPredefined [*0*, 1] Gibt an, ob das Gleitobjekt bereits in der Dokumentklasse definiert ist oder ob das LaTeX-Paket `float` geladen werden muss, um es zu definieren. Die Voreinstellung ist `0`, was bedeutet, dass `float` verwendet wird. Es sollte auf `1` gesetzt werden, wenn das Gleitobjekt bereits in der Dokumentklasse definiert ist.

ListCommand [string=""] Der Befehl der verwendet wird, um eine Liste der Gleitobjekte dieses Typs zu generieren; das '\' muss weggelassen werden. Der Befehl *muss* angegeben werden, wenn `UsesFloatPkg` auf `0` gesetzt ist, da es sonst keine Möglichkeit gibt, diesen Befehl zu erstellen. Er wird ignoriert, falls `UsesFloatPkg` auf `1` gesetzt ist, da es dann eine Möglichkeit gibt.

ListName [string=""] Die Überschrift für das Gleitobjekt-Verzeichnis (z. B. „Abbildungsverzeichnis“). Sie wird für die Bildschirmmarke in LyX verwendet, von LaTeX für den Titel verwendet und als Titel in der XHTML-Ausgabe. Sie wird in die Dokumentsprache übersetzt.

NumberWithin [string=""] Dieses optionale Argument bestimmt, ob Gleitobjekte dieser Klasse mit der Abschnittsnummer dieses Dokuments nummeriert werden. Wenn zum Beispiel `NumberWithin` auf „`chapter`“ gesetzt ist, werden die Gleitobjekte mit den Kapitelnummern nummeriert.

Placement [string=""] Die Standardplatzierung für die Gleitobjektklasse. `string` sollte die Standard-LaTeX-Werte `t`, `b`, `p` und `h` für oben, unten, Seite und hier enthalten.[12] Zusätzlich gibt es den neuen Typ `H`, der nicht wirklich für ein Gleitobjekt steht, denn er bedeutet: drucke es *hier* und nirgendwo sonst. Beachten Sie, dass `H` besonders ist und wegen der Implementierungsdetails nicht bei nicht-eingebauten Gleitobjekttypen benutzt werden kann. Wenn Sie die Platzierung nicht verstehen, benutzen Sie einfach „`tbp`“.

[12] Wie in LaTeX ist die Reihenfolge der Buchstaben unerheblich.

RefPrefix [string] Der Präfix, der verwendet werden soll, wenn auf Marken dieser Gleitobjekte verwiesen wird. Dies erlaubt die Verwendung von Formatierten Querverweisen. Man kann den RefPrefix eines kopierten Stils entfernen, indem string auf „OFF“ gesetzt wird.

Style [string=""] ist der Gleitobjektstil, wenn er mit \newfloat definiert wird.

Type [string=""] ist der „Typ“ der neuen Gleitobjektklasse, wie z. B. Programm oder Algorithmus. Nach dem entsprechenden \newfloat stehen Befehle wie \begin{program} oder \end{algorithm*} zur Verfügung.

UsesFloatPkg [0, *1*] Gibt an, ob dieses Gleitobjekt mit Hilfe des LaTeX-Pakets float definiert wurde, entweder durch die Dokumentklassen, ein anderes Paket oder durch LyX.

Anmerkung: Wenn ein Gleitobjekt vom Typ *type* definiert wurde, gibt es automatisch einen dazugehörigen Zähler namens *type*.

5.3.9. Flexible Einfügungen und InsetLayout

Es gibt drei Arten von flexiblen Einfügungen:

- Zeichenstil (CharStyle): diese definieren semantische Textauszeichnungen, die mit LaTeX-Befehlen wie \noun oder \code korrespondieren.
- benutzerdefiniert (Custom): diese können benutzt werden, um benutzerdefinierte einklappbare Einfügungen zu definieren, ähnlich wie TeX-Code, Fußnote usw. Ein naheliegendes Beispiel ist die Endnotiz, die im endnote-Modul definiert ist.
- XML-Element (Element): diese werden mit DocBook-Klassen benutzt.

Flexible Einfügungen werden mit der InsetLayout-Marke definiert, die weiter unten erklärt wird.

Die InsetLayout-Marke besitzt noch eine andere Funktion: sie kann benutzt werden, um das allgemeine Aussehen vieler verschiedener Einfügungstypen anzupassen. Zurzeit kann InsetLayout benutzt werden, um die Layout-Parameter für Fußnoten, Randnoten, eingefügten Noten, TeX-Code (ERT), Zweige, Stichwortverzeichnisse, Boxen, Tabellen, Algorithmen, URLs und Legenden anzupassen, ebenso um flexible Einfügungen zu definieren.

Die InsetLayout-Definition muss mit folgender Zeile beginnen:

```
InsetLayout <Typ>
```

Hier bezeichnet <Typ> die Einfügung, deren Layout definiert wird. Es gibt vier Möglichkeiten.

1. Das Layout für eine existierende Einfügung wird geändert. In diesem Fall kann `<Typ>` folgendes sein: `Algorithm`, `Branch`, `Box`, `Box:shaded`, `Caption:Standard`, `ERT`, `Figure`, `Foot`, `Index`, `Info`, `Info:menu`, `Info:shortcut`, `Info:shortcuts`, `Listings`, `Marginal`, `Note:Comment`, `Note:Note`, `Note:Greyedout`, `Table`, oder `URL`.

2. Das Layout für eine flexible Einfügung wird definiert. In diesem Fall muss `<Typ>` in der Form „`Flex:<Name>`“ sein, wobei `Name` ein beliebiger gültiger Bezeichner sein kann, der in keiner anderen existierenden Einfügung benutzt wird. Der Bezeichner darf Leerzeichen enthalten, dann muss aber der komplette Typ in Anführungszeichen gesetzt werden. Beachten Sie, dass die Definition einer flexiblen Einfügung *auch* einen `LyXType`-Eintrag enthalten muss, der festlegt welcher Einfügungstyp definiert wird.

3. The layout for user specific branch is being defined. In this case, `<Typ>` must be of the form „`Branch:<Name>`“, where `name` may be be any valid identifier of branch defined in user's document. The identifier may include spaces, but in that case the whole thing must be wrapped in quotes. The main purpose of this feature is to allow LaTeX wrapping around specific branches as user needs.

4. The layout of a user (or class) specific caption is being defined. In this case, `<Typ>` must be of the form „`Caption:<Name>`“, where `name` specifies the name of the caption as it appears in the menu. Have a look at the standard caption (`Caption:Standard`), the specific captions of the KOMA-Script classes (`Caption:Above`, `Caption:Below`) oder das Modul Multilingual Captions (`Caption:Bicaption`
for applications.

Die `InsetLayout`-Definition kann folgende Einträge enthalten:

`Argument` [`int`] Definiert die Argumentnummer eines Befehls/einer Umgebung, die im aktuellen Layout definiert ist. Die Definition muss mit `EndArgument` beendet werden. Siehe Kap. 5.3.6 für Details.

`BabelPreamble` Präambel um Sprachbefehle zu modifizieren; siehe Kap. 5.3.7.

`BgColor` [`<Name>`] ist die Hintergrundfarbe der Einfügung. Siehe Anhang B für eine Liste von verfügbaren Farbnamen.

`ContentAsLabel` [*0*, 1] Ob der Inhalt der Einfügung als Marke verwendet werden soll, wenn die Einfügung geschlossen ist.

`CopyStyle` [`<Typ>`] Wie bei Absatz-Layouts, siehe Kap. 5.3.6. Beachten Sie, dass der komplette Typ angegeben werden muss, z. B. `CopyStyle Flex:<Name>`.

`CustomPars` [*0*, 1] zeigt an, ob der Benutzer den Absatzeinstellungen-Dialog benutzen darf.

Decoration kann `Classic`, `Minimalistic`, oder `Conglomerate` sein. Es beschreibt den Rendering-Stil für den Einfügungsrahmen und die -knöpfe. Fußnoten benutzen im allgemeinen `Classic`, TEX-Code `Minimalistic` und Zeichenstile `Conglomerate`.

Display [0, *1*] Nur sinnvoll wenn der `LatexType Environment` ist. Gibt an, ob die Umgebung in der Ausgabe abgesetzt erscheint oder in einer Zeile mit dem umgebenden Text. Wenn auf `0` gesetzt, wird angenommen, dass die LATEX-Umgebung Leerraum nach den `\begin{`*LatexName*`}` und `\end{`*LatexName*`}` Befehlen ignoriert (inklusive des Zeilenumbruchzeichens).

End beendet die `InsetLayout`-Definition.

Font wird für den Text *und* die Marke benutzt (siehe Kap. 5.3.11). Beachten Sie, dass die Definition dieses Fonts automatisch dem `LabelFont` denselben Wert zuweist, das heißt `Font` muss zuerst definiert werden und `LabelFont` danach, wenn sie unterschiedlich sein sollen.

FixedWidthPreambleEncoding [*0*, 1] Ob eine Zeichenkodierung mit „fester Breite“ für den übersetzten Inhalt von `BabelPreamble` und `LangPreamble` erzwungen wird. Dies wird für spezielle LATEX-Pakete wie `listings` benötigt, die keine variable Zeichenkodierung wie `utf8` erlauben. Diese Einstellung wird ignoriert, wenn LATEX-Varianten wie XeTEX oder LuaTEX verwendet werden, die Unicode voll unterstützen.

ForceLocalFontSwitch [*0*, 1] Wenn babel verwendet wird; ob immer eine lokale Umschaltung der Sprache erfolgen soll (mittels `\foreignlanguage`) und nie eine globale (mittels `\selectlanguage`).

ForceLTR erzwingt die „LATEX-Sprache“ und führt zu einer links-nach-rechts-Ausgabe, zum Beispiel bei TEX-Code oder URL. `ForceLTR` ist eine Behelfslösung.

ForceOwnlines [*0*, 1] erzwingt einen Zeilenumbruch in der LATEX-Ausgabe vor und nach der Einfügung. Dies stellt sicher, dass die Einfügung in eigenen Zeilen ausgegeben wird, um die Ausgabe später besser anderweitig einfacher verändern zu können.

ForcePlain [*0*, 1] zeigt an, ob stattdessen `PlainLayout` benutzt werden soll oder ob der Benutzer den Absatzstil der Einfügung ändern darf.

FreeSpacing [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.6.

HTML* Diese Tags kontrollieren die XHTML-Ausgabe. Siehe Kap. 5.4.

InToc [*0*, 1] Ob der Inhalt der Einfügung für die Zeichenketten des 'Gliederungs'-Fensters verwendet werden soll. Zum Beispiel will man nicht, dass der Inhalt

einer Fußzeile im Namen des Abschnitts im Inhaltsverzeichnis des Gliederungs-Fensters erscheint. Aber man will normalerweise, dass der Inhalt von Zeichenstilen erscheint.

KeepEmpty [*0*, 1] Wie bei Absatz-Layouts, siehe Kap. 5.3.6.

LabelFont ist die für die Marke benutze Schrift (siehe Kap. 5.3.11). Beachten Sie, dass diese Definition niemals vor `Font` erscheinen darf, weil sie sonst unwirksam ist.

LabelString [`string=""`] wird auf dem Knopf und anderswo als Einfügungsmarke angezeigt. Einige Einfügungstypen (TEX-Code und Zweig) ändern diese Marke im Vorübergehen.

LangPreamble Sprachabhängige Präambel; siehe Kap. 5.3.7.

LatexName [`<Name>`] ist der Name der LATEX-Umgebung oder des LATEX-Befehls.

LatexParam [`<Parameter>`] ist ein optionaler Parameter für den zugehörigen `LatexName`, einschließlich möglicher Klammerpaare wie `[]`. Dieser Parameter kann in LYX nicht geändert werden (man verwendet `Argument` für anpassbare Parameters). Dieser wird nach allen anderen LATEX-`Argument`en ausgegeben.

LatexType [`Command, Environment, None`] Wie der Stil in LATEX übersetzt wird.[13]

None bedeutet nichts Spezielles

Command bedeutet `\`*`LatexName`*`{...}`

Environment bedeutet `\begin{`*`LatexName`*`}...\end{`*`LatexName`*`}`.

Zusammenfassend bedeutet das, dass die LATEX-Ausgabe entweder:

```
\LatexName[LatexParam]{...}
```

oder:

```
\begin{LatexName}[LatexParam] ... \end{LatexName}
```

sein wird, je nach LATEX-Typ.

LeftDelim [`string`] Eine Zeichenkette, die zu Beginn des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

LyxType kann `charstyle`, `custom`, `element` oder `end` (zeigt das Ende einer Definition an) sein. Dieser Eintrag wird für flexible Einfügungen benötigt und ist nur dort sinnvoll. Neben anderen Dingen legt er fest, in welchem Menü diese Einfügung erscheinen wird. Wird `LyxType` auf `charstyle` gesetzt, wird `MultiPar`

[13] `LatexType` ist vielleicht etwas missverständlich, da diese Regeln auch für SGML-Klassen gelten. Siehe die SGML-Klassendateienfür spezielle Beispiele.

automatisch auf `0` und `ForcePlain` automatisch auf `1` gesetzt. `MultiPar` kann auf `1` oder `ForcePlain` auf `0` für `charstyle`-Einfügungen gesetzt werden, indem es *nach* dem `LyxType` spezifiziert wird.

`MultiPar` [*0*, `1`] zeigt an, ob in dieser Einfügung mehrfache Absätze erlaubt sind. Dadurch wird `CustomPars` auf denselben Wert gesetzt und `ForcePlain` auf den anderen. Diese können auf andere Werte gesetzt werden, wenn sie *nach* `MultiPar` benutzt werden.

`NeedProtect` [*0*, `1`] zeigt an, ob *zerbrechliche* Befehle in diesem Layout geschützt (`\protect`) werden sollen. Es zeigt *nicht* an, ob der Befehl selber geschützt werden soll.

`NoInsetLayout` [`<Layout>`] Löscht ein vorhandenes `InsetLayout`.

`ObsoletedBy` [`<Layout>`] Name eines `InsetLayout`, das dieses `InsetLayout` ersetzt. Dies wird verwendet um ein `InsetLayout` umzubenennen und dabei die Rückwärtskompatibilität zu erhalten.

`ParbreakIsNewline` [*0*, `1`] Wie bei Absatz-Layouts, siehe Kap. 5.3.6.

`PassThru` [*0*, `1`] Wie bei Absatz-Layouts, siehe Kap. 5.3.6.

`Preamble` Wie bei Absatz-Layouts, siehe Kap. 5.3.6.

`RefPrefix` [`string`] Der Präfix, der verwendet werden soll, wenn auf Marken dieser Einfügung verwiesen wird. Dies erlaubt die Verwendung von Formatierten Querverweisen.

`Requires` [`string`] Wie bei Absatz-Layouts, siehe Kap. 5.3.6.

`ResetArgs` [*0*, `1`] Setzt die LaTeX-Argumente dieses Stils zurück (der via `Argument` definiert wurde). Dies ist nützlich, wenn man einen Stil mit `CopyStyle` kopiert hat, aber nicht dessen (benötigten und optionalen) Argumente übernehmen will.

`ResetsFont` [`0`, *1*] Ob die Einfügung die Schrift der übergeordneten Umgebung verwenden soll oder ihre eigene. Voreinstellung ist *0*: verwendet die Schrift der übergeordneten Umgebung.

`RightDelim` [`string`] Eine Zeichenkette, die am Ende des Inhalts des Stils ausgegeben wird. Ein Zeilenumbruch in der Ausgabe wird mit `<br/>` angegeben.

`Spellcheck` [`0`, *1*] Erlaubt es, den Inhalt der Einfügung auf Rechtschreibung zu überprüfen.

5.3.10. Zähler

Es ist notwendig Zähler (CHAPTER, FIGURE,...) in der Textklasse selber zu definieren. Die Standardzähler sind in der Datei `stdcounters.inc` definiert, so dass Sie nur die Zeile

```
Input stdcounters.inc
```

zu zu Ihrer Layout-Datei hinzufügen müssen, damit sie arbeiten. Aber wenn Sie eigene Zähler definieren wollen, können Sie das tun. Zähler-Deklarationen beginnen mit

```
Counter <Name>
```

wobei `<Name>` der Name Ihres Zählers ist. Die Deklaration endet mit `End`.

Folgende Parameter können auch benutzt werden:

`InitialValue` [`int=1`] Setzt den Startwert für einen Zähler, auf den er zurückgesetzt wird. Normalerweise will man die Voreinstellung „1“ haben.

`LabelString` [`string=""`] definiert, wie der Zähler dargestellt wird. Hierdurch wird `LabelStringAppendix` auf denselben Wert gesetzt. In der Zeichenkette können folgende Konstrukte benutzt werden:

- `\thecounter` wird durch die Erweiterung von `LabelString` (oder `LabelStringAppendix`) des Zählers `counter` ersetzt.
- Zählerwerte können durch LaTeX-ähnliche Makros wie `\`*`numbertype`*`{`*`counter`*`}` ausgedrückt werden, wobei *`numbertype`* Folgendes sein kann: `arabic`: 1, 2, 3,...; `alph` für Kleinbuchstaben: a, b, c,...; `Alph` für Großbuchstaben: A, B, C,...; `roman` für kleine römische Zahlen: i, ii, iii,...; `Roman` für große römische Zahlen: I, II, III.

Wenn `LabelString` nicht definiert ist, wird ein Standardwert wie folgt benutzt: wenn der Zähler einen Hauptzähler `master` (über `Within` definiert) hat, wird der String `\themaster.\arabic{counter}` benutzt, ansonsten `\arabic{counter}`.

`LabelStringAppendix` [`string=""`] ist dasselbe wie `LabelString`, aber für den Anhang.

`PrettyFormat` [`string=""`] Ein Format, das für Formatierte Querverweise auf einen Zähler verwendet wird. Möchte man z. B. Referenzen auf Abschnitte in der Form „Abschnitt 2.4“ haben, sollte der String „##“ enthalten. Diese Zeichen werden später durch die referenzierte Abschnittsnummer ersetzt. Also für Abschnitte lautet der String dann „Abschnitt ##“.

`Within` [`string=""`] Wenn dies auf den Namen eines anderen Zählers gesetzt wird, wird der gegenwärtige Zähler jedes mal zurückgesetzt, wenn der andere erhöht wird. Zum Beispiel wird `subsection` innerhalb `section` nummeriert.

5.3.11. Beschreibung des Zeichensatzes

Eine Zeichensatzbeschreibung sieht folgendermaßen aus:

```
Font oder LabelFont oder DefaultFont
 ...
EndFont
```

und es sind folgende Befehle vorhanden:

Color [none, black, white, red, green, blue, cyan, magenta, yellow, brown, darkgray, gray, lightgray, lime, orange, olive, pink, purple,teal, violet]

Family [Roman, Sans, Typewriter]

Misc [string] Zulässige Argumente sind: emph, noun, strikeout, underbar, uuline, uwave, no_emph, no_noun, no_strikeout, no_bar, no_uuline und no_uwave. Jedes schaltet die entsprechende Eigenschaft an oder aus. Zum Beispiel führt emph zum Stil *Hervorhebung* und no_emph schaltet diesen aus.
Falls Sie Letzteres verwirrt, erinnern Sie sich, dass die Schrifteinstellungen standardmäßig von den umgebenden Stilen übernommen wird. Daher schaltet no_emph die *Hervorhebung* aus, die z. B. in einer Theorem-Umgebung aktiv ist.

Series [Medium, Bold]

Shape [Up, Italic, SmallCaps, Slanted]

Size [tiny, small, normal, large, larger, largest, huge, giant]

5.3.12. Citation format description

The CiteFormat blocks are used to describe how bibliographic information should be displayed, both within LyX itself (in the citation dialog and in tooltips, for example) and in XHTML output. Such a block might look like this:

```
CiteFormat
 article ...
 book ...
End
```

The individual lines define how the bibliographic information associated with an article or book, respectively, is to be displayed, and such a definition can be given for any 'entry type' that might be present in a BibTeX file. LyX defines a default format in the source code that will be used if no specific definition has been given. LyX predefines several formats in the file stdciteformats.inc, which is included in most of LyX's document classes.

The definitions use a simple language that allows BibTEX keys to be replaced with their values. Keys should be enclosed in % signs, e.g.: %author%. So a simple definition might look like this:

```
misc %author%, "%title%".
```

This would print the author, followed by a comma, followed by the title, in quotes, followed by a period.

Of course, sometimes you may want to print a key only if it exists. This can be done by using a conditional construction, such as: {%volume%[[vol. %volume%]]}. This says: If the volume key exists, then print "vol. " followed by the volume key. It is also possible to have an else clause in the conditional, such as:
{%author%[[%author%]][[%editor%, ed.]]}
Here, the author key is printed if it exists; otherwise, the editor key is printed, followed by ", ed." Note that the key is again enclosed in % signs; the entire conditional is enclosed in braces; and the if and else clauses are enclosed in double brackets, "[[" and "]]". There must be no space between any of these.

There is one other piece of syntax available in definitions, which looks like this: {!<i>!}. This defines a piece of formatting information that is to be used when creating "rich text". Obviously, we do not want to output HTML tags when writing plain text, so they should be wrapped in "{!" and "!}".

Two special sorts of definitions are also possible in a CiteFormat block. An example of the first would be:

```
!quotetitle "%title%"
```

This is an abbreviation, or macro, and it can be used by treating it as if it were a key: %!quotetitle%. LyX will treat %!quotetitle% exactly as it would treat its definition. So, let us issue the obvious *warning*. Do not do this:

```
!funfun %funfun%
```

or anything like it. LyX shouldn't go into an infinite loop, but it may go into a long one before it gives up.

The second sort of special definition might look like this:

```
_pptext pp.
```

This defines a translatable piece of text, which allows relevant parts of the bibliography to be translated. It can be included in a definition by treating it as a key: %_pptext%. Several of these are predefined in stdciteformats.inc. Note that these are not macros, in the sense just defined. They will not be expanded.

So here then is an example that use all these features:

```
!authoredit {%author%[[%author%, ]][[{%editor%[[%editor%, %_edtext%, ]]}]]}
```

This defines a macro that prints the author, followed by a comma, if the `author` key is defined, or else prints the name of the editor, followed by the `_edtext` or its translation (it is by default "ed."), if the `editor` key is defined. Note that this is in fact defined in `stdciteformats.inc`, so you can use it in your own definitions, or re-definitions, if you load that file first.

5.4. Tags for XHTML output

As with LaTeX or DocBook, the format of LyX's XHTML output is also controlled by layout information. In general, LyX provides sensible defaults and, as mentioned earlier, it will even construct default CSS style rules from the other layout tags. For example, LyX will attempt to use the information provided in the `Font` declaration for the Chapter style to write CSS that will appropriately format chapter headings.

In many cases, then, you may not have to do anything at all to get acceptable XHTML output for your own environments, custom insets, and so forth. But in some cases you will, and so LyX provides a number of layout tags that can be used to customize the XHTML and CSS that are generated.

Note that there are two tags, `HTMLPreamble` and `AddToHTMLPreamble` that may appear outside style and inset declarations. See Kap. 5.3.4 for details on these.

5.4.1. Paragraph styles

The sort of XHTML LyX outputs for a paragraph depends upon whether we are dealing with a normal paragraph, a command, or an environment, where this is itself determined by the contents of the corresponding `LaTeXType` tag.

For a command or normal paragraph, the output XHTML has the following form:

```
<tag attr="value">
<labeltag attr="value">Label</labeltag>
Contents of the paragraph.
</tag>
```

The label tags are of course omitted if the paragraph does not have a label.

For an environment that is not some sort of list, the XHTML takes this form:

```
<tag attr="value">
<itemtag attr="value"><labeltag attr="value">Environment Label</labeltag>First pa
<itemtag>Second paragraph.</itemtag>
</tag>
```

Note that the label is output only for the first paragraph, as it should be for a theorem, for example.

For a list, we have one of these forms:

```
<tag attr="value">
<itemtag attr="value"><labeltag attr="value">List Label</labeltag>First item.
<itemtag attr="value"><labeltag attr="value">List Label</labeltag>Second item
</tag>
<tag attr="value">
<labeltag attr="value">List Label</labeltag><itemtag attr="value">First item.
<labeltag attr="value">List Label</labeltag><itemtag attr="value">Second item
</tag>
```

Note the different orders of `labeltag` and `itemtag`. Which order we get depends upon the setting of `HTMLLabelFirst`: If `HTMLLabelFirst` is false (the default), you get the first of these, with the label within the item; if true, you get the second, with the label outside the item.

The specific tags and attributes output for each paragraph type can be controlled by means of the layout tags we are about to describe. As mentioned earlier, however, LyX uses sensible defaults for many of these, so you often may not need to do very much to get good XHTML output. Think of the available tags as there so you can tweak things to your liking.

`HTMLAttr` [`string`] Specifies attribute information to be output with the main tag. For example, "`class='mydiv'`". By default, LyX will output "`class='layoutname'`", where `layoutname` is the LyX name of the layout, made lowercase, for example: chapter. This should *not* contain any style information. Use `HTMLStyle` for that purpose.

`HTMLForceCSS` [*`0`*, `1`] Whether to output the default CSS information LyX generates for this layout, even if additional information is explicitly provided via `HTMLStyle`. Setting this to `1` allows you to alter or augment the generated CSS, rather than to override it completely. Default is `0`.

`HTMLItem` [`string`] The tag to be used for individual paragraphs of environments, replacing `itemtag` in the examples above. Defaults to `div`.

`HTMLItemAttr` [`string`] Attributes for the item tag. Defaults to "`class='layoutname_item'`". This should *not* contain any style information. Use `HTMLStyle` for that purpose.

`HTMLLabel` [`string`] The tag to be used for paragraph and item labels, replacing `labeltag` in the examples above. Defaults to `span`, unless `LabelType` is either `Top_Environment` or `Centered_Top_Environment`, in which case it defaults to `div`.

`HTMLLabelAttr` [`string`] Attributes for the label tag. Defaults to "`class='layoutname_label'`". This should *not* contain any style information. Use `HTMLStyle` for that purpose.

`HTMLLabelFirst` [*`0`*, `1`] Meaningful only for list-like environments, this tag controls whether the label tag is output before or inside the item tag. This is used, for example, in the description environment, where we want '`<dt>...</dt><dd>...</dd>`. Default is `0`: The label tag is output inside the item tag.

HTMLPreamble Information to be output in the `<head>` section when this style is used. This might, for example, be used to include a `<script>` block defining an `onclick` handler.

HTMLStyle CSS style information to be included when this style is used. Note that this will automatically be wrapped in a layout-generated `<style>` block, so only the CSS itself need be included. Must end with `EndHTMLStyle`.

HTMLTag [`string`] The tag to be used for the main label, replacing `tag` in the examples above. Defaults to `div`.

HTMLTitle [*0*, `1`] Marks this style as the one to be used to generate the `<title>` tag for the XHTML file. By default, it is false. The `stdtitle.inc` file sets it to true for the `title` environment.

5.4.2. InsetLayout XHTML

The XHTML output of insets can also be controlled by information in layout files.[14] Here, too, LyX tries to provide sensible defaults, and it constructs default CSS style rules. But everything can be customized.

The XHTML LyX outputs for an inset has the following form:

```
<tag attr="value">
<labeltag>Label</labeltag>
<innertag attr="value">Contents of the inset.</innertag>
</tag>
```

If the inset permits multiple paragraphs—that is, if `MultiPar` is true—then the contents of the inset will itself be output as paragraphs formatted according to the styles used for those paragraphs (standard, quote, and the like). The label tag is of course omitted if the paragraph does not have a label and, at present, is always `span`. The inner tag is optional and, by default, does not appear.

The specific tags and attributes output for each inset can be controlled by means of the following layout tags.

HTMLAttr [`string`] Specifies attribute information to be output with the main tag. For example, "`class='myinset' onclick='...'`". By default, LyX will output "`class='insetname'`", where `insetname` is the LyX name of the inset, made lowercase and with non-alphanumeric characters converted to underscores, for example: footnote.

HTMLForceCSS [*0*, `1`] Whether to output the default CSS information LyX generates for this layout, even if additional information is explicitly provided via `HTMLStyle`. Setting this to `1` allows you to alter or augment the generated CSS, rather than to override it completely. Default is 0.

[14] At present, this is true only for "text" insets (insets you can type into) and is not true for "command" insets (insets that are associated with dialog boxes).

HTMLInnerAttr [string] Attributes for the inner tag. Defaults to "class='insetname_inner'".

HTMLInnerTag [string] The inner tag, replacing innertag in the examples above. By default, there is none.

HTMLIsBlock [0, *1*] Whether this inset represents a standalone block of text (such as a footnote) or instead represents material that is included in the surrounding text (such as a branch). Defaults to 1.

HTMLLabel [string] A label for this inset, possibly including a reference to a counter. For example, for footnote, it might be: \arabic{footnote}. This is optional, and there is no default.

HTMLPreamble Information to be output in the <head> section when this style is used. This might, for example, be used to include a <script> block defining an onclick handler.

HTMLStyle CSS style information to be included when this style is used. Note that this will automatically be wrapped in a layout-generated <style> block, so only the CSS itself need be included.

HTMLTag [string] The tag to be used for the main label, replacing tag in the examples above. The default depends upon the setting of MultiPar: If MultiPar is true, the default is div; if it is false, the default is span.

5.4.3. Float XHTML

The XHTML output for floats too can be controlled by layout information. The output has the following form:

```
<tag attr="value">
Contents of the float.
</tag>
```

The caption, if there is one, is a separate inset and will be output as such. Its appearance can be controlled via the InsetLayout for caption insets.

HTMLAttr [string] Specifies attribute information to be output with the main tag. For example, "class='myfloat' onclick='...'". By default, LyX will output "class='float float-floattype'", where floattype is LyX's name for this type of float, as determined by the float declaration (see Kap. 5.3.8), though made lowercase and with non-alphanumeric characters converted to underscores, for example: float-table.

HTMLStyle CSS style information to be included when this float is used. Note that this will automatically be wrapped in a layout-generated <style> block, so only the CSS itself need be included.

`HTMLTag` [`string`] The tag to be used for this float, replacing "`tag`" in the example above. The default is `div` and will rarely need changing.

5.4.4. Bibliography formatting

The bibliography can be formatted using `CiteFormat` blocks. See Kap. 5.3.12 for the details.

5.4.5. LyX-generated CSS

We have several times mentioned that LyX will generate default CSS style rules for both insets and paragraph styles, based upon the other layout information that is provided. In this section, we shall say a word about which layout information LyX uses and how.

At present, LyX auto-generates CSS only for font information, making use of the `Family`, `Series`, `Shape`, and `Size` specified in the `Font` declaration (see Kap. 5.3.11). The translation is mostly straightforward and obvious. For example, "`Family Sans`" becomes "`font-family: sans-serif;`". The correspondence of LyX sizes and CSS sizes is a little less obvious but nonetheless intuitive. See the `getSizeCSS()` function in `src/FontInfo.cpp` for the details.

6. Externes Material einfügen

WARNUNG: This portion of the documentation has not been updated for some time. We certainly hope that it is still accurate, but there are no guarantees.

The use of material from sources external to LyX is covered in detail in the *Embedded Objects* manual. This part of the manual covers what needs to happen behind the scenes for new sorts of material to be included.

6.1. Wie funktioniert das?

Die Einfügung Externes Material basiert auf dem Konzept der Vorlage. Eine solche Vorlage ist eine Spezifikation, wie LyX mit einer bestimmten Sorte von Material umgehen soll. Derzeit gehören zu LyX derartige Vorlagen für XFig-Abbildungen, Dia-Diagramme, diverse Abbildungen im Rasterformat, Gnuplot und noch ein paar mehr. Die vollständige Liste sehen Sie in Einfügen ▷ Datei ▷ Externes Material.

Darüberhinaus ist es möglich, durch eigene Vorlagen beliebige andere Formate einzubinden. Wir werden weiter unten beschreiben, was genau Sie dazu machen müssen und hoffen, dass Sie derartig erstellte Vorlagen an das LyX-Team schicken, damit sie in kommenden LyX-Versionen integriert werden können.

Ein weiteres Merkmal der Idee der externen Einfügung ist die Unterscheidung zwischen der ursprünglichen Datei, die als Grundlage für das eingefügte Material dient, und der erzeugten Datei, die dann letztendlich in Ihr Dokument eingebunden wird. Wir wollen dies am Beispiel einer XFig-Abbildung erläutern.

Das Programm XFig bearbeitet eine speziell formatierte Datei mit der Endung `.fig`. In XFig können Sie Ihre Abbildung editieren und ändern, und zum Schluss speichern Sie diese `.fig`-Datei. Wenn Sie nun eine derartige Abbildung in LyX einbinden wollen, müssen Sie zunächst `transfig` starten, um eine PostScript-Datei zu erzeugen, die von LaTeX eingebunden werden kann. In diesem Fall ist also die `.fig`-Datei die oben erwähnte Originaldatei, und die `.ps`-Datei die tatsächlich eingebundene Datei.

Diese Unterscheidung ist wichtig, denn Sie erlaubt das einfache Ändern und Aktualisieren des Materials, während Sie an Ihrem Text schreiben. Außerdem ist erst so die Flexibilität gegeben, die benötigt wird, um unterschiedliche Exportformate für die LyX-Datei zu ermöglichen.

So ist es im Falle einer Ausgabe als reiner (ASCII) Text sicher nicht sinnvoll, eine PostScript-Datei im Rohformat einzubinden. In diesem Fall wird dann entweder nur eine Referenz auf die Bilddatei angegeben, oder aber es wird ein Konverter gestartet, der eine ASCII-Darstellung erzeugt, die in etwa so aussieht wie die ursprüngliche Gra-

fik. Genau dies ist mit der Einfügung Externes Material möglich, denn sie kennt all die notwendigen Befehle für derartige Konvertierungen (sofern sie von LyX unterstützt werden).

Darüberhinaus erlaubt die Einfügung Externes Material aber auch die einfache Integration mit externen Betrachtern und Editoren. So sind Sie bei einer XFig-Abbildung in der Lage, mit einem einigen Klick XFig zu starten, um die Abbildung zu bearbeiten oder die erstellte PostScript-Datei mit `ghostview` zu betrachten. Kein langes Herumsuchen mit Dateimanagern nach den Original- und Grafikdateien mehr, und Sie müssen sich nicht mehr an die unterschiedlichen Parameter erinnern, die vielleicht für diese Abbildung notwendig sind, um sie in der richtigen Größe zu erstellen. Sie haben ohne viel Aufwand Zugriff auf eine Vielzahl von Applikationen und können so Ihre Produktivität ungemein steigern.

LyX besitzt also die Information über eine Vielzahl von externen Programmen, um diese von Ihnen unbemerkt nutzen zu können und Ihnen so den größtmöglichen Komfort zu bieten. Und genau diese Information ist in den oben erwähnten Vorlagen gespeichert. Jede dieser Vorlagen enthält eine Liste von Befehlen und Optionen, um externe Programme zu starten, Formate zu konvertieren usw.

Ein fortgeschrittener Anwender kann mit derartigen selbst erstellten Vorlagen die Möglichkeiten von LyX stark erweitern, ohne dazu den eigentlichen Quellcode verändern zu müssen. Zwar ist dazu einiges an Arbeit notwendig, um all diese Befehle festzulegen, aber zum Glück hat das LyX-Team das in einigen Fällen ja schon getan.

Eine kleine Einschränkung gibt es aber doch: Da es wie erwähnt eine Vielzahl möglicher Exportformate für das eingefügte Material gibt, wäre es zu vermuten, dass man von LyX aus auch alle diese Formate als Vorschau ansehen kann. Das LyX-Team hat sich entschlossen, das nicht zu tun, um die Benutzerschnittstelle so einfach wie möglich zu halten. Anstatt im Dialog für jedes mögliche Exportformat einen eigenen Knopf für die Vorschau zu haben, wurde das Konzept des primären Formats eingeführt, und es gibt nur einen Schalter in LyX anzeigen, der die Datei in genau diesem primären Format anzeigt.

Dieses Format wird durch die verwendete Dokumentenklasse festgelegt. So ist es für die meisten Klassen LaTeX, für die DocBook-Klassen ist es aber DocBook. Denken Sie also daran, dass Ihnen die Vorschau lediglich das Aussehen in diesem Hauptformat anzeigt; wenn Sie sehen wollen, wie das Ergebnis in anderen Formaten aussieht, müssen Sie wie gewohnt die Konvertierung manuell durchführen.

6.2. The external template configuration file

It is relatively easy to add custom external template definitions to LyX. However, be aware that doing this in a careless manner most probably *will* introduce an easily exploitable security hole. So before you do this, please read the discussion about security in Kap. 6.4.

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

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

A typical template looks like this:

```
Template XFig
GuiName "XFig: $$AbsOrRelPathParent$$Basename"
HelpText
An XFig figure.
HelpTextEnd
InputFormat fig
FileFilter "*.fig"
AutomaticProduction true
Transform Rotate
Transform Resize
Format LaTeX
TransformCommand Rotate RotationLatexCommand
TransformCommand Resize ResizeLatexCommand
Product "$$RotateFront$$ResizeFront
         \\input{$$AbsOrRelPathMaster$$Basename.pstex_t}
         $$ResizeBack$$RotateBack"
UpdateFormat pstex
UpdateResult "$$AbsPath$$Basename.pstex_t"
Requirement "graphicx"
ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pstex_t"
ReferencedFile latex "$$AbsPath$$Basename.eps"
ReferencedFile dvi "$$AbsPath$$Basename.eps"
FormatEnd
Format PDFLaTeX
TransformCommand Rotate RotationLatexCommand
TransformCommand Resize ResizeLatexCommand
Product "$$RotateFront$$ResizeFront
         \\input{$$AbsOrRelPathMaster$$Basename.pdftex_t}
         $$ResizeBack$$RotateBack"
UpdateFormat pdftex
UpdateResult "$$AbsPath$$Basename.pdftex_t"
Requirement "graphicx"
ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pdftex_t"
ReferencedFile latex "$$AbsPath$$Basename.pdf"
FormatEnd
Format Ascii
Product "$$Contents(\"$$AbsPath$$Basename.asc\")"
UpdateFormat asciixfig
UpdateResult "$$AbsPath$$Basename.asc"
FormatEnd
```

```
Format DocBook
Product "<graphic fileref=\"$$AbsOrRelPathMaster$$Basename.eps\">
         </graphic>"
UpdateFormat eps
UpdateResult "$$AbsPath$$Basename.eps"
ReferencedFile docbook "$$AbsPath$$Basename.eps"
ReferencedFile docbook-xml "$$AbsPath$$Basename.eps"
FormatEnd
Product "[XFig: $$FName]"
FormatEnd
TemplateEnd
```

As you can see, the template is enclosed in `Template` ... `TemplateEnd`. It contains a header specifying some general settings and, for each supported primary document file format, a section `Format` ... `FormatEnd`.

6.2.1. The template header

`AutomaticProduction true|false` Whether the file represented by the template must be generated by LyX. This command must occur exactly once.

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

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

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

`InputFormat <format>` The file format of the original file. This must be the name of a format that is known to LyX (see Kap. 3.1). Use "`*`" if the template can handle original files of more than one format. LyX will attempt to interrogate the file itself in order to deduce its format in this case. This command must occur exactly once.

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

`Transform Rotate|Resize|Clip|Extra` This command specifies which transformations are supported by this template. It may occur zero or more times. This command enables the corresponding tabs in the external dialog. Each `Transform` command must have either a corresponding `TransformCommand` or

a `TransformOption` command in the `Format` section. Otherwise the transformation will not be supported by that format.

6.2.2. The Format section

`Format LaTeX|PDFLaTeX|PlainText|DocBook|XHTML` The primary document file format that this format definition is for. Not every template has a sensible representation in all document file formats. Please define nevertheless a `Format` section for all templates. Use a dummy text when no representation is available. Then you can at least see a reference to the external material in the exported document.

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

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

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

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

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

`TransformCommand Rotate RotationLatexCommand` This command specifies that the built in LaTeX command should be used for rotation. This command may occur once or not at all.

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

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

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

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

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

UpdateFormat <format> The file format of the converted file. This must be the name of a format that is known to LyX (see the TOOLS ▷ PREFERENCES ▷ FILE HANDLING ▷ FILE FORMAT dialog). This command must occur exactly once. If the resulting file format is PDF, you need to specify the format `pdf6`. This is the PDF format used for including graphics. The other defined PDF formats are for document export.

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

6.2.3. Preamble definitions

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

6.3. Der Ersetzungsmechanismus

Wenn über die externe Einfügung ein externes Programm gestartet wird, geschieht dies anhand eines Befehls, der in der Vorlage festgelegt wurde. Ein solcher Befehl kann diverse Makros enthalten, die vor dem eigentlichen Aufruf ausgewertet werden. Die Ausführung erfolgt dabei immer in demjenigen Verzeichnis, das auch das LyX-Dokument enthält.

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

Hier finden Sie eine Liste dieser Makros:

$$AbsOrRelPathMaster The file path, absolute or relative to the master LyX document.

$$AbsOrRelPathParent The file path, absolute or relative to the LyX document.

$$AbsPath The absolute file path.

$$Basename The filename without path and without the extension.

$$Contents("filename.ext") This macro will expand to the contents of the file with the name `filename.ext`.

$$Extension The file extension (including the dot).

$$pngOrjpg This will be the string "jpg" if the file is in JPEG format, otherwise it will be the string "png". This is useful to avoid uneeded conversions for output formats that support both PNG and JPEG fomats. The predefined RasterImage template uses this macro for the pdfTEX output format.

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

$$FPath The path part of `$$FName` (absolute name or relative to the LyX document).

$$RelPathMaster The file path, relative to the master LyX document.

$$RelPathParent The file path, relative to the LyX document.

$$Sysdir This macro will expand to the absolute path of the system directory. This is typically used to point to the various helper scripts that are bundled with LyX.

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

All path macros contain a trailing directory separator, so you can construct e. g. the absolute filename with `$$AbsPath$$Basename$$Extension`.

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

$$ResizeFront The front part of the resize command.

$$ResizeBack The back part of the resize command.

$$RotateFront The front part of the rotation command.

$$RotateBack The back part of the rotation command.

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

$$Clip The clip option.

$$Extra The extra option.

$$Resize The resize option.

$$Rotate The rotation option.

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

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

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

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

- Use `$$AbsPath` if an absolute path is required.
- Use `$$AbsOrRelPathMaster` if the substituted string is some kind of LaTeX input.
- Else use `$$AbsOrRelPathParent` in order to preserve the user's choice.

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

6.4. Sicherheitshinweise

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

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

This also implies that you are restricted in what command strings you can use in the external material templates. In particular, pipes and redirection are not readily available. This has to be so if LyX should remain safe. If you want to use some of the shell features, you should write a safe script to do this in a controlled manner, and then invoke the script from the command string.

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

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

A. Liste der Funktionen für die Verwendung in Layout-Dateien

accents	booktabs	feyn	listings	natbib	rotfloat	tfrupee	wasysym
amsbsy	calc	fixltx2e	longtable	nomencl	rsphrase	tipa	wrapfig
amscd	CJK	float	lyxskak	pdfcolmk	setspace	tipx	xargs
amsmath	color	framed	makeidx	pdfpages	shapepar	tone	xcolor
amssymb	covington	graphicx	marvosym	pifont	slashed	txfonts	xy
amstext	csquotes	hhline	mathdesign	pmboxdraw	soul	ulem	yhmath
amsthm	dvipost	hyperref	mathdots	polyglossia	splitidx	undertilde	
array	endnotes	ifsym	mathrsfs	prettyref	subfig	units	
ascii	enumitem	ifthen	mhchem	pxfonts	subscript	url	
bbding	esint	jurabib	multicol	refstyle	textcomp	varioref	
bm	fancybox	latexsym	multirow	rotating	textgreek	verbatim	

B. Namen von verfügbaren Farben für die Verwendung in Layout-Dateien

Die hier aufgelisteten Farben sind die Standardfarben und die, die man in den LyX-Voreinstellungen festlegen kann.

none Keine spezielle Farbe – entfernt Farbe oder setzt Farbe auf Voreinstellung

black

white

red

green

blue

cyan

magenta

yellow

added_space Added space marker color

addedtext Added text color

appendix Appendix marker color

background Hintergrundfarbe

bottomarea Bottom area color

branchlabel Label color for branches

buttonbg Color used for bottom background

buttonhoverbg Color used for button background under focus

buttonframe Color for inset button frames

changebar Changebar color

changedtextauthor1 Geänderter Text des 1. Autors

changedtextauthor2 Geänderter Text des 2. Autors

changedtextauthor3 Geänderter Text des 3. Autors

changedtextauthor4 Geänderter Text des 4. Autors

changedtextauthor5 Geänderter Text des 5. Autors

collapsable_inset_frame Collapsable insets framecolor

collapsable_inset_text Collapsable insets text color

command Text color for command insets

commandbg Background color for command insets

commandframe Frame color for command insets

comment color for comments

commentbg Background color of comments

cursor Cursorfarbe

deletedtext Deleted text color

deletedtextmodifier Deleted text modifying color

depthbar Color for the depth bars in the margin

eolmarker End of line marker color

error Color of the LaTeX error box

footlabel Label color for footnotes

graphicsbg Graphics inset background color

greyedout Label color for greyedout insets

greyedoutbg Background color of greyedout inset

greyedouttext Color for greyedout inset text

indexlabel Label color for index insets

ignore The color is ignored

inherit The color is inherited

inlinecompletion Inline completion color

insetbg Inset marker background color

insetframe Inset marker frame color

language Color for marking foreign language words

latex Text color in LaTeX mode

listingsbg Background color of listings inset

marginlabel Label color for margin notes

math Math inset text color

mathbg Math inset background color

mathcorners Math inset frame color not under focus

mathframe Math inset frame color under focus

mathline Math line color

mathmacrobg Macro math inset background color

mathmacroblend Macro math blended color

mathmacroframe Macro math frame color

mathmacrohoverbg Macro math inset background color hovered

mathmacrolabel Macro math label color

mathmacronewarg Macro template color for new parameters

mathmacrooldarg Macro template color for old parameters

newpage New page color

nonunique_inlinecompletion Inline completion color for the non-unique part

notebg Background color of notes

notelabel Label color for notes

pagebreak Page break/line break color

paragraphmarker Color used for the pilcrow sign to mark the end of a paragraph

phantomtext Text color for phantom insets

preview The color used for previews

previewframe Preview frame color

regexpframe Color for regexp frame

selection Hintergrundfarbe des ausgewählten Texts

selectiontext Vordergrundfarbe des ausgewählten Texts

shadedbg Background color of shaded box

special Special chars text color

tabularline Table line color

tabularonoffline Table line color

urllabel Label color for URL insets

urltext Color for URL inset text