LyXの高度設定
熟練ユーザのための各機能

LyX開発チーム*

第2.4.x版

2024年5月14日

*もしコメントや誤りの修正をお持ちでしたら，LyX 文書化メーリングリスト lyx-docs@lists.lyx.org 宛お送りください．件名ヘッダに「[Customization]」という文字を入れ，このファイルの現在のメンテナ Richard Kimberly Heck <rikiheck@lyx.org>を cc にして送ってください．

目次

1. はじめに

この取扱説明書は，LyX に備わっている高度設定機能を取り扱います．ここでは，ショートカットや画面プレビューオプション，プリンタオプション，LyX サーバ経由での LyX へのコマンド送信，国際化，新しい LaTeX クラスや LyX レイアウトの導入などの題材について論じます．おそらくは変更可能なことすべてについて触れることは無理でしょうが—私たちの開発者たちは私たちが文書化できる速さよりも速く新しい機能を付け加えてしまうので—，もっとも一般的な高度設定については説明を行い，わかりにくいものについては正しい方向を指し示すことができるようにしていくつもりです．

2. LyX 設定ファイル

本章の目的は，LyX 設定ファイル群を理解するための一助となることです．本章を読み進める前に，ヘルプ▷LyX についてを使って，LyX ライブラリとユーザディレクトリがどこにあるかを確認しておいてください．ライブラリディレクトリは，LyX がシステム全体の設定ファイルを置いておくところです．一方，ユーザディレクトリは，自身がそれを修正した版を置いておくところです．私たちは，本書の以下の部分で，前者を LyXDir と呼び，後者を UserDir と呼ぶことにします．

2.1. LyXDir にはなにがあるの？

LyXDir とそのサブディレクトリには，多くのファイルがあり，LyX の挙動を高度設定するのに使用されます．これらのファイルの多くは，LyX 内のツール▷設定ダイアログから変更することができます．LyX 中で行いたいと思うような高度設定は，ほとんどこのダイアログから行うことができるようになっています．しかしながら，LyX の他の多くの内部動作は，LyXDir のファイルを修正することで高度設定されます．これらのファイルは様々なカテゴリに分類しうるので，以下の各小節で説明します．

2.1.1. 自動的に生成されるファイル

UserDir にある各ファイルは，LyX が自動設定を行ったときに生成されます．これらのファイルは，環境構成中に自動的に検出された様々な既定値が置かれています．これらは，随時上書きされてしまうので，一般的には修正しないことが望まれます．

`lyxrc.defaults` このファイルには，様々な既定コマンドが置かれています．

`packages.lst` このファイルには，LyX が認識したパッケージの一覧が収められています．現在のところ，これは LyX プログラム自体には使用されていませんが，抽出された情報その他は，ヘルプ▷LaTeX の設定で見ることができます．

`textclass.lst` ユーザの Plain L ディレクトリで検出されたテキストクラスと，関連した LaTeX 文書クラスおよびその説明の一覧です．

`lyxmodules.lst` ユーザの layout/ディレクトリで検出されたレイアウトモジュールの一覧です．

`*files.lst` ご使用のシステムで検出された様々な種類の LaTeX 関連ファイルの一覧です．

doc/LaTeXConfig.lyx このファイルは，自動設定中に LaTeXConfig.lyx.in から自動的に生成されます．ご使用中の LaTeX の設定に関する情報が納められています．

2.1.2. ディレクトリ

LyXDir に含まれる以下の各ディレクトリは，UserDir にも重複して存在することがあります．特定のファイルが両方の場所にある場合には，UserDir の方にあるものが使用されます．

bind/ このディレクトリには，LyX で使用されるキー割当を定義している，拡張子が .bind のファイルが置かれています．サブディレクトリ bind/xx（“xx” は ISO 言語コード）に割当ファイルの各国語版がある場合には，そちらが用いられます．

citeengines/ このディレクトリには，幅広い文献引用力（natbib, biblatex など）を定義する，拡張子が .citeengine のファイルが置かれています．詳細については，第 5.2.6 節をご覧ください．

clipart/ このディレクトリには，文書に取り込むことのできる画像ファイルが納められています．

doc/ このディレクトリには，LyX の取扱説明書ファイル（今お読みのものも含めて）が納められています．上述のように，LaTeXConfig.lyx ファイルは特に注目に値します．各国語版のヘルプ文書は，doc/xx（「xx」は ISO 言語コード）サブディレクトリにあります．詳しくは，4 をご覧ください．

examples/ このディレクトリには，何らかの機能の使い方を説明する例示ファイルが納められています．ファイルブラウザ中で用例ボタンを押すと，このディレクトリが表示されます．

images/ このディレクトリには，文書ダイアログで使用される画像ファイルが納められています．さらに，ツールバーの各アイコンや，LyX を起動したときに現れるバナーも納められています．

kbd/ このディレクトリには，キーボードのキー割当ファイルが納められています．詳細については，4.2 をご覧ください．

layouts/ このディレクトリには，5 に述べられているテキストクラスおよびモジュールのファイルが納められています．

lyx2lyx このディレクトリには，LyX の各バージョン間の変換に使用される lyx2lyx Python スクリプトが納められています．複数のファイルの変換をバッチ処理したい場合には，これらをコマンドラインから実行することもできます．

scripts/ このディレクトリには，外用ひな型機能の有用性を示すためのファイルがいくつか納められています．LyX 自身が使用するスクリプトもいくつか収められています．

templates/ このディレクトリには，5.2.4 で述べられている標準の LyX ひな型ファイルが納められています．

ui/ このディレクトリには，LyX の操作画面を定義する拡張子.ui のファイルが納められています．つまり，これらのファイルは，どのメニュー項目がどのメニューに現れるかを定義し，どの項目がツールバーに現れるかを定義しています．

xtemplates/ このディレクトリには，LyX 文書への外部素材の挿入のひな型を定義する，拡張子が.xtemplate のファイルが置かれています．第 6 節参照．

2.1.3. 変更を加えない方がよいファイル

これらのファイルは LyX が内部的に使用するもので，あなたが開発者でない限りは，凡そこれらに変更を加える必要はありません．

CREDITS このファイルは，LyX 開発陣の名簿です．この内容は，メニュー項目ヘルプ▷LyX についてで表示されます．

chkconfig.ltx これは，自動設定プロセスによって使用される LaTeX スクリプトです．直接実行しないでください．

configure.py これは，LyX の環境構成によって使用されるスクリプトです．これは，このスクリプトを実行したディレクトリに設定ファイルを生成します．

2.1.4. ひとこと必要なファイル群

encodings このファイルには，各文字エンコーディングがどのように Unicode にマップされるかを示した表が載っています．

languages このファイルには，現在 LyX がサポートしている言語の全一覧が載っています．

latexfonts サポートされているフォントに関する情報が掲載されています．

layouttranslations このファイルは，国際化された段落様式の翻訳が収録されています (5.3.8 参照)．

unicodesymbols このファイルは，unicode エンコーディングされたグリフに関する情報と，LyX が LaTeX を介してそれらをどのようにサポートしているかについての情報を含んでいます．

2.2. ユーザのローカル設定ディレクトリ

LyX を非特権ユーザとして利用している場合でも，自分自身で使うために，LyX の設定を変更したいと思うかもしれません．UserDir ディレクトリには，すべての個人設定ファイルが収められています．これは，ヘルプ▷LyX についてで「ユーザーディレクトリ」として言及されているディレクトリです．このディレクトリは，LyXDir のミラーとして使用されており，これは UserDir 内のすべてのファイルが，LyXDir すし，自分自身で使うために個人のローカルディレクトリに置くこともできます．

わかりやすくするために，いくつか例を挙げましょう．

- ツール▷設定ダイアログで設定されるユーザ設定は，UserDir 中の preferences ファイルに保存されます．
- ツール▷環境構成を使用して環境構成を行うと，LyX は configure.py スクリプトを実行し，その結果のファイルは，ご自身のローカル設定ディレクトリに書き込まれます．これはすなわち，UserDir/layouts にご自身で追加したテキストファイルは，文書▷設定ダイアログのクラス一覧に表示されるようになることを意味します．
- たとえば，LyX の FTP サイトから最新の取扱説明書をとってきたものの，使用中のシステム上で管理者権限がないために，それをインストールすることができなかったとしても，それらのファイルを UserDir/doc/ディレクトリにコピーすれば，ヘルプメニュー項目はこれらを開くようになります！

2.3. LyX を複数の設定を使って実行するには

ローカル設定ディレクトリにおいて設定の自由度があるだけでは，2 つ以上の設定を自由に使いこなしたい場合には充分ではないかもしれません．たとえば，使用する度に異なるキー割当を使用したり，異なるプリンタ設定を使用したいことがあるかもしれません．これは，複数の設定ディレクトリを作ることで実現することができます．そして，実行時にどのディレクトリを使用するか指定するのです．

LyX をコマンドラインスイッチ-userdirP <**ディレクトリ名**>と共に起動すると，設定を既定のディレクトリではなく，指定したディレクトリから読み込むように，指示することになります（LyX を-userdir スイッチなしで実行すれば，既定ディレクトリを指定することになります）．指定したディレクトリが存在しない場合には，LyX は，初めて LyX を実行したときに既定ディレクトリを訊いてくるのと同様に，そのディレクトリを作るかどうか訊いてきます．この追加したユーザディレクトリでは，既定ディレクトリで行うのと全く同じように設定オプションを修正することができます．これらのディレクトリは完全に独立しています（が，読み進めてください）．また，環境変数 LYX_USERDIR_20x を特定の値に設定しても，全く同じ効果があります．

複数の設定を持つことはまた，維持の手間も増えるということです．もし新しいレイアウトを NewUserDir/layouts に加えて，これをすべての設定で利用できるようにしたいならば，これをすべての設定ディレクトリで個々に付け加えなくてはなりません．これを避けるには，次のようなトリックを使用してください．LyX が新しい設定

ディレクトリを生成すると，そのサブディレクトリ（上記参照）はほとんど空です．新しい設定が既存のものをミラーするようにするには，空のサブディレクトリを，既存の設定の対応するサブディレクトリへのシンボリックリンクに置き換えてください．ただし doc/サブディレクトリには，設定スクリプト（ツール▷環境構成で使用可能）が書き出した，設定毎に異なるファイルが含まれていますので，注意を払ってください．

3. 設定ダイアログ

設定ダイアログのオプションのすべては，**ユーザーの手引き**の付録**設定ダイアログ**に述べられています．オプションのうちいくつかについて，ここでさらに詳細に説明します．

3.1. ファイル形式

ファイル形式が定義されていない場合，はじめの一歩は，使いたいと思うファイル形式を定義することです．それには，ツール▷設定ダイアログを開いてください．ファイル処理▷ファイル形式の中で新規...ボタンを押して，登録する新しい形式を定義してください．形式フィールドは，GUI 中で形式を認識するために用いられる名称です．短縮名は，形式を内部的に識別するために用いられます．さらにファイル拡張子も入力する必要があります．これらはすべて必須事項です．オプションのショートカットフィールドは，メニュー中でショートカットを提供するのに使用されます（たとえば，Ctrl+D を押すと表示▷DVI となります）．

形式には，閲覧プログラムと編集プログラムを関連づけることができます．たとえば，PostScript ファイルを閲覧するのに Ghostview を使用したいとしましょう．このプログラムを起動するのに必要なコマンドを対応するフィールドに入力します．ここで，コマンドを定義するのに，次節に掲げる 4 つの変数を用いることができます．この閲覧プログラムは，LyX 中で画像を閲覧したり表示メニューを使用したときに起動されます．一方，編集プログラムは，たとえば，画像を右クリックして現れるコンテクストメニューで外部で編集を選択したときに起動します．

ファイル形式の MIME 型は必須ではありませんが，指定するときには，すべての形式の中で一意的なものでなくてはなりません．これは，この形式のファイルをファイル内容から検出するのに用いられます．重要なファイル形式のうちには，IANA に公式に登録された MIME 型がないものがあります．そこで，LyX は，freedesktop.org で指定されている MIME 型拡張表を使用しています．

文書形式オプションは，LyX に，この形式が文書として書き出すのに適していることを指示するものです．このオプションが有効となっていて，適切な変換経路が存在する場合には（第 3.3 節を参照），この形式がファイル▷書き出し表示 footlabel Label color for footnot す．png のような純粋な画像形式は，このオプションを有効にしてはいけません．pdf のようにベクター画像であると同時に文書でもあるような形式は，これを有効にします．

ベクター画像形式オプションは，LyX にこの形式がベクター画像を含みうることを教示するものです．この情報は，pdflatex を書き出す際に，内包されている画像をどの形式に変換するかを決定するのに使用されます．pdflatex は，pdf・png・Des 以外の画像形式を取り扱うことができないので，内包されている画像は，これらの形式

に変換される必要があるかもしれない為です．内包されている画像が既に pdf・png・jpg のいずれかになっていない場合には，ベクター画像形式オプションが有効になっている場合には pdf に変換され，そうでない場合には png に変換されます．

3.2. 複写子

形式の変換はすべて，LyX の一時ディレクトリで行われるため，変換用にファイルを一時ディレクトリにコピーする前段階で，ファイルに変更を加える必要のあることがあります[1]．これは複写子によって取り扱われ，複写子は，ファイルを一時ディレクトリに（あるいは一時ディレクトリから）コピーすると同時に，その過程でファイルに変更を加えます．

複写子の定義においては，以下の 8 つの変数を用いることができます．

`$$s` LyX のシステムディレクトリ（例：/usr/share/lyx）

`$$i` 入力ファイル

`$$o` 出力ファイル

`$$b` LyX 一時ディレクトリ内でのベース名 (ファイル拡張子なし)

`$$p` LyX 一時ディレクトリのフルパス名

`$$r` 処理されている元の LyX ファイルのフルパス名

`$$f` LyX ファイルのファイル名 (ディレクトリパスなし)

`$$l` 「LaTeX 名」

最後の変数は，LaTeX の**\include** コマンドで使用されるのと同形式のファイル名です．これは，書き出すファイルがそのようなインクルードに適している場合のみ，使用さるべきものです．

複写子は，出力ファイルに関する操作であれば，ほとんどすべてに対応することができます．たとえば，生成した PDF ファイルを，/home/you/pdf/という特別なディレクトリにコピーしたいものとしましょう．その場合には，以下のようなシェルスクリプトを書きます．

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

これを，例えば/home/you/.lyx/scripts/pdfcopier.sh のような，自身のローカル LyX ディレクトリに保存し，お使いのプラットフォームが必要とするならば，実行可能属性を付与します．それから，ツール▷設定ダイアログのファイル処理▷ファイル

[1]たとえば，ファイルが他のファイル—たとえば画像—を，相対ファイル名を用いて参照している場合，このファイルが一時ディレクトリにコピーされると参照が無効になる場合があります．

形式の中で，PDF(pdflatex) 形式—あるいは他の PDF 形式のうちどれか—を選択し，複写子フィールドに pdfcopier.sh $$i $$o と入力します．

複写子は，LyX 自身が様々な変換に使用します．たとえば，適切なプログラムが検出された場合，LyX は自動的に HTML 形式と HTML (MS Word) 形式の複写子を導入します．これらの形式を書き出す際，複写子は，本体の HTML ファイルだけでなく，関連した様々なファイル（スタイルファイルや画像など）もコピーされるように手配します．これらのファイルはすべて，元の LyX ファイルのあるディレクトリのサブディレクトリに書き込まれます．[2]

3.3. 変換子

各形式間でファイルを変換するために，ご自身の変換子を定義することができます．これは，ツール▷設定▷ファイル処理▷変換子ダイアログで行います．

新規に変換子を定義するには，ドロップダウンリストから変換元の形式と変換先の形式を選択し，変換に必要なコマンドを入力してから追加ボタンを押してください．変換子の定義には，以下のような変数を使用することができます．

`$$s` LyX システムディレクトリ

`$$i` 入力ファイル

`$$o` 出力ファイル

`$$b` 入力ファイルのベースファイル名（拡張子をとった部分）

`$$p` 入力ファイルのパス

`$$r` 元の入力ファイルのパス（変換子が連鎖して呼び出されたときの挙動が $$p とは異なります）

`$$e` 文書エンコーディングの iconv 名

追加フラグフィールドには，以下のフラグをコンマで区切って入力することができます．

`latex=flavor` この変換子が LaTeX の一種を実行することを示します．これによって，LyX の LaTeX エラーログに記録を残せるようになります．オプションの `flavor` 値は実行する LaTeX の形を指定します（ `latex`，`pdflatex`，`platex`，`xetex`，`luatex`）．値が指定されなければ `latex` が用いられます．

[2] この複写子の挙動は調整することができます．非必須の「-e」オプションは，コピーする拡張子をコンマ区切りで羅列したものを引数にとります．これを省略した場合には，すべてのファイルがコピーされます．「-t」引数は，生成したディレクトリに書き加える拡張子を指定するものです．既定値では，これは「LyXconv」となっているので，/path/to/filename.lyx から生成された HTML ファイルは，/path/to/filename.html.LyXconv となります．.

needauth　この変換子が安全でないと見做されていて，ユーザーの許可が必要であることを示します．**ツール ▷ 設定 ▷ ファイル処理 ▷ 変換子** の設定によって，ユーザーは，(a.) 現在の文書を一時的もしくは恒久的に信頼するか否かを尋ねられるか，(b.) セキュリティ上の懸念によって変換は不可能であることを伝えられるか，(c.) 恒久的な同意を与えたので通知されないかのいずれかになります．任意のプログラムを実行する可能性のある変換子には，このフラグを設定してください．

needaux=flavor　変換に LaTeX の.aux ファイルが必要であることを示します．オプションの flavor 値は.aux ファイルを生成するのに実行する LaTeX の形を指定します（ latex，pdflatex，platex，xetex，luatex）．値が指定されなければ latex が用いられます．

nice　バックエンドからの”nice” なファイル，つまり LyX が書き出す LaTeX ファイルのように，input@path のない LaTeX ファイルを必要とします．

xml　出力が XML であることを示します．

以下の 4 つのフラグは key = value 形式の引数をとります（したがって厳密にはフラグとは呼べません）．

hyperref-driver　この変換子が hyperref パッケージとともに読み込む必要のあるドライバ名．一部の PDF 機能を利用するには正しいドライバを読み込む必要があります．詳細については，hyperref の取扱説明書を参照してください．

parselog　これを指定すると，変換子の標準エラーが infile.out ファイルにリダイレクトされ，引数に指定されたスクリプトが script < infile.out > infile.log の形で実行されるようになります．引数には$$s を指定することができます．

resultdir　これには，変換子が生成したファイルをダンプするディレクトリ名を指定します．LyX はこのディレクトリを作成せず，ここに何もコピーしませんが，このディレクトリを宛先にコピーします．引数には，$$b を使用することができ，これはディレクトリがコピーされる際に，入力ファイルおよび出力ファイルのベース名で置換されます．
resultdir と usetempdir は，同時に用いることはできませんのでご注意ください．前者が指定されているときには，後者は無視されます．

resultfile　これは出力ファイル名を指定するもので，$$b を使用することができます．resultdir が指定されているときのみ有効で，必ず用いる必要はありません．指定されていなければ，既定値は「index」です．

LyX とともに導入されている変換子の一部には適切な hyperref-driver が設定されています．しかしながら最後の 3 つは，LyX に前もって導入されている変換子には，現在いずれも使用されておりません．

変換しようとするすべての形式のあいだに変換子を定義する必要はありません．たとえば,「LyX から PostScript」変換子が定義されていないのに，LyX は PostScript を書き出していることに気づかれることでしょう．これは，まず LaTeX ファイルを生成した後に（これには変換子を定義する必要はありません）,「LaTeX から DVI」変換子を使用して DVI に変換し，最後に，得られた DVI を PostScript に変換することによって実現しています．LyX はこのような変換子の「連鎖」を自動的に見つけ，つねに最も短い連鎖を選択します．しかしながら，なお形式間に複数の変換方法を定義することも可能です．たとえば，標準的な LyX 設定は，LaTeX から PDF へ変換するのに，以下の 3 つの方法を用意しています．

1. 直接 pdflatex を使用するもの
2. （DVI と）PostScript を経由して ps2pdf を使用するもの
3. DVI 経由で dvipdfm を使用するもの
4. 直接 XeTeX を使用するもの
5. 直接 LuaTeX を使用するもの

このように代替連鎖を定義するには，第 3.1 節に述べられているように，ターゲットとなる「ファイル形式」を複数定義しなくてはなりません．たとえば，標準設定では，pdf(ps2pdf 用)・pdf2(pdflatex 用)・pdf3(dvipdfm 用)・pdf4(XeTeX 用)・pdf5(LuaTeX 用) と命名された形式が定義されていて，すべて共通の拡張子.pdf を持ち，上記で言及した各変換方法に対応しています．

4. LyX の各国語対応

LyX は，翻訳された操作画面の利用をサポートしています．私たちが最後に確かめたところでは，LyX は 30 言語の翻訳を提供しています．選択した言語は，使用する**ロケール**と呼ばれます（ロケール設定についての詳しい資料は，お使いの基本ソフトに添付のロケール関連説明書をご覧ください．Linux の場合は，マニュアルページの `locale(5)` から見ると良いかもしれません）．

これらの翻訳は適切に機能しますが，欠点もいくつかあることに注意してください．たとえば，ダイアログはすべて英文を念頭にデザインされているため，翻訳文の一部は，割り当てられたスペースに収めるには大きすぎるかもしれません．これは表示上の問題に過ぎず，他の障害は引き起こしません．また，翻訳によっては，すべてのショートカットが定義されていないことに気づかれるでしょう．ショートカットのために空いている文字が十分ないことが時々あるのです．単に翻訳者がまだショートカットを定義していないこともあるでしょう．もちろん，私たちの各国語対応チーム—あなたも参加したいと思われるかもしれません[1]—は，LyX の将来のバージョンでこれらの欠点を修正しようとするでしょう．

4.1. LyX を翻訳する

4.1.1. グラフィカル・ユーザ・インタフェース（テキスト・メッセージ）を翻訳する

LyX は，操作画面の国際化対応に GNU `gettext` ライブラリを使用します．LyX のすべてのメニューやダイアログでお好みの言語を話させたいときには，その言語の `po` ファイルが必要です．このファイルが利用可能であれば，そこから `mo` ファイルを生成して，この `mo` ファイルをインストールしなくてはなりません．この全過程は，GNU `gettext` の取扱説明書に説明があります．この作業をあなたのためだけに行うこともできますが，もしせっかくするのであれば，あなたの骨折りの結果を LyX コミュニティの他の人々と分かち合いませんか．どのように段取りを進めればよいか，詳しくは LyX 開発者メーリングリストにメールを送ってください．

要約すれば，以下のように行います（xx は言語コードを表します）．

- LyX ソースコードをチェックアウトしてください（ウェブ上の情報参照）．
- `lyx.pot` ファイルを`**.po` ファイルのあるフォルダにコピーして，`xx.po` に名前を付け替えてください（`lyx.pot` がどこにもない場合には，コンソールから

[1] もしあなたが英語以外の言語を流暢に操れるならば，これらのチームに参加することは，LyX コミュニティに報いるたいへん素晴らしい方法です！

そのディレクトリで make lyx.pot コマンドを実行し，作成し直すか，他言語の既存の po ファイルをひな型として使用することができます）.

- xx.po を編集します．[2]メニューラベルやウィジェットラベルのうちには，翻訳しなくてはならないショートカットがある場合があります．これらのキーは「|」の後に記されており，当該言語の単語やフレーズに対応して翻訳しなくてはなりません．さらに，新しい po ファイルの冒頭に，あなたの電子メールアドレスなどの情報も書き加えて，人々があなたに提案や，滑稽な怒りのメッセージを届けることができるようにしてください．

もし，あなたがこれを自身のためだけに行っているのであれば，

- xx.mo を生成してください．これは msgfmt -o xx.mo < xx.po でできます．
- この mo ファイルを，お使いのロケールツリー中，言語 xx のアプリケーションメッセージ用の正式なディレクトリにコピーして，lyx.mo という名称にしてください（例：/usr/local/share/locale/xx/LC_MESSAGES/lyx.mo）.

しかしながら前述のように，この新しい po ファイルを他の人たちが使用できるよう，LyX 頒布版に追加できることが最善です．これを追加するには，LyX に変更を加える必要がありますので，もしその気があれば，開発者メーリングリストに電子メールを送ってください．

4.1.1.1. 多義訳語メッセージ

時には，一つの英語のメッセージが，翻訳先の言語では複数のメッセージに翻訳されなくてはならないことが判明することがあります．一つの例は，To というメッセージで，これは英語で「to」がどういう意味を持っているかによって，独語では Nach と訳されたり Bis と訳されたりします．GNU gettext は，このような多義訳語を To の代わりに，To[[as in 'From format x to format y']] や To[[as in 'From page x to page y']] としなくてはなりません．これによって，これら 2 つの To は，gettext には別物と解釈され，それぞれ正しく Nach と Bis に訳すことができるようになります．

もちろん，この文脈情報は，翻訳が存在しないときには取り去られる必要がありますので，メッセージの終わりに二重大括弧で囲わなくてはなりません（上例参照）．LyX の翻訳機構では，メッセージの終わりに二重大括弧で囲われているものはすべて，メッセージを表示する前に取り去るようにされています．

4.1.2. 説明書を翻訳する

（Help メニュー中の）オンライン説明書は翻訳することができます（そして翻訳されるべきです！）．説明書の翻訳版が利用可能であり[3]，ロケールがその言語に設定されてい

[2] これは単なるテキストファイルなので，どのテキストエディタでも編集できます．しかし，Poedit（全プラットフォーム用）や KBabel （KDE 用）のように，この目的の編集をサポートする特別なプログラムがあります．Emacs にも po ファイルを編集するための「モード」があります，第 https://www.gnu.org/software/gettext/manual/html_node/PO-Mode.html#PO-Mode.

[3] 2008 年 3 月現在，説明書の少なくとも一部が翻訳されている言語は 14 言語に上り，入門編が訳されているものはさらにいくつかあります．

る場合，LyXはこれを自動的に使用します．LyXは，翻訳版を`LyXDir/doc/xx/DocName.lyx`（`xx`は現在使用している言語コード）で探します．翻訳文書がない場合には，既定の英語版が表示されます．翻訳版は，原典と同じファイル名（上述の`DocName`）を持っていなくてはならないことに注意してください．説明書を翻訳する気がおありであれば（これは原典の校正としてもたいへん役立ちます！），以下のような点をすぐに行うべきです．

- 説明書翻訳ウェブページ https://www.lyx.org/Translation を確認してください．ここで，どの文書が（もしあれば）お使いの言語に既に翻訳されているかを見つけることができます．また，説明書をお使いの言語に翻訳する作業の面倒を見ている人を（もしいれば）見つけることができます．この作業の面倒を見ている人がいない場合には，私たちにあなたが興味をお持ちであることを知らせてください．

いったん実際の翻訳の仕事に取りかかったならば，トラブルを回避するいくつかのヒントがあります．

- 文書化チームに所属してください！そのための情報が`Intro.lyx`（ヘルプ▷はじめの一歩）にあります．また，この`Intro.lyx`が最初に訳すべき文書です．

- 翻訳しようとする言語での印刷慣行を学んでください．活版印刷は古来の技術であり，何世紀にもわたって世界の至る所で，様々な慣行を発達させてきました．また，あなたの国で活版工が用いる専門用語も学んでください．自分で勝手な専門用語を捻出するとユーザを混乱させるだけです．（**警告！活版技術は病みつきになる可能性があるので注意してくさい！**）

- 文書のコピーをとってください．これを作業用コピーとします．これをお使いの`UserDir/doc/xx/`ディレクトリにコピーすれば，個人用の翻訳ヘルプファイルとして使用することができます．
  【註】外部素材 (画像など) のある複雑な文書の場合，例えば一時ディレクトリなどにコピーを作ると，文書を別の場所に移したとき，外部素材へのリンクは壊れてしまうかもしれないことに注意してください．最も良い方法は，LyX ツリーを git (`https://www.lyx.org/HowToUseGIT` 参照) からとってきて，その doc ファイルを直接編集するのが良いでしょう．

- （LyXチームが維持している）原典の説明書は，時折更新されます．変更点については，https://www.lyx.org/trac/timeline のソースビューアでご覧ください．この方法で，翻訳文書のどの部分を更新しなくてはならないか，たやすく見つけることができます．

もし原典に誤りを見つけたならば，修正して文書化チームの他のメンバーに変更したことを知らせてください（文書化チームに参加することをお忘れになっていませんよね）．

4.2. 国際キー配列

以下の 2 節では，.kmap および .cdef ファイルの文法を詳細に解説します．これらの節は，提供されているキー配列があなたのニーズに合わない場合に，自身用のキー配列をデザインする手助けとなるでしょう．

4.2.1. .kmap ファイル

.kmap ファイルは，打鍵したものを文字や文字列に割り当てます．名前が示唆するように，これはキーボード配列表を定義します．.kmap ファイルは，以下の各項で説明するように，kmap・kmod・ksmod・kcomb のキーワードを定義します．

kmap　　文字を文字列に割り当てる

　　\kmap **文字 文字列**

これは，**文字**を**文字列**に割り当てます．**文字列**中では，二重引用符（"）とバックスラッシュ（\）は，前にバックスラッシュ（\）を付けてエスケープしなくてはならないことに注意してください．
　&を打鍵すると/記号が出力される kmap ステートメントを，一例としてあげると，

```
\kmap & /
```

のようになります．

kmod　　アクセント文字を指定する

　　\kmod **文字 アクセント 許可文字**

これは**文字**を**許可文字**のアクセントとするものです．これはデッドキー[4]機構です．
　文字を打鍵してから**許可文字**にないキーを打鍵すると，**文字**の後に許可文字ではないその文字が出力として表示されます．Backspace はデッドキーを取り消しますので，**文字**→ Backspace と打鍵すると，カーソルは一文字戻ることなく，文字が次の打鍵したものに及ぼしたはずの効力を取り消します．
　以下の例は，’ 文字を acute アクセントとして，a・e・i・o・u・A・E・I・O・U の文字に許可するものです．

```
\kmod ’ acute aeiouAEIOU
```

ksmod　　アクセント文字に例外を指定する

　　\kxmod　**アクセント 文字 結果**

[4] **デッドキー**という用語は，それ自身で文字を出力しないけれども，別のキーを続けて打つと，望んだアクセント文字を出力するキーのことを指し示します．たとえば，独語で $\ddot{a}$ のようなウムラウトのついた文字は，このようにして出すことができます．

これは**文字**上の**アクセント**について例外を指定するものです．ここで**アクセント**には，前出の\kmod 宣言で打鍵キーを既に割り当てられてなくてはならず，**文字**は**アクセント**の**許可文字**の集合に属していてはなりません．こうして**アクセント→文字**の順に入力すると，**結果**が出力されるようになります．.kmap ファイルにこの宣言がない場合には，**アクセント→文字**と入力すると，**アクセントキー→文字**（**アクセントキー**は\kmod 宣言の最初の変数）と出力されます．

以下のコマンドを用いると，acute-i（'i）と入力した場合，äi と出力されるようになります．

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

kcomb　　2 つのアクセント文字を結合する

\kcomb **アクセント** *1* **アクセント** *2* **許可文字**

これはなかなか難解になってきます．これは**アクセント** *1* と**アクセント** *2* を（この順番で）結びつけて，**許可文字**に効果を及ぼすようにします．**アクセント** *1* と**アクセント** *2* の打鍵キーは，ファイル内のこのコマンド**よりも前に**，\kmod コマンドで設定されていなくてはなりません．

greek.kmap ファイル上にある例をとってみましょう．

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

これは;:i を押すと\'{\"{i}}という効果を得るようにするものです．この場合のバックスペースは，最後のデッドキーを取り消すので，;: Backspace i と押した場合には，\'{i}となります．

4.2.2. .cdef ファイル

.kmap による割り当てが行われた後，.cdef ファイルは，記号の作り出す文字列を現在のフォントの文字に割り当てます．LyX 頒布版には，現在のところ，少なくとも iso8859-1.cdef ファイルと iso8859-2.cdef ファイルが含まれています．

一般的に.cdef ファイルは，

セット中の文字番号　文字列

という形の宣言の羅列です．たとえば，\'{e}を iso-8859-1 セットの対応する文字 (233) に割り当てるには，以下の宣言を用います．

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

ここで，**文字列**中の\と"はエスケープされています．同一の文字を二つ以上の文字列に充てることができることに注意してください．iso-8859-7.cdef ファイルには，

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

という例があります．LyXは，キー打鍵やデッドキーの組み合わせで生成される文字列の割り当てを見つけることができないとき，それがアクセント付き文字のように解釈ができないかどうかチェックして，画面上の文字にアクセントを引くことを試みます．

4.2.3. デッドキー

国際文字のサポートを追加する第2の方法として，いわゆるデッドキーによる方法があります．デッドキーは文字と一緒に用いて，アクセント付き文字を生成します．ここではその機能を説明するために，きわめて単純なデッドキーの作り方を説明します．

仮に，曲折アクセント記号「^」が必要になったものとしましょう．この場合，自身の`lyxrc`ファイル中で，^キー（すなわちShift-6キー）を，LyXコマンドの`accent-circumflex`に結びつけることができます．すると^キーの後に文字を打ったときはいつでも，この文字上に曲折アクセントが付けられるようになります．たとえば「^e」という打鍵順は「ê」という文字を生成します．しかしながら，もし「^t」と打鍵したならば,「t」は曲折アクセントをとることは決してないために，LyXはビープを鳴らして文句を付けます．デッドキーの後にスペースを打つと，アクセントだけが生成されます．この最後の点に注意してください．あるキーをデッドキーに割り当てる場合には，このキー上の文字を別のキーに割り当て直す必要があります．たとえば，, キーをセディーユに割り当てるのはよくありません．コンマを入力しようとすると必ずセディーユが出てくるようになるためです．

デッドキーを割り当てるのによく用いられる方法は，Meta-・Ctrl-・Shift-キーを,「~」・「,」・「^」のようなアクセントと一緒に用いる方法です．また，`xmodmap`や`xkeycaps`を使って，特別な`Mode_Switch`キーを設定する方法もあります．`Mode_Switch`キーは，ちょうどShiftキーのように機能するので，アクセント文字を割り当てるのに使用できます．また，特定のキーを`usldead_cedilla`などに割り当てることで，これらのキーをデッドキーに仕立て，このシンボリックキーを対応するLyXコマンドに割り当てることもできます[5]．この`Mode_Switch`キーには，Ctrl-キーの片方や使われていないファンクションキーなど，ほぼ何でも指定することができます．アクセントを生み出すLyXコマンドについては，*LyX* **関数説明書**の`LFUN_ACCENT_*`の項をご覧ください．ここには完備した一覧があります．

4.2.4. 自分の言語設定を保存する

ツール▷設定ダイアログを使えば，LyXを起動したときに，ご希望の言語環境に自動的に設定されるように，設定を編集することができます．

[5] John Weiss からの註：これはまさに私が，自分の`~/.lyx/lyxrc`と`~/.xmodmap`で行っていることです．私は，Scroll Lockキーを`Mode_Shift`に仕立てて，多数の`usldead_*`シンボリックキーをScroll Lock-^やScroll Lock-~などに割り当てています．私はこの方法でアクセント文字を入力しています．

5. 文書クラス・レイアウト・ひな型の作成と新規導入

この章では，新しく LyX のレイアウトファイルやひな型ファイルを作成して，導入する手順を説明すると共に，新規に LaTeX 文書クラス（ドキュメントクラス）を正しく導入する方法を復習します．

まず，LyX と LaTeX の間の関係をどのように考えるべきか，若干の註釈を加えておくことにしましょう．理解していただきたいことは，ある意味において，LyX は，LaTeX について何も知らないと云うことです．実際のところ，LyX の観点からは，LaTeX は，LyX が出力を生成することができる，複数の「バックエンド形式」のうちの一つに過ぎないということです．同種のバックエンド形式には，DocBook・平文・XHTML があります．もちろん LaTeX は，とくに重要な形式ですが，LyX が LaTeX について持っている情報のほとんどは，実はプログラム本体には含まれていないのです[1]．このような情報は，`article.cls` のような標準クラスでも，「レイアウトファイル」に保管されています．同様に，LyX は，DocBook や XHTML についてもほとんど知りません．LyX が知っていることは，レイアウトファイルの中にあります．

文書クラス用のレイアウトファイルは，LyX 構成体—対応する様式や何らかの差込枠などを有する段落群—と，それに対応する LaTeX 構成体・DocBook 構成体・XHTML 構成体との間の翻訳指南書のようなものです．たとえば，LyX が `article.cls` について知っていることのほとんど総ては，`article.layout` と，それが呼び出す他の様々なファイルに書き込まれています．このことから，レイアウトファイルを書こうとする人は，既存のファイルを研究することを勧めます．とっかかりとしては，`article.layout` や `book.layout` や，文書クラス用の他のレイアウトファイルに取り込まれている `stdsections.inc` から見始めるのがよいでしょう．このファイルは，節などの定義が為されている場所です．`stdsections.inc` は，節様式や小節様式などとしてマークされている段落を，対応する LaTeX・DocBook・XHTML のコマンドやタグにどのように翻訳すべきかを LyX に知らせるものです．基本的に `article.layout` ファイルは，これらの `std*.inc` ファイルを取り込んでいるだけのものです．

しかしながら，LyX–LaTeX 間の対応を定義するだけが，レイアウトファイルが行うことではありません．レイアウトファイルが行うもう一つの仕事は，LyX 構成体自身が画面上にどのように表示されるべきかを定義することです．この 2 つの仕事は全く独立したものであるので，レイアウトファイルが 2 つの仕事を行うという事実は，しばしば混乱を引き起こす元となります．ある段落様式を LaTeX に翻訳する仕方を LyX に指示することは，その表示の仕方を LyX に指示するものではありません．逆に，ある段落様式の表示の仕方を LyX に指示することは，その段落様式をどのよう

[1] 過度に複雑なため，LyX に「ハードコード化」されているコマンドもありますが，一般的に開発者は，これを「わるいこと」とみなしています．

に LaTeX に翻訳するかを LyX に指示するものではありません（ましてや LaTeX に表示の仕方を指示するものではありません）．つまり，一般的に，新しい LyX 構成体を定義する際には，(i) LaTeX にどのように翻訳するかを LyX に指示する，(ii) それをどのように表示するかを LyX に指示する，という，二つのかなり異なることを行わなくてはならないのです．

もちろん，LyX の他のバックエンド形式に関しても，ほぼ同じことが言えますが，XHTML の場合には若干事情が異なり，LyX が，ブラウザ中での段落の表示方法を（CSS の形で）出力するにあたって，当該段落を LyX が画面上に出力する仕方の情報を，ある程度利用することができます．しかし，この場合でも，LyX が内部的に行うことと，外部的に行う物事の間の区別は，依然として有効であり，この 2 つは独立して制御することができます．詳細に関しては，第 5.4 節をご覧ください．

5.1. 新しい LaTeX ファイルの導入

頒布版によっては，LyX で使いたい LaTeX パッケージやクラスファイルが含まれていないことがあるかもしれません．たとえば，オーバーヘッドプロジェクタ用のスライドを準備するためのパッケージである，FoilTeX がないかもしれません．TeXLive（2008 年以降）や MiKTeX のような最近の LaTeX 頒布版には，これらのパッケージを導入するためのユーザーインタフェースが用意されています．たとえば，MiKTeX では，付属の「Package Manager」プログラムを起動すると，利用できるパッケージの一覧を得ることができます．どれかを導入するには，その上で右クリックするかツールバーボタンを押してください．

お使いの LaTeX 頒布版がこのような「パッケージマネジャー」を提供していなかったり，使用中の頒布版にそのパッケージが入っていない場合には，以下のステップに従って手動で導入してください．

1. CTAN などから欲しいパッケージを入手してください．

2. パッケージに「`.ins`」で終わるファイル名が入っている場合（FoilTeX がその一例です）は，コンソールを開いて，このファイルのフォルダに移動し，コマンド `latex foiltex.ins` を実行してください．すると，パッケージが解凍されて，導入すべきすべてのファイルが展開されます．たいていの LaTeX パッケージは圧縮されていないので，このステップは飛ばすことができます．

3. ここで，パッケージを全ユーザーに使用可能にするか自分自身だけで使うかを決定する必要があります．

   a) （Linux・OSX などの）*nix 系システムでは，システム上の全ユーザーに新パッケージを利用可能にしたければ，「ローカル」TeX ツリーに導入し，そうでなければ「ユーザー」TeX ツリーに導入してください．これらのツリーが存在しない場合にどこに作成すればよいかは，お使いのシステムに依存します．これを見いだすには，`texmf.cnf` ファイルを参照してください[2]．「ローカル」TeX ツリーの場所は，`TEXMFLOCAL` 変数で定義され

[2] このファイルは，通常$TEXMF/web2c ディレクトリにありますが，コマンド `kpsewhich texmf.cnf` を実行してその場所を見つけることもできます．

ており，通常は/usr/local/share/texmf/や/usr/local/texlive/XXXX (XXXX は導入されている TeXLive 頒布版の年次) のような場所になっています.「ユーザー」TeX ツリーの場所は，TEXMFHOME で定義されており，通常は$HOME/texmf/や$HOME/.texliveXXXX です（もしこれらの変数が事前定義されていなければ，定義しなくてはなりません）.「ローカル」ツリーを作成したり変更したりするには，おそらく root 権限が必要ですが,「ユーザー」ツリーにはこのような制限はありません.
一般的に，システムをアップグレードした際に，ユーザーが修正されたり上書きされたりということが起こらないので，ユーザーツリーに導入することが推奨されます．こうすると，自分のホームディレクトリをバックアップする際に，パッケージも他のものと一緒にバックアップされます（もちろん通常行われるようにすればの話です）.

b) Window で，システム上の全ユーザーに新パッケージを利用可能にしたい場合には，LaTeX の導入されているフォルダのサブフォルダ~\tex\latex に移動します（MiKTeX の既定値では，これは~:Programs\MiKTeX\tex\latex です）[3]．ここに新規フォルダ foiltex を作成し，パッケージの全ファイルをそこにコピーしてください．パッケージを自分だけで使用したい場合や，admin 権限を持っていない場合には，ローカル LaTeX フォルダで同じことを行います．たとえば MiKTeX 2.8 では，これは WinXP 上では
~:\Documents and Settings\<ユーザー名>\Application Data\
MiKTeX\2.8\tex\latex
フォルダ，WinVista 上では
~:\Users\<ユーザー名>\AppData\Roaming\2.8\MiKTeX\tex\latex
フォルダになります.

4. ここまで来れば，あとは LaTeX に新しいファイルがあることを告げるだけです．これは使用している LaTeX 頒布版に依存します.
   a) TeXLive の場合には，コンソールから texhash コマンドを実行してください．パッケージを全ユーザー用に導入した場合には，おそらく root 権限で行う必要があります.
   b) MiKTeX では，パッケージを全ユーザー用に導入した場合には,「Settings (Admin)」を起動し,「Refresh FNDB」と記してあるボタンを押してください．そうでない場合には,「Settings」を起動して同様に行ってください.
5. 最後に，LyX に新しいパッケージがあることを告げなくてはなりません．そこで，LyX からツール▷環境構成メニューを実行して，LyX を再起動します.

これでパッケージが導入されました．この例では，文書クラス Slides (FoilTeX) が文書▷設定▷文書クラスで利用可能になっているはずです.
文書▷設定▷文書クラスメニューに列挙されてもいない LaTeX 文書クラスを使用したい場合には，その「レイアウト」ファイルを作り出さなくてはなりません．これが次節のトピックです.

[3] これは，英語版でのみ正しいパスになっています．独語版では~:Programme\MiKTeX\tex\latex となり，他の言語でも同様です.

5.2. レイアウトファイルの型

この節は，レイアウト情報を含む各種 LyX ファイルについて述べます．これらのファイルは，各種段落様式や文字様式についての記述がされているものであり，LyX がそれらをどのように表示すべきなのか，また，それらをどのように LaTeX や DocBook, XHTML その他の出力形式に翻訳すればよいのかが記されています．

ここでは，レイアウトファイル作成過程の包括的な解説を試みたいと思いますが，LaTeX クラスだけでもサポートする文書の種類があまりにたくさんあるので，読者が出会いうるケースや問題をすべてカバーすることはとても望めません．LyX ユーザーメーリングリストには，自身の経験を人々と分かち合いたいと望む，レイアウトデザインの経験豊かな人々がよく顔を出していますので，気軽に質問を投げかけてみてください．

新しいレイアウトを準備するに当たっては，LyX と共に頒布されているレイアウトの例を見るのがたいへん役立ちます．他の人々も使用できる LaTeX 文書クラス用の LyX レイアウトを作ったり，他の人々にも有用なモジュールをお書きになった場合には，LyX Wiki のレイアウトに関する節か，LyX 開発者メーリングリストに投稿して，LyX 頒布版に同梱することができるようにしてください [4]．

5.2.1. レイアウトモジュール

ここまで,「レイアウトファイル」についてお話してきました．しかし，レイアウト情報を含むものには，他の種類のファイルもあります．厳密にレイアウトファイルと呼ぶとき，それは .layout 拡張子を持ち，文書クラスに関する情報を LyX に提供するものを指します．しかしながら，LyX 1.6 以降，レイアウト情報は，拡張子が .module のレイアウト**モジュール**にも含めることができます．レイアウトが LaTeX クラスに対応しているように，モジュールは LaTeX パッケージに対応するものであり，endnotes モジュールのように，特定のパッケージにサポートを提供するためのモジュールもあります．レイアウトモジュールは，特定の文書レイアウトに特化したものではなく，多くのレイアウトで使用できるという意味において，一面，stdsections.inc 等のインクルードファイル [5] のようなものです．相異なる点といえば，article.cls でインクルードファイルを使用するには，そのファイルを編集しなくてはなりませんが，モジュールの場合は，文書 ▷ 設定ダイアログで選択するだけですみます．

モジュール作成は，新しく段落様式を一つ加えたり，自由差込枠を加えたりするだけで済むことも多いので，レイアウト編集を学ぶ上でもっとも易しい方法です．しかし原理的には，レイアウトファイルに入れることのできるものはすべて入れることができます．

新しいモジュールを作成し，それを layout/ フォルダにコピーした後，モジュールがメニューに現れるようにするためには，LyX の環境構成を行って再起動しなくてはなりません．しかしながら，モジュールの修正の場合には，文書 ▷ 設定を開いてどれかを選択し「OK」を押せば，直ちに反映されます．**これを実行する前に，作業中の文書を保存しておくことを強く勧めます**．もっと言えば，**実際の文書で作業している**

[4] LyX は General Public License の下でライセンスされていますので，LyX に寄贈されたものは総て同じライセンス下に置かれることに注意してください．

[5] これらは任意の拡張子をつけることができますが，慣習的に .inc 拡張子が用いられます．

ときに，同時にモジュールの編集をしようとしないことを強く勧めます．もちろん開発陣は，そのような場合でも LyX が安定性を維持するように努力していますが，あなたが作成したモジュール中の文法エラー等によって，奇妙な挙動が引き起こされることがあるからです．

5.2.1.1. ローカルレイアウト

LyX に於けるモジュールは，LaTeX に於けるパッケージに当たります．しかしながら，特定の差込枠や文字様式を，ひとつの文書のためだけに作りたいこともあるでしょうから，そのような場合に，他の文書でも利用できるようなモジュールをわざわざ書くことには，あまり意味がありません．このような場合に必要となるのが，「ローカルレイアウト」なのです．

これは，文書▷設定▷ローカルのレイアウトにあります．そこにある大きなテキストボックスは，本来ならレイアウトファイルやモジュールに入力すべきものを，入力するためのものです．特定の文書のローカルレイアウトは，その文書専属のモジュールだと考えることができます．したがって，`Format` タグは挿入しなくてはなりません．どの書式を使用しても構いませんが，通常は，執筆時点の最新書式を用いることになるでしょう (LyX では，最新書式はです)．

ローカルのレイアウト面に何かを入力すると，下部にある「検証」ボタンが有効になります．このボタンを押すと，ユーザーの入力したものが，指定された書式に沿って，有効なレイアウト情報になっているかどうかが検証されます．すると，LyX がその結果を返しますが，残念ながら，エラーがあった場合にそれが何のエラーであるかは返しません．しかしながら，LyX をターミナル (擬似端末) から起動した場合には，そのエラーがターミナルに返されます．ローカルレイアウトは，正しい書式で入力されないうちは，保存することができません．

ここで，前節と同じ警告があります．ローカルレイアウトは，作業中の文書上で，特に保存していない文書上ではいじらないでください．それに注意すれば，テスト用文書上でローカルレイアウトを用いるのは，レイアウトのアイディアを試すのに便利ですし，モジュール開発の第一歩としても便利です．

5.2.2. .sty ファイル用のレイアウト

新しく LaTeX 文書クラスをサポートしようとするとき，LaTeX 2_ε クラス（`.cls`）ファイルが絡む場合と，スタイル（`.sty`）ファイルが絡む場合の 2 つの状況があり得ます．スタイルファイルのサポートは，通常は，かなり容易ですが，新しくクラスファイルをサポートすることは，もう少し難しくなります．この節では，前者について述べることとし，後者については次節に譲ります．

この易しい方の場合では，新しい文書クラスは，既にサポートされている文書クラスと共に使うスタイルファイルとして提供されています．例示のために，スタイルファイルは myclass.sty という名称で，標準的なクラスである report.cls と共に用いられるものと仮定しましょう．

既存の文書クラスのレイアウトファイルを，以下のように，お使いのローカルディレクトリにコピーすることから始めてください[6]．

[6] もちろん，どのディレクトリがローカルディレクトリとなるかは，プラットフォームに依存します．

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

それから，myclass.layout を編集して，

```
\DeclareLaTeXClass{report}
```

という行を

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

のように変更してください．それから，ファイル冒頭辺りに

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

と書き加えてください．

LyX を起動してツール▷環境構成を選択してください．それから LyX を再起動し，新規文書を作成してみてください．すると，文書▷設定ダイアログの文書クラスオプションに「report (myclass)」が現れるはずです．新しいクラスにおいて，節区切り用コマンドなどの一部が，基礎となったクラス—この例では report—とは違う挙動をすることはよくありますので，希望に応じて，各節の設定をいじると良いでしょう．各節のレイアウト情報は，stdsections.inc に含まれていますが，このファイルをコピーしたり変更したりする必要はありません．代わりに，自身のレイアウトファイル中，stdsections.inc も取り込む Input stdclass.inc の後に変更を加えるだけです．たとえば，章見だしのフォントをサンセリフ体に変更するには，以下のような行を加えます．

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

これは，既存の章様式宣言を上書き（あるいはこの場合には追加）します．

新しいパッケージでは，基礎となったクラスには存在しないコマンドや環境を提供することもできます．この場合には，これらをレイアウトファイルに加えます．そのやり方については，第 5.3 節の情報を参照してください．

もし myclass.sty が他の文書クラスで使用することができたり，あるいはできない場合でも，基礎となるクラスから読み込むことのできるモジュールを書くのが最も簡単であることがわかるでしょう．最も簡単なモジュールの例としては，以下のようなものになります．

LyX では，起動時に-userdir オプションを指定することによって，ローカルディレクトリを指定することも可能です．

```
#\DeclareLyXModule{My Package}
#DescriptionBegin
#Support for mypkg.sty.
#DescriptionEnd
Format 69
Preamble
    \usepackage{mypkg}
EndPreamble
```

もう少し複雑なモジュールでは，既存の構成物の挙動を修正したり，新しい構成物を定義したりすることになるでしょう．この辺りの議論については，第 5.3 節を参照してください．

5.2.3. .cls ファイル用のレイアウト

これには 2 つのケースがありえます．ひとつは，クラスファイル自体が既存の文書クラスに立脚している場合です．たとえば，多くの学位論文用クラスは book.cls に基づいています．お使いのものがどうであるかを見るには，クラスファイル中に

```
\LoadClass{book}
```

という行がないかどうか探してください．もしこれがあれば，DeclareLaTeXClass 行は異なりますが，おおよそ前節のように進めることができます．あなたが新しく作るクラスが thesis であり，book クラスに基づいていれば，DeclareLaTeXClass 行は以下のようにします[7]．

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

他方，新しいクラスが既存のクラスに基づいていない場合には，おそらくあなた自身のレイアウトをしたためる必要があります．もし可能であれば，類似した LaTeX クラスを使用している既存のレイアウトファイルをコピーして，それに修正を加えるようにすることを強くお勧めします．少なくとも，どの項目を考慮すべきかがわかるように，既存のファイルを作業の開始点としてください．

5.2.4. ひな型を作成する

新しい文書クラス用のレイアウトファイルを書いたならば，そのレイアウト用のひな型も書くことを検討されるかもしれません．ひな型は，内容はダミーですが，レイアウトの使い方を示す一種のチュートリアルとして動作します．もちろん，イメージを得るために，LyX 添付のひな型をあれこれ見てみるのもよいでしょう．

ひな型は，通常の文書と同様，LyX を使って作成することができます．唯一違う点は，通常の文書では，フォント構成や用紙寸法を含め，すべてのあり得る設定が為

[7] さらに LyX は，文書クラス名がレイアウトファイル名と同じだと仮定するので，クラスファイルを thesis.layout という名前で保存するのが最も簡単です．

されている点です．これらの場合，通常ユーザーはひな型が彼の設定値を上書きすることを望みません．この理由から，ひな型の設計者は，\fontscheme や\papersize などの対応するコマンドをひな型 LyX ファイルから取り除く必要があります．これは，たとえば vi や notepad のような，どの軽いテキストエディタでも行うことができます．

編集したひな型を UserDir/templates/に置き，グローバルなひな型ディレクトリ LyXDir/templates/から使用したいものを同じ場所にコピーし，ツール▷設定▷パスダイアログのひな型パスを再定義してください．

ところで，特別な意味を持つひな型 defaults.lyx があることに注意してください．このひな型は，ファイル▷新規を使って新規文書を作成する際，便利な既定値を提供する目的で必ず読み込まれます．このひな型を LyX 内部から作成するのにしなくてはならないことは，対応する設定を持つ文書を開き，文書既定値として保存ボタンを押すことです．

5.2.5. 旧レイアウトファイルの更新

レイアウトファイルの書式は，LyX のリリース毎に変更されますので，古いレイアウトファイルは変換されなくてはなりません．LyX が古い書式のレイアウトファイルを読み込むと，LyX は，自動的に変換ツール LyXDir/scripts/layout2layout.py を呼び出し，それを現在の書式の一時ファイルに変換します．元のファイルは変更を加えられずに措かれます．もしこのレイアウトファイルをよく使うならば，LyX がこれを毎回行なわくて済むように，レイアウトファイルを恒久的に変換しておきたいと思うかもしれません．これを行うには，以下のように変換子を手動で呼び出してください．

1. ファイル myclass.layout を myclass.old に改称
2. 以下のコマンドをコール
   python LyXDir/scripts/layout2layout.py myclass.old myclass.layout
   ここで LyXDir は LyX システムディレクトリの名前です．

手動変換は，インクルードされているファイル内部の変更までは取り扱いませんので，それらのファイルは別に変換されなくてはなりません．

5.2.6. 引用エンジンファイル

citeengines/サブディレクトリ以下に収められている，いわゆる*.citeengine ファイルは，レイアウトファイルの特殊形です．これらの目的は，natbib・jurabib・biblatex など，書誌情報を生成する LaTeX パッケージの詳細を定義することですが，通常の（追加パッケージのない）BibTeX 引用が LyX 中でどのように取り扱われるかも，これらのファイルの中で定義されます．

より具体的には，どのパッケージを LyX が読み込む必要があるのか，どの引用コマンドが利用可能であるのか，これらが LyX 中（作業領域・ダイアログ・コンテクストメニュー）で，さらには XHTML や平文出力中でどのように表示されるのかを定義します．さらに，これらのファイルは，使用できる派生様式（著者-刊行年，数

値など）とその子細を特定します．引用エンジンファイルは，文書▷設定...▷書誌情報▷様式整形子で利用できる選択肢を生成するのにも使用されます．

引用エンジンファイルは，本質的に通常のレイアウトファイルであり，理論上どのようなレイアウト情報も内包することができますが，通常は`MaxCiteNames`・`CiteFramework`・`CiteEngine`・`CiteFormat`ブロックのような特定のパラメータをもっぱら含みます．この最後の2つの文法は，ファイル自身に加え，第5.3.14節と第5.3.15節に述べられています．

5.3. レイアウトファイルの書式

以下の各節では，いよいよ自分の手を汚してレイアウトファイルを作成したり編集したりする段階になった際，直面することの説明を行います．私たちからのアドバイスとしては，ゆっくりと進めるようにして，ちょっと進むごとに保存やテストを行い，心休まる音楽を聴き，好きな大人の飲み物を一二杯口にしながら行うのがいいでしょう．特に行き詰まってしまったときにはそうです．実際にはそんなに難しいことではないのですが，特に一度に多くのことをやろうとすると，選択肢が多くありすぎて圧倒されてしまうのです．さて，もう一杯大人向け飲料をどうぞ．適量ね．LyXの既存のレイアウトを例や参考に使ったり，既存のレイアウトを自身の目的に合わせて修正したりすると，作業が容易になります．

この章で述べられているタグは，すべて大文字小文字を区別しません．つまり，`Style`・`style`・`StYlE`は同じコマンドとなります．機能名の後にある角括弧は，その機能が取り得る値を示します．テキストクラス設定内で機能が特定されていない場合には，既定値は**強調**で表記されます．引数が「文字列」や「浮動小数点型」などのデータ型をとる場合には，既定値は浮動小数点型=**既定値**のように表示されます．

5.3.1. 文書クラス宣言と分類

レイアウトファイル中の#で始まる行はコメントです．この規則には一つだけ例外があります．すべての`*.layout`ファイルは，以下のような行で始めなくてはならないのです．

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

2行目と3行目は，LyXが環境構成を行う際に用いられます．このレイアウトファイルは，LaTeXスクリプト`chkconfig.ltx`が，#を無視する特別なモードで読み込みます．1行目は単なるLaTeXコメントですが，2行目にはテキストクラスの宣言が，3行目にはクラスの分類(非必須)が書かれています．これらの行が`article.layout`と名付けられたファイルにあると，`article`（レイアウトファイル名）という名称のテキストクラスを定義し，LaTeX文書クラス`article.cls`を使用するようになります（既定ではレイアウトと同じ名称のものを使用します）．上記に現れる「Article (Standard Class)」という文字列は，文書▷設定ダイアログのテキストクラスの説明に使用されます．分類(例中の「Articles」)は，文書▷設定ダイアログで使用され，文書クラスは

この分類によってグループ化されます (分類は通常ジャンルを表し，典型的なものには,「Articles」「Books」「Reports」「Letters」「Presentations」「Curricula Vitae」等があります)．分類を宣言しない場合には，このクラスは「Uncategorized」グループに属することになります．

節見出し表示に変更を加えた，article.cls 文書クラスを使用するテキストクラスを自分で書いたものとしましょう．これを myarticle.layout というファイルに置いたとすると，このファイルのヘッダは以下のようになります．

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

これは，LaTeX 文書クラス article.cls に関連づけられ,「Article (with My Own Headings)」と表示される，myarticle テキストクラスを宣言するものです．もしこのテキストクラスが複数のパッケージに依存するならば，以下のように宣言すると良いでしょう．

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

これは，このテキストクラスが foo.sty パッケージを使用することを示しています．

これらの宣言には，文書クラス名を宣言する非必須パラメータ（ただしリストではない）を与えることができることに注意してください．

できる限り明示的に要約すると，レイアウト宣言は以下の形をとります．

```
#  \DeclareLaTeXClass[クラス, パッケージ名.sty]{レイアウトの説明}
#  \DeclareCategory{分類}
```

ここで「クラス」は，LaTeX クラスファイル名とレイアウトファイル名が異なるときのみ，指定する必要があります．クラスファイル名が指定されなければ，LyX は単純に，クラスファイル名がレイアウトファイル名と同じであると仮定します．

テキストクラスがあなたの嗜好に合うように修正できたならば，他にしなくてはならないことは，それを LyXDir/layouts/か UserDir/layouts にコピーし，ツール▷環境構成を実行し，LyX を終了して再起動するだけです．そうすれば，この新しいテキストクラスが，他のテキストクラスと同様に使用できるようになります．

レイアウトファイルが導入されたならば，これを編集して，環境構成したり LyX を再起動したりすることなく，その変更を確認することができます[8]．LyX 関数 layout-reload を使用すれば，現在使っているレイアウトの再読み込みを強制することができるのです．この関数への既定のキー割り当てはありません—もちろん自分でどれかのキーに割り当てることもできますが—．しかし，通常は，この関数を使用する場合は，これをミニバッファに入力します．

[8]第 1.6 版よりも前の LyX では，これを行うことはできませんでした．その結果，レイアウトファイルに加えた変更を反映させるには，その度に LyX を再起動しなくてはならなかったので，レイアウトファイルを編集する作業は，たいへん時間を浪費する作業だったのです．

注意：layout-reload はかなり「高度な機能」です．この機能を利用する前に，作業中の文書を保存しておくことを**強く**勧めます．もっと言えば，大事な文書の作業をしているときに，同時にレイアウト情報の編集をしようとしないことを**強く**勧めます．テスト用文書を使用してください．レイアウトファイル中の文法エラー等が奇妙な挙動を引き起こす可能性があります．特に，そのようなエラーが起こると，LyX は現在のレイアウトが無効であるものと判断して，別のレイアウトに切り替えようとする可能性があります[9]．LyX 開発陣は，このような状況下でも安定性を保つよう努力していますが，後悔よりも安心の方が良いでしょう[10]．

5.3.2. モジュール宣言

モジュールは，以下のような行で始まらなくてはなりません．

```
#\DeclareLyXModule[endnotes.sty]{Endnotes}
#\DeclareCategory{Foot- and Endnotes}
```

波括弧内に入っている\DeclareLyXModule の必須引数はモジュール名で，これは文書▷設定内に表示されます．角括弧内の引数は非必須です．これは，モジュールが依存する LaTeX パッケージをすべて宣言します．また，非必須引数として，変換元-> 変換先の形を使用することができます．これは，変換元形式から変換先形式への変換鎖が存在するときのみ，このモジュールを使用できることを宣言するものです．\DeclareLyXModule 宣言は，厳密には必須ではありませんが，モジュールを見つけやすくするために書いておくべきです．既存のモジュールカテゴリを見て，適切ならばそのどれかを使用してください．

モジュール宣言とカテゴリ宣言の後には，以下のような行を続けます[11]．

```
#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
```

ここで説明（Description）は，文書▷設定でこのモジュールが何をするものか，ユーザに情報を与えるために使用されます．Requires 行は，このモジュールが共に使用する必要がある，他のモジュールを特定するのに用いられます．一方，Excludes 行は，このモジュールが共に使用してはならない，他のモジュールを特定するのに用いられます．この 2 つの行は必須ではなく，上記のようにモジュールが複数ある場合に

[9] 非常に悪質な文法エラーの場合には，LyX が終了してしまうことさえあります．これは，ある種のエラーでは，LyX がレイアウト情報を全く読めなくなる可能性があるからです．ご注意ください．

[10] 重ねての助言ですが，つねにバックアップを取ってください．それから，お母さんのお片づけに注意．

[11] モジュールを LyX に公開する場合には英語が望ましいです．この説明は翻訳メッセージ一覧に現れるようになるので，次回のインタフェースの更新時に翻訳されることになります．

は，パイプ記号「|」で区切らなくてはなりません．Requires に指定されたモジュールは，選言的に取り扱われることに注意してください．つまり，Requires に指定されたモジュールのうち，**少なくとも一つ**が使用されていればよいということです．同様に，Excludes に指定されたモジュールは，一つも使用されてはなりません．ここでモジュールは，.module 拡張子を除いたファイル名で認識されることに注意してください．つまり somemodule とは，実のところ somemodule.module に他なりません．

5.3.3. 引用エンジンファイルの宣言

引用エンジンファイルは，次のような行で始まらなくてはなりません．

```
#\DeclareLyXCiteEngineModule[biblatex.sty]{Biblatex}
```

波括弧内の必須引数はモジュール名で，文書 ▷ 設定 ▷ 書誌情報にそのまま現れます．角括弧内の引数は非必須です．これは，引用エンジンが依存する LaTeX パッケージを宣言します．

引用エンジン宣言には，その後，以下のような行が続きます [12]．

```
# DescriptionBegin
#   Biblatex supports many author-year and numerical styles.
#   It is mainly aimed at the Humanities. It is highly
#   customizable, fully localized and provides many features
#   that are not possible with BibTeX. The use of 'biber' as
#   bibliography processor is advised.
# DescriptionEnd
```

この説明は，文書 ▷ 設定 ▷ 書誌情報の中で，ユーザーに引用エンジンに関する情報を提供するために使われます．

5.3.4. 書式番号

レイアウトファイルやインクルードされたファイル，またはモジュールの最初の非コメント行には，以下のように，かならずファイル形式番号が記されていなくてはなりません．

Format [整数型] このレイアウトファイルの書式

このタグは LyX 1.4.0 で導入されました．LyX 1.3.x 以前のレイアウトファイルには，明示されたファイル形式がないため，書式 1 と解されます．LyX 現行版のファイル形式は，書式 69 です．しかし，LyX の各版は，旧版の LyX で作成されたファイルを読むことができるように，旧版のレイアウトファイルも読むことができます．しかしながら，以前の書式に変換する方法はありません．

[12] LyX とともに公表するモジュールの場合は，英語で書かれることが望まれます．この説明は，翻訳対象メッセージのリストに含まれますので，次回インタフェースの更新の際に翻訳されることになります．

5.3.5. 汎用テキストクラスパラメータ

以下は，文書クラス全体の挙動を決定する汎用パラメータです（これは，.layout ファイルのみに使用されるべきで，モジュールでは使ってはならない，ということを意味するものでは**ありません**．モジュールには，すべてのレイアウトタグを使用することができます）．

AddToCiteEngine <エンジン> 引用参考文献の表示能力を拡張します．詳細については、第 5.3.14 節を参照してください.「End」で閉じる必要があります.

AddToHTMLPreamble この文書クラスが XHTML に出力されるときに，<head>ブロックに追加出力される情報です．典型的には，これは CSS スタイル情報を出力するのに用いられますが，<head>に出力するものであれば，何でも使用することができます.「EndPreamble」で閉じる必要があります.

AddToPreamble 文書プリアンブルに書き加えられる情報です.「EndPreamble」で閉じる必要があります.

BibInToc [*0*,1] この文書クラスが書誌情報を目次に入れるとき，このオプションの値を 1（もしくは true）にしてください．これによって，書誌情報が 2 回目次に現れるのを防ぐことができます.

CiteEngine <エンジン> 文献参照を表示する方法を定義します．詳細については，第 5.3.14 節をご覧ください.「End」で閉じる必要があります．主に引用エンジンファイルで使用されます（第 5.2.6 節参照）．これをレイアウトファイルやモジュールに追加した場合，引用エンジンの定義は，すべて上書きされることに注意してください．AddToCiteEngine も参照してください.

CiteFormat 書誌情報の表示に使う書式を定義します．詳細については，第 5.3.15 節をご覧ください.「End」で閉じる必要があります．主に引用エンジンファイルで使用されます（第 5.2.6 節参照）．レイアウトやモジュールに CiteFormat が指定されると，引用エンジンの定義は上書きされます.

CiteFramework [*bibtex*,biblatex] 書誌情報を生成するのに，Biblatex を用いるのか BibTEX を用いるのかを特定します．主に引用エンジンファイルで使用されます（第 5.2.6 節参照）.

ClassOptions 文書クラスがサポートする様々な大域オプションを記します．説明は，第 5.3.6 節を参照してください.「End」で閉じる必要があります.

Columns [*1*,2] 文書クラスが既定で 1 段組か 2 段組かを指定します．文書▷設定ダイアログで変更することができます.

Counter [文字列] この部分はカウンタの特性を定義します．カウンタがまだ存在していなければ，生成されます．もし存在しなければ修正されます.「End」で閉じる必要があります.
カウンタについての詳細は，第 5.3.12 節を参照してください.

DefaultFont 文書を表示するのに用いられる既定フォントを設定します．フォントの宣言の仕方については，第 5.3.13 節を参照してください.「EndFont」で閉じる必要があります.

DefaultModule [<モジュール>] この文書クラスに，既定で取り込むモジュールを指定します．モジュールは，.module 拡張子を除いたファイル名で指定します．ユーザはこのモジュールを除外することができますが，当初は有効の状態になっています（これは新しいファイルが作成されたときや，既存の文書にこの文書クラスが選択したときのみ該当します）.

DefaultStyle [<様式>] これは新規段落に割り当てられる様式であり，通常は標準です．もしこれを指定しなければ，最初に定義される様式がこれに割り当てられるようにはなっていますが，このディレクティブを使用することが推奨されます.

DocBookRoot [文字列] 文書を DocBook のこのクラスで出力する際に使用する（文書の最上位の）ルート要素．既定値は「article」です.

DocBookForceAbstract [ブール値]「true」ならばルート要素は常に<abstract>タグを持つことになります．既定値は「false」です.

ExcludesModule [<モジュール>] このタグは，指定されたモジュール—.module 拡張子を除いたファイル名で指定します—が，この文書クラスでは使用できないように設定します．これはたとえば，特定の学術誌用レイアウトファイルの中で，定理番号を節毎に振る theorems-sec モジュールが使用されるのを防ぐために用いたりすることができます．このタグは，モジュール内で**使用してはいけません**．モジュールは，他のモジュールを排除する独自の枠組みがあります（第 5.2.1 節参照）.

Float フロートを新規に定義します．詳細は，第 5.3.9 節を参照してください.「End」で閉じる必要があります.

HTMLPreamble この文書クラスが XHTML に出力されるときに，<head>ブロックに出力される情報です．これより前に出現した HTMLPreamble や AddToHTMLPreamble 宣言は，すべて完全に上書きされることに注意してください (プリアンブルに何かを追加したい時には，AddToHTMLPreamble を使用してください)．これは「EndPreamble」で閉じる必要があります.

HTMLTOCSection [<様式>] 文書が HTML に出力されるときに，目次や書誌情報などに使用されるレイアウトです．article の場合には，これは通常「節」であり，book の場合は「章」です．これを指定しない場合には，LyX はどのレイアウトを使用するべきか，解析しようと試みます.

IfCounter [<カウンタ>] 与えられたカウンタの特性を修正します．カウンタが存在しない場合には，この節は無視されます.「End」で閉じる必要があります.
カウンタについての詳細は，第 5.3.12 節をご覧ください.

IfStyle [<様式>] 与えられた段落様式の特性を修正します．様式が存在しない場合には，この節は無視されます.「End」で閉じる必要があります.

Input [<ファイル名>] 名称の指し示すように，このコマンドは，同じコマンドを何度も指定せずに済むように，別のレイアウト定義ファイルを取り込ませます．よく使われる例は，基本的なレイアウトのほとんどを収録している stdclass.inc のような標準レイアウトファイルです.

InputGlobal [<ファイル名>] は Input コマンドの派生ですが，ユーザーディレクトリのファイルは探しません．これによって，ユーザーディレクトリ中に name.layout または name.inc というファイルを作成して，InputGlobal name あるいは InputGlobal name.inc と指定し（ファイル名同順），同名のグローバルファイルを読み込むことができます（Input ではファイルを再帰的に読み込んでしまいます）．このようにすることで，グローバルファイル全体をコピーすることなく，修正できるようになります.

InsetLayout [<型>] このセクションは，差込枠のレイアウトを定義（再定義）します．これは，既存の差込枠にも，新しい文字様式のような新規のユーザ定義差込枠にも使用することができます.「End」で閉じる必要があります.
詳しい情報は，第 5.3.10 節をご覧ください.

LeftMargin [文字列] 画面上の左余白の幅を指示する文字列．例：「MMMMM」.（これは，「2ex」のような「長さ」ではないことに注意してください.）

MaxCiteNames [整数] 著者-刊行年引用で，引用が「第一著者 et al.」に切り替わる前に，表示される名前の最大数を定める整数．主に引用エンジンファイルで使用されます（第 5.2.6 節参照）.

ModifyInsetLayout [<型>] は差込枠のレイアウトを修正します．レイアウトが存在しない場合には，このセクションは無視されます.「End」で閉じる必要があります.

NoCounter [<カウンタ>] このコマンドは，既存のカウンタ (通常インクルードファイル内で定義されたもの) を削除します.

NoFloat [<フロート>] このコマンドは既存のフロートを削除します．これは特に，Input で取り込んだファイルに定義されていたフロートを抑制するのに便利です.

NoStyle [<様式>] このコマンドは既存の様式を削除します．これは特に，Input で取り込んだファイルに定義されていた様式を抑制するのに便利です.

OutlinerName [<型>] [<文字列>] 型が<型>で名前が<文字列>の新しい一覧表を定義します．AddToToc コマンドもご覧ください.

OutputFormat [<形式>] この文書クラスによって生成されるファイル形式（LyX 設定で定義される形のもの）．おもに，OutputType が literate になっていて，新しい型の literate 文書を定義したい時に便利です．対応する OutputType パラメーターに遭遇したときには，形式は「latex」にリセットされます.

OutputType [*latex*, literate] このクラスを使用する文書がどのような種類の出力をするかを示す文字列.

PackageOptions [文字列 文字列] 第1文字列で指定したパッケージ用のオプションを第2文字列で指定します. 例えば,「PackageOptions natbib square」とすると, natbib を square オプションとともに読み込みます (TEXperts 向けに述べると, これは, natbib を読み込む前に, LyX に\PassOptionsToPackage{natbib}{square}を出力させます).

PageSize [*custom*, letter, legal, executive, a0, a1, a2, a3, a4, a5, a6, b0, b1, b2, b3, b4, b5, b6, c0, c1, c2, c3, c4, c5, c6, b0j, b1j, b2j, b3j, b4j, b5j, b6j] 既定のページサイズです. これは一部の変換子に使われます.

PageStyle [*plain*, empty, headings] 既定ページ様式. 文書▷設定ダイアログで変更することができます.

Preamble LaTeX 文書のプリアンブルを設定します. 前に行った Preamble 宣言や AddToPreamble 宣言は, すべて上書きされてしまうので注意してください. (プリアンブルに何かを追加したい時には, AddToPreamble を使用してください.)「EndPreamble」で閉じる必要があります.

ProvideInsetLayout [<型>] 差込枠のレイアウトが存在しない場合に, それを定義します. レイアウトが存在する場合には, このセクションは無視されます.「End」で閉じる必要があります.

Provides [文字列] [*0*, 1] このクラスが文字列で示される機能を既に提供しているかどうかを示します. 機能は, 一般的にパッケージ名 (amsmath・makeidx・...) やマクロ名 (url・boldsymbol・...) です. 機能一覧については, A を参照.

ProvidesModule [文字列] このレイアウトが文字列で表されているモジュールの機能を提供することを示し, .module 拡張子を除いたファイル名で指定します. DefaultModule タグを使用すると, モジュールを使用しなくてはならないことを示しますが, このタグは主に, このレイアウトがモジュールを直接取り込んでしまっていることを示すのに用いられます. 同じ機能を別に実装しているモジュール中で使用するなどすることもできます.

Requires [文字列] このクラスが文字列で表されている機能を要求することを示します. 機能が複数ある場合には, コンマで区切らなくてはなりません. サポートされている機能以外は要求できないことに注意してください (機能一覧については A を参照). 特定のオプションをとるパッケージを要求する場合, PackageOptions を追加して使うことができます.

RightMargin [文字列] 画面上の右余白の幅を指示する文字列. 例:「MMMMM」.

SecNumDepth [int=3] どの節区切りまで連番を振るかを指定します. LaTeX における secnumdepth カウンタに対応します.

Sides [*1*, 2] クラスの既定値として，用紙の片面に印字するか両面に印字するかを指定します．文書▷設定ダイアログで変更することができます．

Style [<名称>] この部分は段落様式を定義します．様式がまだ存在していなければ，生成されます．既に存在していれば，そのパラメータが修正されます．「End」で閉じる必要があります．
段落様式に関する詳細は，5.4.1 をご覧ください．

TableStyle [<名称>] は，表を挿入する際に用いられる既定の表様式を定義します．下記の様式が利用可能です．

- Formal_with_Footline：フォーマル様式（「ブックタブ」様式）．すなわち，水平罫線のみで最上部と最下部が太く，ときに第一行と最終行は表本体とは細い内部罫線で区切られます．
- Formal_without_Footline：上記と同様ですが，最終行は本体から内部罫線で区切られることはありません．
- Simple_Grid：シンプルな表罫線．
- Grid_with_Head：Simple_Grid と同様ですが，ヘッダ行は 2 本目の罫線が余分に引かれます．これは LyX の既定様式でもあります．
- No_Borders：罫線のない表です．

TitleLatexName [文字列="maketitle"] コマンド名あるいは環境名．TitleLatexType とともに使用します．

TitleLatexType [*CommandAfter*, Environment] 文書のタイトルを定義するのに，どのようなマークアップを使用するのかを示します．CommandAfter は，「InTitle 1」が指定されている最後のレイアウトの後に，TitleLatexName で指定したマクロ名を挿入することを意味します．Environment は，「InTitle 1」を持つ段落群を TitleLatexName で指定した環境でくるむ場合に対応します．

TocDepth [int=3] どの節区切りまで目次に取り込むかを指定します．LaTeX の tocdepth カウンタに対応します．

5.3.6. ClassOptions 部

ClassOptions 部は，以下の項目を取り得ます．

FontSize [文字列="10|11|12"] 文書のメインフォントが使用できるフォント寸法の一覧です．「|」で区切ります．任意の数値が使用できます．

FontSizeFormat [文字列] フォント寸法オプションのフォーマット．既定値：$$spt，ここで$$s はフォント寸法のプレイスホルダです．

PageSize [文字列="letter|legal|executive|a0|a1|a2|a3|a4|a5|a6|b0|b1|b2|b3|b4|b5|b6|c0|c1|c2|c3|c4|c5|c6|b0j|b1j|b2j|b3j|b4j|b5j|b6j"] 使用できるページ寸法を「|」で区切ったのリスト．現在，表記の寸法のみサポートされています．他の寸法は，クラスオプション設定で入力することができます．

PageSizeFormat [文字列] ページ寸法オプションの書式．既定値：$$spaper．ここで$$s はページ寸法のプレイスホルダです．

PageStyle [文字列="empty|plain|headings|fancy"] 使用できるページ様式の一覧です．「|」で区切ります．

Other [文字列=""] \documentclass コマンドの非必須パラメータとして付け加える文書クラスオプションです．コンマで区切ります．

ClassOptions 部は「End」で閉じる必要があります．

5.3.7. 段落様式

段落様式の記述は，以下のようになります[13]．

```
Style 名称
 ...
End
```

ここでは，以下のコマンドを使用することができます．

AddToToc [文字列=""] この段落は指定された型の一覧表に現れます．空の文字列を与えると無効になります．OutlinerName コマンドと IsTocCaption コマンドもご覧ください．既定値：無効．

Align [*block*, left, right, center] 段落の揃え．

AlignPossible [*block*, left, right, center] 使用できる揃えのコンマ区切りリスト（LaTeX スタイルには，意味を成さない一部の揃えが禁じられているものがあります．たとえば，連番箇条書きを右揃えや中央揃えにすることはできません）．

Argument [整数] 現在の様式に関連付けられたコマンドまたは環境の引数番号<整数>を定義します．定義は「EndArgument」で閉じる必要があります．詳細については第 5.3.11 節をご覧ください．

AutoNests 現在のレイアウト中もしくは後にネストすべきレイアウトのコンマ区切りリストを入れます．(環境など）ネスト可能なレイアウトでのみ意味を持ちます．「EndAutoNests」で閉じる必要があります．IsAutoNestedBy もご参照ください．

BabelPreamble これは，前に現れたこの様式の BabelPreamble 宣言をすべて，完全に上書きしますので注意してください．「EndBabelPreamble」で閉じる必要があります．これの利用法についての詳細は，第 5.3.8 節をご覧ください．

[13] これは新しいレイアウトを定義するか，既存のレイアウトを修正することになることに注意してください．

BottomSep [浮動小数点型=0][14] このレイアウト型の段落塊の最後の段落と，次の段落とを分離する垂直空白．次の段落が別のレイアウト型である場合，分離幅は足し上げられるのではなく，最大値がとられます．

Category [文字列] この様式のカテゴリです．これは，ツールバーのレイアウト・コンボボックスで関連した様式をグループ化するのに用いられます．任意の文字列を使用することができますが，作成した様式に既存のカテゴリを使用したいと思うことが多いでしょう．

CopyStyle [文字列] 既存の様式から，すべての機能を現在の様式にコピーします．これは，そのときに定義されている様式をコピーすることに注意してください．その後に加えられた変更は，コピーされた様式には影響しません．

DocBookGenerateTitle [ブール型=false] ラッパータグの後に title タグを生成します．このパラメータは，DocBookWrapperTag でのみ使用してください．他の場合はタイトルは環境の中身の後に出力されます．生成されるタイトルは LyXHTML ラベルと同じで，環境型とその番号の組み合わせです．主な使用法は，DocBook に LaTeX 環境を閉じるマッピングがなく，LaTeX にはないタイトルが必要となる figure のような汎用コンテナをユーザが頼らざるを得ないときに使います．この機能は，定理型の環境でたいへんよく用いられます．

DependsOn [<名称>] この**前に**プリアンブルを出力させる様式名．マクロ定義がお互いに依存関係にある場合に，プリアンブルの断片の順序を確実にするためのものです[15]．

EndLabelType [*No_Label*，Box，Filled_Box，Static] 段落の最後（あるいは LatexType が，Environment・Item_Environment・List_Environment のいずれかの場合は，段落群の最後）に置くラベル．No_Label の場合は「何もない」ことを指し，Box（あるいは Filled_Box）の場合は，証明終了マーカ用の白い箱型（あるいは黒い箱型）を指し，Static は明示したテキスト文字列を指します．

EndLabelString [文字列=""] Static 型 EndLabelType のラベルで用いる文字列．

Font 本文テキストとラベルの**両方で**用いられるフォント．第 5.3.13 節を参照．このフォントを定義すると，自動的に LabelFont も同じ値で定義されることに注意してください．したがって，LabelFont も同時に定義したい場合には，これを先に定義してください．

ForceLocal [整数型=0] 新しい様式を LyX 安定版にバックポートするのに用いられます．このタグを最初にサポートした安定版は LyX 2.1.0 です．引数は数字で，0・-1・1 以上の任意の数をとることができます．様式の ForceLocal フラグが 1 以上ならば，これは常に文書ヘッダに書き込まれます．.lyx ファイルが読み込まれると，文書ヘッダからの様式定義が文書クラスに追加されます．したがっ

[14] ここで「浮動小数点型」とは 1.5 のような実数を指します．

[15] この機能以外には，プリアンブルの順序を確定する方法はないことに注意してください．LyX の特定のバージョンで観察された順序は，将来のバージョンで警告なしに変わる可能性があります．

て，古い版の LyX もこの様式を取り扱えるようになります．ForceLocal の引数は版数です．様式が読み込まれたときに，文書クラス中の既存の様式の版数よりも小さいと，新しい様式は無視されます．版数が大きいと，既存の様式に代わって新しい様式が用いられます．値-1 は無限大の版数を意味し，この様式が常に用いられることになります．

FreeSpacing [*0*, 1] LyX は，空白をそれ自体文字や記号ではなく，2 つの単語の間の分割子として捉えているため，単語間に 2 つ以上の空白を入れることは，通常許可していません．これ自体はとても素晴らしいことですが，たとえばプログラムコードや生の LaTeX コードを入力しようとするときなどに，煩わしくなることがあります．このことから，FreeSpacing を有効にすることが認められています．Passthru 1 が指定されていなければ，LyX は 2 つめ以降の空白には非改行空白を生成します．FreeSpacing は KeepEmpty を意味することに注意してください．

HTML* これらのタグは，XHTML 出力で使用されます．第 5.4.1 節をご覧ください．

InPreamble [*0*, 1] 1 の場合，様式が文書本体ではなく，文書プリアンブルにインクルードされるようにします．これは，タイトルや著者の情報をプリアンブルで設定する必要のある文書クラスに便利です．これは，LatexType が Command または Paragraph の様式に対してのみ動作します．

InTitle [*0*, 1] 1 の場合，このレイアウトをタイトルブロックの一部としてマークします（大域項目の TitleLatexType と TitleLatexName も参照）．

IsAutoNestedBy このレイアウトがネストされるべき親レイアウトのコンマ区切りリストを入れます．(環境など）ネスト可能なレイアウトでのみ意味を持ちます．「EndAutoNestedBy」で閉じる必要があります．AutoNests もご参照ください．

IsTocCaption [*0*, 1] これを 1 に設定すると，AddToToc が有効になり，段落は，その内容の要約を一覧表の項目に表示します．0 に設定すると，ラベルが存在すれば，それのみが表示されます．

ItemCommand [文字列="item"] 箇条書きの項目を宣言する LaTeX コマンド．コマンドは，前置されるバックスラッシュを除いた部分で定義されます (既定値は「item」です．これは，LaTeX 出力中では\item となります)．

ItemSep [浮動小数点型=0] これは，同じレイアウトを持つ段落群の間に追加する空白を与えるものです．複数のレイアウトを一つの環境に入れると，それぞれのレイアウトは，その環境の ParSep だけ分離されます．しかし，その環境の項目全体は，さらにこの ItemSep 分だけ離されます．これは**乗数**であることに注意してください．

KeepEmpty [*0*, 1] 段落を空のままにすると，LaTeX 出力が空になってしまうので，通常，LyX は段落を空にすることを許可しません．しかしながら，これを無効にすることが望ましい場合が存在します．たとえば，書簡のひな型では，必須フィールドを人々が忘れないように，空のフィールドのまま提供する手もあります．特

別なクラスにおいては，レイアウトを実際には文章を含まないある種の改行として使用することもあります．

LabelBottomsep [浮動小数点型=0] ラベルと本文テキストとの間の垂直余白．本文テキストの上に来るラベルにのみ使用されます（Top_Environment および Centered_Top_Environment）．

LabelCounter [文字列=""] 自動連番に使われるカウンタ名（詳しくは第 5.3.12 節参照）．カウンタがラベル中に表示されるようにするためには，LabelString 中で参照する必要があります．これは，少なくとも Static・Above・Centered の各 LabelType で動作します．
また，LabelType が Enumerate 型のときにも，若干複雑にはなりますが，本項目を使うことができます．たとえば,「LabelCounter myenum」と宣言したものとしましょう．すると，LaTeX におけるのと同様，実際に使われるカウンタは，myenumi・myenumii・myenumii・myenumiv のようになります．これらのカウンタは，全て別々に宣言されなくてはなりません．
カウンタの詳細については，5.3.12 をご覧ください．

LabelFont ラベルに使用されるフォント．5.3.13 を参照．

LabelIndent [文字列=""] ラベルをどれくらい行頭下げすべきかを示す文字列．

LabelSep [文字列=""] ラベルと本文テキストの間の水平余白の大きさを表す文字列．本文テキストの上に来ないラベルにのみ使用されます．

LabelString [文字列=""] Static ラベル型でラベルに使用する文字列．LabelCounter を設定している場合，5.3.12 に述べられている特別な整形コマンドを含めることができます．

LabelStringAppendix [文字列=""] これは付録の中で LabelString の代わりに用いられます．各 LabelString ステートメントは，LabelStringAppendix をもリセットすることに注意してください．

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

Manual は，ラベルが最初の単語（最初の本当の空白まで）であることを示します．ラベルに 2 単語以上使用したいときは，非改行空白を使用してください．

Static は，ラベルが LabelString で宣言したものであることを示します．これは段落冒頭の「行中」に表示されます．LatexType が Environment のときは，連続する同じ Style の段落中，最初の段落にのみ表示されます．

Above および Centered は，Static の特別な場合です．ラベルは段落の上部に行頭か中央揃えで印字されます．

Sensitive はキャプションラベルの「図」や「表」の特別な場合です．Sensitive は，(ハードコードされた）ラベル文字列がフロートの種類に依存することを示します．これは，フロートに関連付けられたカウンタの値が N であるものとすると，「FloatType N」にハードコードされています．キャプションがフロートの外に挿入されると，LabelString は「意味を成しません!」と表示されます．

Enumerate は，通常の連番ラベルを生成します．数値型を Counter で設定する必要があります．第 5.3.12 節を参照してください．

Itemize は，各階層でさまざまなブリットを生成します．表示されるブリット型は文書 ▷ 設定 ▷ ブリットで設定できます．

Bibliography は LatexType BibEnvironment とともにのみ使用されます．

LangPreamble これは，この様式で既に出現した LangPreamble 宣言をすべて，完全に上書きしますので，注意してください．使用法についての詳細は，5.3.8 をご覧ください．

LatexName [<名称>] 対応する LaTeX の名称です．環境名かコマンド名を指します．

LatexParam [<パラメータ>] 対応する LatexName の非必須パラメータです．このパラメータは，LyX 内部から変更することはできません (変更可能なパラメータには Argument を使用してください)．これは，全ての LaTeX Argument の後にそのままの形で出力されます．

LatexType [*Paragraph*, Command, Environment, Item_Environment, List_Environment, Bib_Environment] レイアウトがどのように LaTeX に変換されるべきかを示します [16].

Paragraph は，何も特別なことは意味しません．

Command は，*LatexName*{...}を意味します．

Environment は，\begin{*LatexName*}...\end{*LatexName*}を意味します．

Item_Environment は Environment と同じですが，\item がこの環境のすべての段落に付けられるところだけが異なっています．

List_Environment は Item_Environment と同じですが，LabelWidthString が環境の引数として渡されるところだけが異なっています．LabelWidthString は，編集 ▷ 段落設定ダイアログで定義することができます．

Bib_Environment は Environment に似ていますが，以下のように，書誌情報環境の begin ステートメントに必須引数 (最長ラベル) を追記します．
\begin{thebibliography}{99}
したがって，これは書誌情報環境にのみ有用です．既定の最長ラベル「99」は，書誌情報項目の段落設定で，ユーザーが変更することができます．

[16] これらのルールは SGML クラスにも適用されるので，LatexType の名称は，少しミスリーディングかもしれません．特定の例については，SGML クラスファイル (ファイル名 db_*.inc) を見てください．

上記最後のいくつかをまとめると，LaTeX 出力は，LaTeX 型に依存して

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

のようになるか，

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

となります．

LeftDelim [文字列] 様式の内容の最初に置かれる文字列．出力中の改行は
で指示できます．

LeftMargin [文字列=""] レイアウトを環境の中に入れた場合，左余白は単純に加えられるのではなく，因子 $\frac{4}{\mathrm{depth}+4}$ をかけて加えられます．このパラメータは，Margin が Manual あるいは Dynamic に設定されているときにも用いられることに注意してください．その場合には，これは手動設定余白または動的設定余白に加えられることに注意してください．
たとえば「MM」と指定すると，段落を通常フォントの「MM」の幅だけ行頭下げを行います．文字列の前に「-」を付けると，負の幅を与えることができます．この方法が採用されたのは，どの画面フォントでも見かけが同じになるようにするためです．

Margin [*Static*, Manual, Dynamic, First_Dynamic, Right_Address_Box]
このレイアウトの左余白の種類です．

Static は固定余白を示します．

Manual は，左余白が編集▷段落設定ダイアログで入力した文字列によって決められることを示します．これは，タブを用いずに整った一覧表を組むのに使用されます．

Dynamic は，余白がラベルの大きさに依存することを示します．これは，自動連番の見出しに使用されます．「5.4.3.2.1 非常に長い見出し」という見出し行が，「3.2 非常に長い見出し」よりも広い左余白（5.4.3.2.1 足す空白と同じ幅）を必要とすることは明らかでしょう（標準的「ワープロ」はこんなことはやってくれませんが）．

First_Dynamic は似ていますが，段落の最初の行だけが Dynamic でその他の行は Static です．これは，たとえば，箇条書き（記述）に使用されます．

Right_Address_Box は，段落中，最も長い行が右余白に合うように余白を選択します．これは，ページの右端に住所を組版するのに用いられます．

NeedProtect [*0*, 1] このレイアウト中の脆弱なコマンドが\protect されるべきか否か（註：これはこのコマンド自体が\protect されるべきかではありません）．

NeedCProtect [*0*, 1] これは必要ならば，このレイアウトを含むマクロを\cprotect（cf. cprotect パッケージ）を用いて保護するようにし，マクロ中で verbatim を使えるようにします．

NeedMBoxProtect [*0*, 1] この様式中の（\cite や\ref のような）特定のコマンドが\mbox 中で保護されるか否か．これは，中身を複雑な方法で解析する ulem や soul コマンドに頼る様式でとくに必要になります．

Newline [0, *1*] 新規行を LaTeX の新規行（\\）に変換するか否か．LyX 中で LaTeX 編集をやりやすくするために，変換は無効にすることができます．

NextNoIndent [*0*, 1] 真に設定すると，DefaultStyle (通常 Standard) 段落が行頭下げになっている場合でも，この型の段落に続くそれらの段落は，字下げされません (逆に言えば，既定でない段落には影響を与えません)．

ObsoletedBy [<名称>] このレイアウトが置き換えられたレイアウト名．これは，後方互換性を維持しながら，レイアウトの名称を変更するのに使用されます．

ParagraphGroup [0, 1] 同じ型の段落が引き続く場合，同じ段落として取り扱うか否かを決定します．これは，そのような連続グループに対して，GuiLabel が一度だけ出力される効果があります．既定では，LaTeXType，Environment および Bib_Environment については有効であり，他の型すべてについて無効になっています．

ParbreakIsNewline [*0*, 1] LaTeX 出力中で，段落を空行ではなく，改行で区切るよう指定します．PassThru 1 と併用すれば，(TeX コードを使用したときのように）テキストエディタをエミュレートすることができます．

ParIndent [文字列=""] 段落の最初の行の行頭下げ．レイアウトによっては Parindent は固定されています．例外には標準レイアウトがあり，標準レイアウトの段落の行頭下げは，NextNoIndent で禁止することができるようになっています．また，環境中の標準レイアウト段落は，当該段落の Parindent ではなく，この環境の Parindent を使用します．たとえば，箇条書き（連番）内の標準段落は，行頭下げされません．

ParSep [浮動小数点型=0] このレイアウトの 2 段落間の垂直余白．

Parskip [浮動小数点型=0] LyX では，文書を組版するのに，ユーザが「行頭下げ」か「スキップ」を選ぶことができます.「行頭下げ」を選択した際には，この値は完全に無視されます.「スキップ」を選択した際には，LaTeX 型「段落」レイアウトの ParIndent は無視され，すべての段落はこの Parskip 引数分だけ引き離されます．垂直余白は，DefaultHeight を標準フォントでの 1 行の高さとすると，Parskip の値 ×DefaultHeight によって計算されます．このようにして，画面フォントを変更しても同じように表示されるのです．

PassThru [*0*, 1] この段落の内容が，LaTeX が必要とするような特別な変換を行わずに，生の形で出力されるべきかどうか．

PassThruChars [文字列] LaTeX が要請する特別な翻訳なしで．生の形で出力されるべき，それぞれの文字を定義します．

Preamble この様式が使用されたときに，LaTeX プリアンブルに付け加えるべき情報．この特定の様式が要求するマクロを定義したり，パッケージを読み込んだりと言ったことに使用します．「EndPreamble」で閉じる必要があります．

RefPrefix [文字列] この型の段落を参照する際，生成されるラベルに使用する前置句．これによって，整形参照を使用することができるようになります．

Requires [文字列] この様式が機能文字列を必要とするかどうか (機能詳細については，第 A 節参照)．特定のオプションをとるパッケージを要求する場合，PackageOptions を汎用テキストクラスパラメータとして追加して使うことができます (5.3.5 を参照)．

ResetArgs [*0*,1] (Argument タグで定義された) この様式の LaTeX 引数をリセットします．これは，様式を CopyStyle でコピーし，その (必須及び非必須) 引数は継承したくない場合に便利です．

ResumeCounter [*0*,1] レイアウトの新しい一群で通常リセットされるカウンタを元に戻します．これは，LabelType が Enumerate の時のみ有効です．

RightDelim [文字列] 様式の内容の最後に置かれる文字列．出力中の改行は
で指示できます．

RightMargin [文字列=""] LeftMargin に同様．

Spacing [*single*, onehalf, double, other <**値**>] これはレイアウト中の既定の行間をどうすべきか定義するものです．引数の single・onehalf・double は，それぞれ乗数 1・1.25・1.667 に対応します．引数 other を指定した場合には，実際の乗数値も引数として指定しなくてはなりません．他のパラメータと違って Spacing は，LaTeX パッケージ setspace パッケージを使用した，限定的な LaTeX コードを生成することを意味することにご注意ください．

Spellcheck [0, 1] この様式の段落をスペルチェックするか否か．既定値は真です．

StepParentCounter [*0*,1] 新しいレイアウト群を開始する際に，このカウンタの親カウンタを進めるか否か．これは，現在のところ，LabelType が Enumerate の時のみ有効です．

TextFont 本文に使うフォント．第 5.3.13 節参照．

TocLevel [整数型=3] 目次中でのこの様式の階層．これは，節見出しの自動連番に使用されます．

ToggleIndent [*default*, always, never] このタグは，この段落の 1 行目の行頭下げが，段落設定ダイアログで切り替えられるかどうかを決めます．default が指定されると，文書設定が「行頭下げ」段落様式を使用しているときに，行頭下げを変更することができます．always は，文書設定にかかわらず変更することができ，never は，どのような場合でも変更することができません．

TopSep [浮動小数点型=0] このレイアウトを持つ一連の段落群の最初の段落と，その前の段落の間の垂直余白．前の段落が別のレイアウトを持っていれば，余白は単純に追加されるのではなく，それらの最大値がとられます．

5.3.8. 段落様式の国際化

LyX は，長きにわたってレイアウト情報の国際化をサポートしてきましたが，第 2.0 版までは，これは操作画面にのみ適用されるものであって，たとえば PDF 出力には適用されませんでした．たとえば，フランスの著者が,「Theorem 1」の代わりに「Théorème 1」としたければ，醜いハックに頼るしかありませんでした．Georg Baum のおかげで，これは解消されました．

もし Style が，組版文書に出力される文字列を定義するのであれば，非英語文書や複数言語文書をサポートするために，LangPreamble や BabelPreamble を使用することができます．以下の抜粋（theorems-ams.inc より）は，これがどう動作するかを示すものです．

```
Preamble

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

原則として，LangPreamble と BabelPreamble タグ内には，有効な LaTeX コードはすべて用いることができますが，実際においては，ここで典型的に示したような形になるでしょう．組版文字列が正しく翻訳されるための鍵となるのは，LaTeX コマンド\claimname とその\newtheorem 中での用い方です．

LangPreamble タグは，文書全体の言語に基づいた国際化を提供します．タグの内容は，Preamble タグと同様，プリアンブルに置かれるのですが，これを特別なものにしているのは,「関数」_() が使用されていることです．これは，LyX が LaTeX 出力を生成する際，その引数を文書言語に翻訳したもので置き換えられます．

BabelPreamble タグは，複数言語文書をサポートし，babel パッケージへのインタフェースを提供することを意図しているので，もう少し複雑です．その内容は，文書に現れる言語それぞれについて一度，プリアンブルに追加されます．この場合には，_() の引数は，その当該言語への翻訳で置き換えられ，$$lang は言語名（babel パッケージで使用されるもの）で置き換えられます．

したがって，フランス語のセクションを持つドイツ語文書では，以下のような内容がプリアンブルに追加されます

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

それから，LaTeX と babel は協力して，出力に正しい文字列を生成します．

ここで注意しておくべき一つの重要な点は，翻訳は，layouttranslations ファイルを通じて，LyX 自身によって提供されるということです．つまり，ユーザー作成のレイアウトファイルに入力された文字列は，layouttranslations ファイルをそれに応じて変更しない限り，LyX の国際化ルーチンでは取り扱われないので，LangPreamble と BabelPreamble は，事実上，LyX とともに提供されるレイアウトファイルでのみ，使うことができるということを意味します．とはいえ，こういうことでありますので，将来的に LyX に同梱させようという意図を以て作成されたレイアウトは，適切なところではすべて，これらのタグを使用するべきです．LyX が提供する段落様式の翻訳は，マイナー更新 (例えば 2.1.x 版から 2.1.y 版) では変更されないことに注意してください．しかしながら，メジャー更新 (例えば 2.0.x 版から 2.1.y 版) では，新しい翻訳や修正が導入される可能性は大いにあります．

5.3.9. フロート

LyX 第 1.3.0 版以来，テキストクラス自体の中でフロート（figure・table・...）を定義することが可能となり，かつ必要となりました．標準的なフロートは stdfloats.inc ファイルに含まれているので，作業中のレイアウトファイルに

```
Input stdfloats.inc
```

と加えるだけで済むことも多いでしょう．LyX に同梱されている AGU クラスのように，それ以外のフロート型を提供するテキストクラスを実装するには，以下の情報が役立つでありましょう．

AllowedPlacement [文字列=!htbpH] このフロート型に許可された配置用の選択肢．値は，配置文字からなる文字列です．使用できる文字には，*h* (“here if possible：可能ならば現在位置に”), *t* (“top of page：ページ上部”)・*b* (“bottom of page：ページ下部”)・*p* (“page of floats：フロートを独立したページに”)・*H* (“here definitely：何としても現在位置に置く”)・*!* (“ignore LaTeX rules：LaTeX の規則を無視する”) があります．文字列中の文字の順序は関係ありません．配置の選択肢を与えない場合には，文字列として *none* を与えてください．

AllowsSideways [0, *1*] フロートを，LaTeX パッケージの rotfloat (横向きフロート) を使って回転させることを許可するか否かを指定．フロートにこの機能をサポートさせないときには，0 を指定してください．

AllowsWide [0, *1*] このフロートに，二段組段落において段落をまたぐ，星付き版があるか否かを定義．フロートにこの機能をサポートさせないときには，0 を指定してください．

Extension [文字列=""] 図などのリストを含む外部ファイルのファイル拡張子名．LaTeX がキャプションを書き込むファイルです．

GuiName [文字列=""] メニューとキャプションに使用される文字列．babel が使用される場合には，これは現在の言語に翻訳されます．

HTML* これらは，XHTML 出力で使用されます．5.4 をご覧ください．

IsPredefined [*0*, 1] フロートがドキュメントクラス中に既に定義されているのか，あるいは LaTeX パッケージ `float` を読み込む必要があり，そのファイル内のものを使用して，オン・ザ・フライで定義するかを示します．既定は `0` で，この場合 `float` を使用します．LaTeX ドキュメントクラス中に既に定義されているときには，`0` に設定しなくてはなりません．

ListCommand [文字列=""] この型のフロートの一覧を生成するのに使用するコマンド．頭部の「\」は書きません．`NeedsFloatPkg` が偽の時には，このコマンドを生成する標準的な方法はないので，これは**必ず**指定しなくてはなりません．`NeedsFloatPkg` が真の時は，標準的な方法が存在するので，これは無視されます．

ListName [文字列=""] この種類のフロート一覧（図一覧・表一覧など）に使用される見出し．LyX 中では，これは画面上のラベルとして使用されます．また，見出しとして使用するために，LaTeX に渡され，XHTML 出力でも見出しとして使用されます．これは，文書言語に翻訳されます．

NumberWithin [文字列=""] この（非必須の）引数は，このクラスのフロートが文書中のある節単位ごとに番号を振り直されるべきかどうかを規定します．例えば，`NumberWithin` が「`chapter`」に指定されていれば，フロートは章ごとに番号が振り直されます．

Placement [文字列=""] このクラスのフロートの既定の配置法．文字列は，標準的な LaTeX 表記に従い，`t` ならば上部（top），`b` ならば下部（bottom），`p` ならばページ（page），`h` ならばここ（here）を表します[17]．これらの他に新しい型 `H` があり，これはフロートを「ここ」に置いていいけれども他の場所はだめ，というものなので，本当はフロートにあるものではありません．しかし，`H` 指定子は特別なものであり，その細かい実装上の理由で，組み込み以外のフロート型では使用することができません．これが何を意味するかおわかりにならない場合には，代わりに「`tbp`」を指定してください．

PrettyFormat [文字列=""] このカウンタへの書式付き参照に使われる書式．たとえば，表への参照を「表 2」のように表示させたいとしましょう．この文字列には，「##」やカウンタ指定を入れることができます（5.3.12 の `LabelString` の説明を参照）．前者はカウンタ番号そのもので置換されます．したがって，節の場合には「第##節」のように指定するか，「第\arabic{section}節」のように指定します（これは第 2.7 節のように翻訳されます）．

[17] LaTeX 同様，文字列中でのこれらの文字の順序は関係ありません．

RefPrefix [文字列] この型のフロートを参照する際，生成されるラベルに使用する前置句．これによって，整形参照を使用することができるようになります．コピーした様式が設定した RefPrefix は，特別な文字列「OFF」（すべて大文字）を使えば，いつでも削除することができます．

Requires [文字列] 段落様式におけるのと同様です．第 5.3.7 節を参照．

Style [文字列=""] \newfloat を使用してフロートを定義する際に使用される様式．

Type [文字列=""] プログラムやアルゴリズムのような，フロートの新しいクラス「型」．適切な\newfloat の後で，\begin{program}や\end{algorithm*}といったコマンドが利用できます．

UsesFloatPkg [0, *1*] このフロートが，クラスファイルやパッケージ中で，LaTeX パッケージ float が提供しているものを使用して定義されているのか，LyX 自身がオン・ザ・フライで定義しているのかを示します．

type 型のフロートを定義すると，自動的に**対応する** *type* 名カウンタが定義されます．

5.3.10. 自由差込枠と差込枠レイアウト

自由差込枠には次の 2 種類があります．

- 文字様式（CharStyle）．これは，\noun や\code などの LaTeX コマンドに対応した意味論的マークアップを定義するものです．
- ユーザ設定（Custom）．これは，TeX コードや脚註などに似たユーザ設定の折りたたみ式差込枠を定義するのに使用することができます．わかりやすい例は endnote 差込枠で，これは endnote モジュール中で定義されています．

自由差込枠は，以下で説明する InsetLayout タグを使用して定義されます．

InsetLayout タグは，もう一つ別の機能も提供します．これを使えば，いろいろな種類の差込枠全体のレイアウトを設定するのに使用することができるのです．現在のところ，InsetLayout は自由差込枠を定義することの他に，脚註・傍註・註釈差込枠・TeX コード（ERT）差込枠・派生枝・リスト・索引・ボックス・表・アルゴリズム・URL・キャプションをユーザー定義するのに使用されます．

InsetLayout 定義は以下の形の行では始まらなくてはなりません．

```
InsetLayout <型>
```

ここで<型>は，レイアウトを定義しようとしている差込枠を指し，4 つの場合があります．

1. 既存の差込枠のレイアウトを変更する場合．この場合，<型>は以下のいずれかになります：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・URL．

2. 自由差込枠のレイアウトを定義する場合．この場合，<型>は Flex:<名称>の形でなくてはなりません．ここで，名称は，既存の自由差込枠で使用されていない有効な識別子であれば，何でも構いません．識別子には空白を入れることもできますが，この場合には全体を引用符で囲まなくてはなりません．自由差込枠の定義には，この定義がどの差込枠型なのか宣言するために，LyXType 項目が含まれていなくてはならないことに注意してください．

3. ユーザー定義派生枝を定義する場合．この場合，<型>は「Branch:<名称>」の形でなくてはなりません．ここで，名称は，ユーザーの文書で定義されている有効な派生枝名です．派生枝名には空白を入れても構いませんが，その場合は派生枝全体を引用符で囲まなくてはなりません．この機能の主な目的は，ユーザーの必要に応じて，特定の派生枝を LaTeX がくるむことができるようにすることです．

4. ユーザー (またはクラス) 定義のキャプションを定義する場合．この場合，<型>は「Caption:<名称>」の形でなくてはなりません．ここで，名称は，メニューに表示されるキャプション名です．応用例については，標準キャプション (Caption:Standard)，若しくは KOMA-Script クラスのクラス定義キャプション (Caption:Above, Caption:Below)，多言語キャプションモジュール (Caption:Bicaption) をご覧ください．

InsetLayout 定義には以下の項目を入れることができます．

AddToToc [文字列=""] この差込枠は，この型の一覧表に表示されます．空の文字列を入れると無効になります．OutlinerName コマンドと IsTocCaption コマンドも参照してください．これは，自由差込枠にのみ実装されています．既定値：無効．

AllowedInInsets この差込枠を挿入することのできる差込枠を，コンマ区切りリストの形で入れます．「EndAllowedInInsets」で閉じる必要があります．挿入先差込枠の特定の引数に挿入を許可したい場合，引数名を@の後に付記してください（例：My_Inset@post:1）．現在のところ，これは直下の差込枠しかサポートしていないことに注意してください（入れ子となっているものは不可）．AllowedInLayouts も参照のこと．

AllowedInLayouts この差込枠を挿入することのできるレイアウトを，コンマ区切りリストの形で入れます．「EndAllowedInLayouts」で閉じる必要があります．挿入先差込枠の特定の引数に挿入を許可したい場合，引数名を@の後に付記してください（例：My_Inset@post:1）．現在のところ，これは直下のレイアウトしかサポートしていないことに注意してください（入れ子となっているものは不可）．AllowedInInsets も参照のこと．

AllowedOccurrences [整数] AllowedInInsets または AllowedInLayouts が定義されているときに，これを使用すると，特定の差込枠や段落（グループ）にこの差込枠を何回挿入することができるかを指定することができます．

AllowedOccurrencesPerItem [0, 1] これが真に設定されていると,(\item を用いる) 箇条書き型環境の中にいるとき，AllowedOccurrences は単一段落ごとに適用されます.

Argument [整数] 現在のレイアウトに関連付けられたコマンドまたは環境の引数番号を定義します．定義は EndArgument で閉じなくてはなりません．詳細は第 5.3.11 節参照.

BabelPreamble 言語コマンドを変更するプリアンブル．第 5.3.8 節参照.

BgColor [<名称>] 差込枠の背景色．使用できる色名一覧は第 B 節参照.

ContentAsLabel [*0*, 1] 差込枠を閉じた際，差込枠の内容をラベルとして使用するか否か．既定値は偽です.

CopyStyle [<型>] 段落様式と同様です．第 5.3.7 節参照のこと．完全な型を指定する必要があることに注意してください．例:CopyStyle Flex:<名称>.

CustomPars [*0*, 1] 段落を設定するのに，段落設定ダイアログをユーザが使えるかどうかを指定します.

Decoration 差込枠の枠とボタンをレンダリングするのに使用する様式を指定するもので，Classic・Minimalistic・Conglomerate のいずれかを指定することができます．脚註は通常 Classic を使用し，TEX コード差込枠は通常 Minimalistic，文字様式は Conglomerate を使用します.

Display [0, *1*] LatexType が Environment の時のみ意味をもちます．環境を，出力中で独立して出力させるか，周囲のテキストとインラインで出力させるかを示します．偽にすると，LATEX 環境は，\begin{LatexName}タグと\end{LatexName}タグの後の空白 (改行文字を含む) を無視するものと仮定します.

EditExternal [*0*, 1] 差込枠の内容を外部で編集できるか否か（文書の出力形式に定義されている編集ソフトウェアを使用）.

End InsetLayout 宣言を閉じるのに必要です.

Font 本文本体とラベル**両方に**使用されるフォントです．第 5.3.13 節を参照．このフォントを定義すると自動的に LabelFont も同じ値に定義されるので，これらを別々の値にしたいときは，これを先に定義してから後に LabelFont を定義しなくてはならないことに注意してください.

FixedWidthPreambleEncoding [*0*, 1] このレイアウトによって生成される BabelPreamble および LangPreamble コードの訳出後の内容が固定幅エンコーディングを持つように強制するか否か．これは，listings のように，utf8 などの可変幅エンコーディングでは機能しない特殊な LATEX パッケージのために必要です．この設定は，XeTEX や LuaTEX のような Unicode を完全に解する LATEX バックエンドを使用している場合には，無視されます.

ForceLocalFontSwitch [*0*, 1] babel使用時に，常にローカルフォント切替（\foreignlanguage）を使い，グローバル切替（\selectlanguage など）は使わない．

ForceLTR [*0*, 1] たとえばTEXコードやURLで「latex」言語が「左から右」（ラテン式）出力になるように強制します．うまく機能しません．

ForceOwnlines [*0*, 1] LATEX出力において，この差込枠が開始する前と終了する後に，改行を強制します．これは，解析上の目的のために，差込枠が単独行に出力されるように保証します．

ForcePlain [*0*, 1] PlainLayout を使用するべきなのか，それともユーザが差込枠で使用されている段落様式を変更できるのかを指定します．既定値は偽です．

FreeSpacing [*0*, 1] 段落様式と同様．第 5.3.7 節参照．

HTML* これらは，XHTML 出力で使用されます．第 5.4 節をご覧ください．

InheritFont [*0*, 1] このパラメータが 1 のとき，この差込枠内のフォントは，スクリーン上のみならず LATEX への書き出しにおいても親から継承されます．0 のときには，文書の既定フォントが使用されます．

InToc [*0*, 1] AddToToc の設定に関わらず,「文書構造」面用に出力される文字列に，この差込枠の内容を含めるかどうか．たとえば，節見出しの脚註の内容が，文書構造の目次に表示されることは望まないでしょうが，通常，文字様式の内容は表示されることを望むでしょう．既定値は偽，すなわち含めません．

IsTocCaption [*0*, 1] AddToToc が有効なときに，これが 1 に設定されていると，差込枠の内容の要約が，目次項目に表示されます．0 の場合は，ラベルのみが表示されます．

KeepEmpty [*0*, 1] 段落様式と同様．第 5.3.7 節参照．

LabelFont ラベルに使用されるフォント．第 5.3.13 節を参照．非効率を回避するため，この定義は Font の前には決して現れてはなりません．

LabelString [文字列=””] ボタンなどに差込枠のラベルとして表示されるもの．差込枠型によっては（TEX コードや派生枝），ラベルが動的に変更されます．

LangPreamble 言語依存のプリアンブル．第 5.3.8 節参照．

LatexName [<名称>] 対応する LATEX 関連物の名称．環境名ないしはコマンド名．

LatexParam [<パラメータ>] 対応する LatexName 関連物の非必須パラメータ．[] のような括弧対を含む．このパラメータは LYX 内部から変更することはできません (変更可能なパラメータには Argument を使用してください)．これは，全ての LATEX Argument の後にそのままの形で出力されます．

LatexType [Command, Environment, None] 様式がどのように LaTeX に変換されるべきかを示します[18].

None は，何も特別なことは意味しません

Command は，*LatexName*{...}を意味します

Environment は，\begin{*LatexName*}...\end{*LatexName*}を意味します

上記最後のいくつかをまとめると，LaTeX 出力は，LaTeX 型に依存して

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

のようになるか，

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

となります.

LeftDelim [文字列] 様式の内容の最初に置かれる文字列．出力中の改行は
で指示できます.

LyxType charstyle・custom・end（charstyle の定義の終わりなどを示すダミー定義）の各値を取ることができます．この項目は，自由差込枠に必須であり，かつ自由差込枠でしか意味を持ちません．この項目は，就中，差込枠がどのメニューに表示されるかを決定します．LyxType を charstyle に設定すると，MultiPar が偽に，ForcePlain が真に設定されます．charstyle 差込枠で MultiPar を真，あるいは ForcePlain を偽に設定したい場合は，LyxType を設定した後に設定します.

MenuString [文字列] メニュー用の文字列．この文字列に特定の文字を「|」で区切って追加することでアクセラレーターを定義することができます（例：My Inset|M）．この指定は必須ではありません．指定がない場合には，型宣言で指定された差込枠名が，代わりにメニューに用いられます.

MultiPar [*0*, 1] この差込枠中に複数の段落を入れることができるかどうか．これは同時に，CustomPars を同じ値に設定し，ForcePlain を逆の値に設定します．これらは，MultiPar の**後に**指定されれば，他の値に指定し直すことができます．既定値は真です.

NeedProtect [*0*, 1] 本レイアウト中で脆弱なコマンドを\protect するか否か（註：当該コマンド自身を protect するかどうかでは**ありません**）．既定値は偽です.

NeedCProtect [*0*, 1] これは必要ならば，このレイアウトを含むマクロを\cprotect（cf. cprotect パッケージ）を用いて保護するようにし，マクロ中で verbatim を使えるようにします．既定値は偽です.

[18] これらのルールは SGML クラスにも適用されるので，LatexType の名称は，少しミスリーディングかもしれません．特定の例については，SGML クラスファイルを見てください.

NeedMBoxProtect [*0*, 1] この様式中の（\cite や\ref のような）特定のコマンドが\mbox 中で保護されるか否か．これは，中身を複雑な方法で解析する ulem や soul コマンドに頼る様式でとくに必要になります．既定値は偽です．

NewlineCmd [文字列] 改行に使用するコマンドを（\\以外に）定義するオプションです．初期値のバックスラッシュは指定できません．

NoInsetLayout [<レイアウト>] 既存の InsetLayout を削除します．

ObsoletedBy [<レイアウト>] この古い InsetLayout を引き継いだ InsetLayout の名称．これは，InsetLayout の改名を，後方互換性を維持しながら行うために用いられます．

ParbreakIgnored [*0*, 1] 1 に設定すると，段落の改行は出力では無視されます．これは，内容が LyX 作業領域内でのみ改行可能で，出力に影響を与えない差込枠において有用です．

ParbreakIsNewline [*0*, 1] 段落様式と同様．第 5.3.7 節参照．

PassThru [*0*, 1] 段落様式と同様．第 5.3.7 節参照．

Preamble 段落様式と同様．第 5.3.7 節参照．

RefPrefix [文字列] この型の差込枠を参照する際，生成されるラベルに使用する前置句．これによって，整形参照を使用することができるようになります．

Requires [文字列] 段落様式と同様．第 5.3.7 節参照．

ResetArgs [*0*, 1] (Argument タグで定義された) この様式の LaTeX 引数をリセットします．これは，様式を CopyStyle でコピーし，その (必須及び非必須) 引数は継承したくない場合に便利です．

ResetsFont [0, *1*] *1* ならば，差込枠がフォント変更指定の中にあるとき，各差込枠の内部で再度フォント変更指定が為されます（たとえば\textbf{周りの文字列\myinset{中身}...}ではなく\textbf{周りの文字列\myinset{\textbf{中身}}...}）.（脚註のような）内部的にフォント設定をリセットするコマンドで意味を持ちます．これを誤って設定してしまうと，望まない結果を生むことがあります（例：\emph{周りの文字列\myinset{\emph{中身 t}}...}では\emph がトグルですので，中身がアップライト体になります）．既定値は 0：フォント変更は差込枠内部で繰り返されません．

RightDelim [文字列] 様式の内容の最後に置かれる文字列．出力中の改行は
で指示できます．

Spellcheck [0, *1*] この差込枠の内容をスペルチェックするか否か．既定値は真です．

5.3.11. 引数

段落様式と差込枠レイアウトは，内容本体に加えて引数を取ることができます．これは，節見出しのようなものに便利で，また LaTeX でのみ意味を持ちます．コマンドまたは環境の引数は全て–段落の内容自身に関連付けられた必須引数を除き，必須・非必須を問わず–，別の場所で定義され，引数番号はその順序を表します．定義は EndArgument で閉じなくてはなりません．非必須引数が 2 つあるコマンドの場合は，以下のようになります．

```
Argument 1
...
EndArgument
Argument 2
...
EndArgument
```

Argument 定義内部では，以下の指定をすることができます．

- LabelString [文字列] この引数を挿入するメニューと引数差込枠ボタンに表示される文字列です (別に MenuString を指定した場合を除く)．メニューには，後ろに「|」で区切った文字を置くことでアクセラレーターを定義することができます (例: “Short Title|S”).

- MenuString [文字列] メニュー単独用の文字列．後ろに「|」で区切った文字を置くことでアクセラレーターを定義することができます (例: “Short Title|S”). この指定は必須ではありません．指定しない場合には，代わりに LabelString がメニューに用いられます．

- Tooltip [文字列] 引数差込枠にマウスをかざしたときに現れるツールチップに表示される長めの説明文．

- Mandatory $[0, 1]$ これが必須の引数なのか (1) 非必須の引数なのか (0) を宣言します．必須引数は，与えられなければ空の出力を行いますが，非必須引数の場合はそもそも出力されません．既定では，必須引数は{...}で区切られ，非必須引数は [...] で区切られます．

- NewlineCmd [文字列] 改行に使用するコマンドを（\\以外に）定義するオプションです．初期値のバックスラッシュは指定できません．

- Requires [整数=0] この引数が出力されるとき，必要となる別の引数を (番号によって) 指定します．例えば，LaTeX コマンドでは，\command[][引数]{文字列}のように，前に (少なくとも空の) 別の非必須引数を要求する非必須引数があります．これは，Argument 2 中に Requires 1 ステートメントを置くことで実現できます．

- LeftDelim [文字列] ({や [ではない) ユーザー定義の左区切りを定義します．出力中の改行は
で指示できます.

- RightDelim [文字列] (}や] ではない) ユーザー定義の右区切りを定義します．出力中の改行は
で指示できます.

- DefaultArg [文字列] ユーザー指定引数が与えられなかったとき，すなわち引数差込枠が挿入されなかったときに限り，挿入される引数を定義します (空の引数差込枠が挿入されても DefaultArg は無効になることに注意してください)．引数が複数のときはコンマで区切る必要があります.

- PresetArg [文字列] どのような場合にも (単独もしくはユーザー定義引数とともに) 挿入される引数を定義します．引数が複数のときはコンマで区切る必要があります.

- Font 引数の内容に用いられるフォント．5.3.13 参照.

- FreeSpacing [*0*, 1] 段落様式と同様．第 5.3.7 節参照.

- LabelFont ラベルに用いられるフォント．5.3.13 参照.

- Decoration [*Classic*, Minimalistic, Conglomerate] 差込枠の枠とボタンに用いられる装飾様式.

- AutoInsert [整数=0] これが 1 に設定されると，各様式が選択されたときに，この引数が自動的に挿入されます.

- InsertOnNewline [整数=0] これが 1 に設定されると，AutoInsert 時にこの変数は新規行に挿入されます（自由差込枠でのみ使用可能）.

- InsertCotext [整数=0] これが 1 に設定されると，この引数は，副文 (選択した文ないしは段落全体) のコピーを内容にとって挿入されます.

- PassThru [*inherited*, true, false] この引数の内容が，LaTeX が要求する特別な書き換えを行うことなく，原文のまま出力されるべきかどうかを指定します．既定値では，PassThru の状態は，引数が属する差込枠または段落レイアウトに継承されます．true または false は，この引数のみの状態を変更します.

- PassThruChars [文字列] LaTeX が要求する特別な翻訳は抜きにして，生の形で出力されるべき各文字を定義します．PassThru とは違って，引数用に，これは明示的に定義されなくてはなりません．つまり，引数は，親差込枠や親レイアウトから PassThruChars を継承しません.

- IsTocCaption [*0*, 1] これを 1 に設定すると，引数は，その内容を対応する一覧表の項目に出力します．AddToToc を参照.

既定では，LatexType が Command のとき，各レイアウトで LyX 作業領域に入力した文字列は，コマンドの最後の (必須) 引数になります．しかしながら，前置句 post: を付けた文字列は，この作業領域引数の後に出力されます．後置引数の番号は 1 から振られ直されますので，作業領域引数の後の最初の引数は post:1 となります．後置引数は，Command 以外の LatexType では無視されます．

(\item[foo] のような) 箇条書き\item の引数は，前置句 item: の後に番号を付けます (例:Argument item:1)．

最後に，前置句 listpreamble: を用いる特別な引数型があります．これは本当は引数ではありませんが，引数インタフェースを用います（したがって前置句後には数値が続きます．例：Argument listpreamble:1）．名前が示すように，Itemize・Enumerate・Description・Bibliography のようなリストが対象です．その中身はリスト開始時，最初の\item の前に独立した行として（LyX からは他の方法ではアクセス不能な場所です）出力されます．このようにして，ユーザーは各リストに（長さ等の）再定義を入れることができます．既定では，これらの引数は区切り文字を持ちません．

5.3.12. カウンタ

テキストクラスには，カウンタ（chapter・figure・...）を定義することが必要です．標準的なカウンタは stdcounters.inc ファイルに含まれているので，作業中のレイアウトファイルに

```
Input stdcounters.inc
```

と加えるだけで済むことも多いでしょう．しかし自製カウンタを定義したければ，そうすることもできます．カウンタ宣言は，

```
Counter カウンタ名
```

で始まらなくてはなりません．ここで「カウンタ名」は，実際のカウンタ名で置き換えます．また，宣言は「End」で終わらなくてはなりません．

以下のパラメータを使用することができます．

InitialValue [整数=1] カウンタの初期値を設定します．リセットの度にカウンタはこの値に戻ります．通常，既定値 1 のままで充分でしょう．

LabelString [文字列=""] 定義されていると，ここで指定した文字列がカウンタの表示の仕方を定義します．この値を指定すると，LabelStringAppendix も同じ値に設定されます．文字列中では，以下の構成要素を使用することができます．

- \thecounter は，カウンタ counter の LabelString（または LabelStringAppendix）を展開したもので置き換えられます．
- カウンタ値は，LaTeX 型マクロ*numbertype*{**カウンタ**}を用いて表現することができます．ここで *numbertype* は以下のいずれかです．arabic：1, 2, 3,...；alph：a, b, c, ...（小文字）；Alph：A, B, C, ...（大文字）；roman：i, ii, iii, ...（小文字ローマ数字）；Roman：I, II, III...（大文字ローマ数字）．

LabelString が定義されていないときは，既定値は以下のように組み立てられます．このカウンタに親カウンタ master（Within で定義）があるときには，文字列 \themaster.\arabic{カウンタ}が使用されます．それ以外の場合は，\arabic{カウンタ}が使用されます．

LabelStringAppendix [文字列=""] LabelString と同様ですが，付録で使用するためのものです．

LaTeXName [文字列=””] LaTeX で使用されるカウンタ名（たとえば LyX では「theorem」というカウンタがありますが，LaTeX には「thm」と出力されます）．

PrettyFormat [文字列=””] このカウンタの整形参照で使用する書式．たとえば，節番号への参照を「Section 2.4」のように表示させたい場合には，文字列に「##」を含めるか，LabelString でのようにカウンタ指定を含めます．前者はカウンタ番号で置換されます．したがって，節の場合には「Section ##」のようにするか,「\S\arabic{section}」のようにします（これは§2.7 のように翻訳されます）．

RefFormat [文字列, 文字列] とくに，単一のカウンタを複数の種類の様式で使用するとき「書式付き参照」で使用するためのものです．たとえば，theorem カウンタは，往々にして Theorem や Lemma など，定理型の環境の全種類で使用されます．第 1 引数は，ラベルで使用される前置詞（たとえば，”thm” や”lem”）を与え，第 2 変数は，LabelString や PrettyFormat に与えるような書式指定文字列を与えます．これが与えられなければ，PrettyFormat が使用されます．

Within [文字列=””] これを別のカウンタ名に設定すると，現在のカウンタは，別のカウンタが増加する毎にリセットされます．たとえば，subsection は section 毎に番号がリセットされます．

5.3.13. フォント指定

フォント指定は，以下のような形を取ります．

```
Font または LabelFont または DefaultFont
  ...
EndFont
```

以下のコマンドを使用することができます．

Color [文字列] 有効な引数については，付録 B をご覧ください．

Family [*Roman*, Sans, Typewriter]

Misc [文字列] 有効な引数は，emph・noun・strikeout・underbar・uuline・uwave・no_emph・no_noun・no_strikeout・no_bar・no_uuline・no_uwave です．それぞれ，対応する属性を有効にしたり無効にしたりします．
たとえば，emph は強調を有効にし，no_emph はそれを無効にします．もし後者

がわかりにくければ，現在のコンテクストのフォント設定は，一般的に周囲のコンテクストから継承していることを思い出してください．ですから no_emph は，たとえば定理環境で，何をせずとも有効となっている強調を無効にするのです．

Series [*Medium*, Bold]

Shape [*Up*, Italic, SmallCaps, Slanted]

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

5.3.14. 引用エンジンの説明

主に引用エンジンファイル（第 5.2.6 節参照）で用いられる CiteEngine ブロックは，特定の「引用エンジン」で提供される引用コマンドを定義します．LyX の用語では，引用エンジンとは，番号や著者名，刊行年を使って，引用を整形する特定の方法のことを指します．LyX は，3 つのエンジンの型をサポートします．すなわち，

1. default: 既定の BibTeX 流の引用方法である，単純な番号による様式（例：「[1]」）
2. authoryear: 著者名と刊行年を使った Harvard 様式の引用（例：「Smith and Miller (2017b)」）
3. numerical: 番号に隣接して著者名やタイトルを付けることのできる拡張された番号引用（例：「Smith and Miller [1]」）

CiteEngine ブロックは以下のようになります．

```
CiteEngine default
 cite
 Citep*[][]
 citeyearpar[][]=parencite*
 ...
End
```

CiteEngine に続くタグがエンジンを表します．各行は，このエンジンによってサポートされる引用コマンドや引用コマンドパラダイムを定義します．行は，LyX コマンドと LaTeX 出力を命名するのに使われる引用コマンドのみを含むこともあれば，色々変えるために複雑なこともあります．完全な文法は

```
LyXName|alias$*<!_stardesc!_stardesctooltip>[][]=latexcmd
```

という形になっています．ここで

- LyXName: *.lyx ファイルで使用される名前.

  可搬性のために，異なる引用パッケージ中の同じ形をしたコマンドには，同じ名前をつけるようにしています（したがって，多くの名前が natbib から派生しており，LaTeX コマンド名が異なる場合には，しばしば latexcmd を変える必要があります）.

- alias: このエンジンにおいて，与えられた LyXName にフォールバックするコマンドの（コンマで区切られた）リスト．これによって，引用パッケージとエンジンを切り替えるのが楽になります．alias はレイアウト定義中での ObsoletedBy に相当するものと考えてよいでしょう.

- latexcmd: 出力される実際の LaTeX コマンド.

Alias と latexcmd は必須ではありません．latexcmd が与えられない時は，LyXName が LaTeX に出力されます.

さらに，下記の点に注意してください.

- 大文字にするとコマンドも大文字化されたものになります（\latexcmd が\Latexcmd に）．これらは通常，名前の前置詞を大文字化するのを確実にします（*von Goethe* ⇒ *Von Goethe*）.

- 括弧 [] は非必須引数の数を表します（0-2 をとりえます）.

- 星印*は星印付きコマンドを示します（\latexcmd が\latexcmd*に）.

  既定では，星印付きバージョンは，MaxCiteNames の閾値を超えたために「et al.」に省略されるべき時にも全ての著者を出力することを意味します.

  星印が，当該コマンドについては別の意味を持つ場合には，<!_stardesc!_stardesctooltip> ように三角括弧で指定することができます．前置詞!_で標識した，翻訳可能なキーワードを最大 2 つ与えることができます．最初のキーワードは，引用ダイアログ中の「Full aut&hor list」チェックボックスラベルを書き換える文字列を指し，二つ目のキーワードは，このチェックボックスに対する非必須のツールチップの文字列を指します.

  これらの 2 つのマクロは，下記のように，前置詞から!を落とした形で CiteFormat（次節参照）で定義されなくてはなりません.

  ```
  _stardesc Sta&rred command label
  _stardesctooltip Tooltip for the starred command checkbox.
  ```

- ドル記号$は，このコマンドが「qualified citation lists」を取り扱うことを示します．これは，リスト中の個別の引用に対して，前置文字列と後置文字列を置くことができる，Biblatex に特有の複数文献引用機能です．詳細については，Biblatex の取扱説明書をご覧ください.

引用エンジンに引用コマンドを追加したい場合（たとえばクラスによって提供される特定のコマンドを追加するなど）には，AddToCiteEngine <engine type> ... End を使うことができます．まだ存在しない引用コマンドのみ追加することができます.

5.3.15. 引用書式指定

（引用ダイアログやツールチップなどの）LyX 内部や XHTML 出力において，書誌情報をどのように表示するべきかの叙述には，CiteFormat ブロックが使用されています．このブロックは，以下のような形をしています．

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

あるいは

```
CiteFormat
 cite ...
 citet*[][] ...
End
```

最初の例の各行は，それぞれ article や book に関連付けられた書誌情報をどのように表示するべきかを定義するものですが，このような定義は，BibTeX ファイル中に存在しうる「項目型」すべてについて与えることができます．特定の定義が与えられなければ，LyX は，ソースコード中に定義されている既定書式を使用します．LyX は，いくつかの書式を stdciteformats.inc ファイルで事前定義しており，これはほとんどの LyX 文書クラスにインクルードされています．

2 つ目の例では，各行は，特定の引用コマンド（この例では\cite 及び\citet）が，引用差込枠ラベルや引用ダイアログ，メニュー，XHTML 出力でどのように表示されるべきかを定義しています．LyX は，LyX に同梱されている個別の*.citeengine ファイル中において，文書 ▷ 設定 ▷ 書誌情報... でサポートしている引用様式用に，そのような書式を定義しています（第 5.2.6 節参照）．

この定義は，BibTeX キーをその値で置換できる機能を持った，簡単な言語を使用しています．キーは，%author%のように%記号でくくらなくてはなりません．したがって，簡単な定義は以下のようになります．

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

これは,「著者名・コンマ・引用に囲まれたタイトル・終止符」を出力します．

もちろん，キーが存在するときのみ，そのキーを出力したい時があるはずです．このようなときには{%volume%[[vol. %volume%]]}のように，条件付きの構成を使用することができます．これは，volume が存在するならば,「vol. 」と volume キーを出力するという意味です．また，
{%author%[[%author%]][[%editor%, ed.]]}
のように，条件の中に else 節を含めることも可能です．ここでは，もし author キーが存在するならば出力され，そうでなければ editor キーと「, ed.」が出力されます．ここでもキーは，%記号でくくられていることに注意してください．条件全体は，波括弧で囲まれています．if 節および else 節は,「[[」と「]]」の二重角括弧で囲まれています．これらすべてのあいだには，空白は入ってはなりません．

これらの条件文に使うことができるものには，項目キーの他に，以下の特殊キーがあります.

- {%dialog%[[真]][[偽]]}: ダイアログとメニューに対しては「真」の部分を処理し，他の文脈（作業領域や書き出し）には「偽」の部分を処理します
- {%export%[[真]][[偽]]}: 書き出しとメニューに対しては「真」の部分を処理し，他の文脈（作業領域やダイアログ）には「偽」の部分を処理します
- {%next%[[真]]}: 他の項目が続く場合には「真」の部分を処理します（複数キーを持つ引用など）
- {%second%[[真]][[偽]]}: これが複数項目の 2 番目の場合には「真」の部分を処理し，それ以外の場合には「偽」の部分を処理します
- {%ifstar%[[真]][[偽]]}: （\cite*など）星印付き引用コマンドの場合には「真」の部分を処理し，星印付きでない場合には「偽」の部分を処理します
- {%ifentrytype:<type>%[[真]][[偽]]}: 現行の項目が<型>に一致する場合には「真」の部分を処理し，それ以外の場合には「偽」の部分を処理します（例：引用定義において{%ifentrytype:book%[[これは book です]][[これは book ではありません]]}）
- {%ifmultiple:<authortype>%[[真]][[偽]]}: 現行の項目が著者型（著者・編者など）が複数の著者を含む場合には「真」の部分を処理し，それ以外の場合には「偽」の部分を処理します（例:書誌情報定義において{%ifmultiple:editor%[[eds.]][[ed.]]
- {%ifqualified%[[真]][[偽]]}: 現行の引用が qualified citation list である（複数文献引用に対する Biblatex 特有の形式）場合には「真」の部分を処理し，それ以外の場合には「偽」の部分を処理します

%author%は書誌情報ファイルに記録されている形のままの著者キーを出力すると述べました．これは「Miller, Peter and Smith, Mary and White, Jane」のような文字列に終わることがあり（BibTEX では著者を区切るのに「and」が用いられるため），望ましい結果ではありません．したがって，LyX は，正しく整形された名前のリスト（これも翻訳の対象となります）を得るための方法を提供しています．以下のキーが提供されています.

1. 書誌情報項目の主要著者・編者に適した，姓名付き名前リスト用．<nametype>部分は要請されているリスト型を表します（例：<nametype:author>）
   - %abbrvnames:<nametype>%: MaxCiteNames に達した場合に短縮される（「et al.」と共に供される）名前リストを提供します.
   - %fullnames:<nametype>%: 完全な名前リストを提供します（「et al.」を用いて短縮されることはありません）.
   - %forceabbrvnames:<nametype>%: MaxCiteNames に関わらず，常に短縮される（「et al.」と共に供される）名前リストを提供します.

2. 書誌情報項目中の姓名の順序が異なっている場合の，姓名付き名前リスト（例：「Miller, John: 何らかの文, in: Mary Smith, ed.: A volume」
   - %abbrvbynames:<nametype>%: MaxCiteNames に達した場合に短縮される（「et al.」と共に供される）名前リストを提供します.
   - %fullbynames:<nametype>%: 完全な名前リストを提供します（「et al.」を用いて短縮されることはありません）.
   - %forceabbrvbynames:<nametype>%: MaxCiteNames に関わらず，常に短縮される（「et al.」と共に供される）名前リストを提供します.
3. 著者-発行年引用ラベルで用いられるような，姓のみを含む姓リスト．これらは<nametype>部分を取りませんが,（著者-発行年ラベルで通例であるように）常に著者リストか，存在しない場合には編者リストを返します.
   - %abbrvciteauthor%: MaxCiteNames に達した場合に短縮される（「et al.」と共に供される）名前リストを提供します.
   - %fullciteauthor%: 完全な名前リストを提供します（「et al.」を用いて短縮されることはありません）.
   - %forceabbrvciteauthor%: MaxCiteNames に関わらず，常に短縮される（「et al.」と共に供される）名前リストを提供します.

前二者の姓名の順序は，下記のマクロで変更することができます.

- !firstnameform %surname%, %prename%（1 の場合の最初の著者）
- !othernameform %surname%, %prename%（1 の場合の他の著者）
- !firstbynameform %prename% %surname%（2 の場合の最初の著者）
- !otherbynameform %prename% %surname%（2 の場合の他の著者）

これによって，名前を「Miller, Peter and Mary Smith: . . . , in: John Doe and Pat Green, eds.:. . . 」のように設定することができます.

もう一つ，定義中で使用することのできる文法として，{!<i>!}という形をしたものがあります．これは,「リッチテキスト」を生成するときに使われる整形情報を定義するものです．当然のことながら，平文を書き出すときには，HTML タグを出力させたくはありませんから，HTML タグは「{!」と「!}」でくるんでやらなくてはならないのです.

CiteFormat ブロックでは，他に 2 つの特殊な定義が可能です．一つめの例としては,

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

といった例が挙げられます．これは，短縮形ないしはマクロであり，%!quotetitle%のように，これがキーであるかのように扱って使用することができます．LyX は，%!quotetitle%を，そこで定義されているものを扱う場合と同じように取り扱います．ですから，明白な警告を敢えてさせて頂くと,

```
!funfun %funfun%
```

のようなことはしないでください．LyX は，無限ループに陥るようなことはありませんが，諦めるまでに時間のかかる長いループに入るかもしれません．

特殊な定義の二つめは，

```
B_pptext pp.
```

のようなものです．これは翻訳対象となるテキスト部分を定義し，これによって書誌情報や引用の対応する部分が翻訳されるようになります．%B_pptext%のように，これをキーとして扱って，定義の中に入れることもできます．以下の 2 つの翻訳パスがあることに注意してください．上記の例のように B_で始まる定義は，すべて現在アクティブなバッファ言語に翻訳されます（したがって翻訳は生成された文書に一致します）．アンダースコアだけから始まる定義は，すべて GUI 言語に翻訳されます．これは

```
_addtobib Add to bibliography only
```

のように，ダイアログやボタンにのみ現れる文字列に適した翻訳です．

これらの翻訳可能な文字列のうちいくつかは，stdciteformats.inc 及び様々な*.citeengine ファイル中に事前定義されています．これは，上記で述べたような意味でのマクロではないことに注意してください．

以下は，これらの機能を全て使った例です．

!authoredit {%author%[[%author%,]][[{%editor%[[%editor%, %B_edtext%,]]}]]}

これは，author キーが定義されているならば，著者とコンマを出力し，author キーが定義されておらず，editor キーが定義されているならば，編集者名の後に B_edtext ないしはその翻訳（既定では「ed.」）を出力します．これは実は stdciteformats.inc の中で定義されていますので，このファイルをまず読みこめば，ご自身の定義ないしは再定義の中で使用することができます．

5.4. XHTML 出力のタグ

LaTeX や DocBook と同様，LyX の XHTML 出力の書式も，レイアウト情報によって制御することができます．一般的に，LyX は適切な既定値を提供し，前述したように，他のレイアウトタグに基づいて，既定の CSS スタイルの構成まで行ないます．たとえば，章見出しを適切に整形するための CSS を書き出すために，LyX は，章様式の Font 宣言で提供されている情報を利用しようと試みます．

したがって，多くの場合，使いたい環境やユーザ設定差込枠などのために満足のいく XHTML 出力を得るために，まったく何もしなくてよいことになるでしょう．しかしながら，これが必要になる場合もあるので，LyX は，生成される XHTML や CSS をカスタマイズするために使用できるレイアウトタグを，たくさん提供しています．

様式宣言や差込枠宣言の外で使用することができるタグに，HTMLPreamble と AddToHTMLPreamble の 2 つがあることに注意してください．これらの詳細については，5.3.5 をご覧ください．

5.4.1. 段落様式

LyX が段落のために出力する XHTML の種類は，通常の段落を取り扱っているのか，コマンドを取り扱っているのか，あるいは環境を取り扱っているのかに依存し，これは対応する LaTeXType タグの内容によって決定されます．

コマンドや通常の段落の場合には，XHTML 出力は以下の形になります．

```
<tag attr="value">
<labeltag attr="value">ラベル</labeltag>
段落の内容
</tag>
```

もちろん，段落にラベルがなければ，ラベルタグは省略することができます．

環境のうち，リストの変種でないものに関しては，XHTML は以下の形を取ります．

```
<tag attr="value">
<itemtag attr="value"><labeltag attr="value">環境ラベル</labeltag>
最初の段落．
</itemtag>
<itemtag>二つめの段落．</itemtag>
</tag>
```

ラベルは，たとえば定理の場合にそうであるように，最初の段落にだけ出力されることに注意してください．

リストに関しては，次のような形になります．

```
<tag attr="value">
<itemtag attr="value"><labeltag attr="value">リストのラベル
</labeltag>最初の項目．</itemtag>
<itemtag attr="value"><labeltag attr="value">リストのラベル
</labeltag>二つめの項目．</itemtag>
</tag>
<tag attr="value">
<labeltag attr="value">リストのラベル
</labeltag><itemtag attr="value">最初の項目．</itemtag>
<labeltag attr="value">リストのラベル
</labeltag><itemtag attr="value">二つめの項目</itemtag>
</tag>
```

ここで labeltag と itemtag の順序が違っていることに注意してください．どちらの順序になるかは，HTMLLabelFirst の設定に依存します．もし HTMLLabelFirst が偽であれば（既定値），最初のケースのようになり，これが真であれば，二番めのケースのように，label が item の外側に来るようになります．

各段落の特定のタグ出力や属性出力は，以下に述べるようなレイアウトタグを使って制御することができます．しかしながら，前述のように，多くの場合，LyX は適切

な既定値を生成するので，たいしたことをしなくても，望ましい XHTML 出力を得ることができるということになるはずです．ここで利用出来るタグは，自分の好みにあわせて微調整する目的でここにあるものと考えてください．

HTMLAttr [文字列]　主幹タグと共に出力される属性情報を指定します．たとえば，「class='mydiv'」のようなものです．既定においては，LyX は「class='レイアウト名'」と出力します．ここでレイアウト名は，レイアウトの LyX 名であり，chapter のように小文字で記述します．

HTMLClass [文字列]　この段落に使用する CSS クラス．段落が連番もしくは記号の箇条書きであるならば，既定値は「lyxenum」または「lyxitem」および階層の深さにより「i」「ii」「iii」「iv」となることに注意してください．これはここで書き換えることができます．しかしながら，その場合後置句は付け加えられません．すなわち，CSS クラスは，つねにここで宣言されたものの通りとなります．

HTMLForceCSS [*0*,1]　HTMLStyle で追加情報が明示的に与えられているときでも，LyX がこのレイアウト用に生成する既定 CSS 情報を出力するか否か．これを 1 にすると，生成された CSS を完全に上書きする代わりに，変更したり追加したりすることができます．既定値は 0 です．

HTMLForceCSS [*0*,1]　この段落（通常節やその類い）を TOC に入れるか否か．既定値は真となっているので，たとえば星付きの節については偽に設定しなくてはなりません．

HTMLItem [文字列]　環境の段落に使用されるタグ．上記各例の itemtag を置き換えます．既定値は div です．

HTMLItemAttr [文字列]　item タグの属性．既定値は class='レイアウト名_item' です．ここには，様式情報は含まれては**なりません**．その目的のためには，HTMLStyle を使用してください．

HTMLLabel [文字列]　段落と項目ラベルに使用されるタグ．上記各例の labeltag を置き換えます．LabelType が Top_Environment や Centered_Top_Environment の時は，既定値は div ですが，それ以外の時の既定値は span です．

HTMLLabelAttr [文字列]　label タグの属性．既定値は class='レイアウト名_label' です．ここには，様式情報は含まれては**なりません**．その目的のためには，HTMLStyle を使用してください．

HTMLLabelFirst [*0*,1]　このタグは，リスト関係環境でのみ意味を持ち，label タグが，item タグの前に出力されるか，中に出力されるかを制御します．これは，たとえば，description 環境の中で，'<dt>...</dt><dd>...</dd>という形を得るために使用されます．既定値は 0 で，label タグは item タグの中に出力されます．

HTMLPreamble この様式が使用されたときに，<head>セクションに出力される情報．これは，たとえば，onclick ハンドラを定義するために<script>ブロックをインクルードするのに使用することができます．

HTMLStyle　この様式が使用されたときに，インクルードする CSS スタイル情報．これは，レイアウトが生成する<style>ブロックで自動的に包まれますので，CSS 自体をインクルードするだけで大丈夫です．EndHTMLStyle で閉じなくてはなりません．

HTMLTag [文字列]　主幹ラベルに使用されるタグ．上記各例の tag を置き換えます．既定値は div です．

HTMLTitle [*0*,1]　この様式が，XHTML ファイルの<title>タグを生成するのに使用する様式であるという印をつけます．既定値は偽です．stdtitle.inc ファイルでは，title 環境のこの項目を真に設定しています．

5.4.2. 差込枠レイアウト XHTML

差込枠の XHTML 出力も，レイアウトファイル内の情報によって制御することができます[19]．ここでも，LyX は適切な既定値を提供しようと試み，既定の CSS 様式を構成します．しかし，すべてカスタマイズ可能です．

LyX が差込枠用に出力する XHTML は，以下の形を取ります．

```
<tag attr="value">
<labeltag>ラベル</labeltag>
<innertag attr="value">差込枠の内容．</innertag>
</tag>
```

差込枠が多段落を許可している—つまり MultiPar が真—ならば，差込枠の内容は，それ自身段落として出力され，それらの段落に用いられる様式（標準，引用など）を用いて整形されます．もちろん，段落にラベルがなければ，label タグは省略され，ラベルがあれば，現在のところ，つねに span が用いられます．inner タグは非必須であり，既定では出力されません．各差込枠用に出力される特定のタグや属性は，以下のレイアウトタグによって制御することができます．

HTMLAttr [文字列]　主幹タグと共に出力される属性情報を指定します．たとえば，「class='myinset' onclick='...'」のようなものです．既定においては，LyX は「class='差込枠名'」と出力します．ここで差込枠名は，差込枠の LyX 名であり，小文字で記述します．アルファベットや数字以外の文字は，アンダースコアに置き換えられます．例：footnote.

HTMLForceCSS [*0*,1]　HTMLStyle で追加情報が明示的に与えられているときでも，LyX がこのレイアウト用に生成する既定 CSS 情報を出力するか否か．これを 1 にすると，生成された CSS を完全に上書きする代わりに，変更したり追加したりすることができます．既定値は 0 です．

HTMLInnerAttr [文字列]　inner タグの属性．既定値は class='差込枠名_inner' です．

[19]現在のところ，これは「テキスト」差込枠（中に書き込みができる差込枠）にのみ有効で，「コマンド」差込枠（ダイアログボックスに関連付けられた差込枠）には適用されません．

HTMLInnerTag [文字列]　inner タグです．上記各例の innertag を置き換えます．既定値はなしです．

HTMLIsBlock [0,*1*] この差込枠が（脚註のように）独立した文字列ブロックを表すのか，それとも,（派生枝のように）周囲の文字列の中に取り込まれる素材を表すのか．既定値は 1 です．

HTMLLabel [文字列]　場合によっては，カウンタへの参照を含む，この差込枠のラベル．たとえば，脚註用には\arabic{footnote}など．これは非必須であり，既定値はありません．

HTMLPreamble この様式が使用されたときに，<head>セクションに出力される情報．これは，たとえば，onclick ハンドラを定義するために<script>ブロックをインクルードするのに使用することができます．

HTMLStyle　この様式が使用されたときに，インクルードする CSS スタイル情報．これは，レイアウトが生成する<style>ブロックで自動的に包まれますので，CSS 自体をインクルードするだけで大丈夫です．

HTMLTag [文字列]　主幹ラベルに使用されるタグ．上記各例の tag を置き換えます．既定値は MultiPar の設定に依存し，MultiPar が真ならば div，偽ならば span です．

5.4.3. フロート XHTML

フロートの XHTML 出力も，レイアウトファイル内の情報によって制御することができます．出力は，以下の形を取ります．

```
<tag attr="value">
フロートの内容.
</tag>
```

キャプションは，存在している場合には，独立した差込枠となり，そのような形で出力されます．その外観は，キャプション差込枠の InsetLayout で制御することができます．

HTMLAttr [文字列]　主幹タグと共に出力される属性情報を指定します．たとえば，「class=‘myfloat’ onclick=‘...’」のようなものです．既定においては，LyX は「class=‘float フロート-フロート型’」と出力します．ここでフロート型は，フロート宣言で定義された（5.3.9 参照），この型のフロートの LyX 名です．ただし，これは小文字に変換され，アルファベットや数字でない文字はアンダースコアに変換されます．例：float-table.

HTMLStyle　このフロートが使用されたときに，インクルードする CSS スタイル情報．これは，レイアウトが生成する<style>ブロックで自動的に包まれますので，CSS 自体をインクルードするだけで大丈夫です．

HTMLTag [文字列]　このフロートに使用されるタグ．上記各例の tag を置き換えます．既定値は div であり，ほとんどの場合変更する必要はありません．

5.4.4. 書誌情報の整形

書誌情報は，CiteFormat ブロックを使用して整形することができます．詳細については，5.3.15 を参照してください．

5.4.5. LyX が生成した CSS

LyX は，提供されている他のレイアウト情報に基づいて，差込枠と段落様式の両方の既定 CSS 様式ルールを生成ということすることを，これまでに何度か触れました．この節では，LyX がどのレイアウト情報を，どのように使うのか，ひとこと述べておきたいと思います．

LyX は，現在のところ，Font 宣言で指定されている Family・Series・Shape・Size を利用して，フォント情報についてのみ CSS を自動生成します（5.3.13 を参照）．この変換は，きわめて分かりやすく自明です．たとえば，「Family Sans」は「font-family: sans-serif」になります．LyX の寸法と CSS の寸法のあいだの対応は，少し複雑ですが，それでも直感的に分かります．詳細については，src/FontInfo.cpp の getSizeCSS() 関数をご覧ください．

5.5. DocBook 出力のタグ

LaTeX や XHTML と同様に，LyX の DocBook 出力の書式もレイアウト情報によって制御されています．一般的に LyX は賢明な既定値を提供していますが，DocBook は厳密にセマンティックであり，フォーマッティングを許さないため，スタイリングの多くは変換の途中で失われてしまいます．可能な場合には，LyX からの情報は role 属性に渡されます．

多くの場合，設定した環境や特別差込枠等々について，納得のいく DocBook 出力を得るために，何かしなくてはならないということはまったくありません．しかしながら，いくつかの場合において必要となることがあるため，LyX は多くのレイアウトタグを提供し，生成される DocBook を調整できるようにしています．

ラベルは DocBook では冗長であるため，めったに出力されることはありません．この情報はタグ自身によって伝達され，（DocBook ファイルを処理した後の）最終文書にラベルが現れるかどうかは，スタイルシートによって制御されます．しかしながら，定義リストのように，ラベルが冗長な内容ではないことが時々あり，この場合には，定義される用語がラベルとなります．

5.5.1. 段落様式

段落に対して LyX が出力する DocBook の類いは，通常の段落を取り扱っているのか，コマンドなのか環境なのかによって変化し，それ自体も対応する LaTeXType タグの内容によって決定されます．

コマンドや通常の段落については，DocBook 出力は下記の形を取ります：

```
<tag attr>
段落の中身
```

```
</tag>
```

箇条書きの類い出ない環境については，生成される DocBook は下記の形を取ります：

```
<tag attr>
<itemtag>第 1 段落</itemtag>
<itemtag>第 2 段落</itemtag>
</tag>
```

箇条書きについては，DocBook 出力は下記の形を取ります：

```
<tag attr>
<itemtag attr>第 1 項目</itemtag>
<itemtag attr>第 2 項目</itemtag>
</tag>
```

各段落型に対する特定のタグやロール出力は，これから説明するレイアウトタグによって制御することができます．まさに DocBook の性質そのものによって，妥当な既定値というものはまったく存在せず，値はつねに注意深く選ばなくてはならないと言うことに注意してください．

DocBookAttr [文字列] メインタグに出力される属性情報を，上記の例の「attr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookTag [文字列] この差込枠に使用するタグ．上記の例の「tag」の位置に出力します．既定値はフロート名で，DocBook には一般的なタグはないので，つねに変更する必要があります．

DocBookTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第 5.5.2 節を参照のこと．

5.5.2. 新規行ポリシー

すべてのタグに関して,（DocBook*TagType 属性で指定される）新規行の出力ポリシーに下記の 3 つの選択肢があります．

- 「block」: 開始タグと終了タグは独立した行に置きます（つまり開始タグと終了タグの前後にラインフィードします）．典型的な要素はフロートです．たとえば，

  前の内容
  <blocktag>
  　ブロックの内容
  </blocktab>
  後の内容

のようになります.

- 「paragraph」: 開始タグと終了タグは同じ新規行に置きます（開始タグの前と終了タグの後にラインフィードが出力されます）. 典型的な要素は段落と箇条書き項目です. たとえば,

  前の内容
  <paratag>段落の内容</paratag>
  後の内容

  のようになります.

- 「inline」: 開始タグと終了タグは内容と同じ行に置かれます. ラインフィードは出力されません. 典型的な要素はフォントです. たとえば,

  前の内容<inlinetag>段落の内容</inlinetag>後の内容

  のようになります.

既定値はつねに「block」です.

5.5.3. InsetLayout DocBook

差込枠の DocBook 出力もレイアウトファイルの情報によって制御されます.
差込枠に対して LyX が出力する DocBook は次のような形を取ります.

```
<wrappertag wrapperattr>
  <tag attr>
    <innertag innerattr>
      差込枠の内容
    </innertag>
  </tag>
</wrappertag>
```

項目立てをする差込枠については, 以下のようになります.

```
<wrappertag wrapperattr>
  <tag attr>
    <innertag innerattr>
      <itemwrappertag itemwrapperattr>
        <itemlabeltag itemattr>
          第 1 項目のラベル
        </itemtag>
        <itemtag itemattr>
          <itemtag itemattr>
            第 1 項目の内容 item.
          </itemtag>
```

```
        </itemtag>
      </itemwrappertag>
      <itemwrappertag itemwrapperattr>
        <itemlabeltag itemattr>
          第２項目のラベル
        </itemtag>
        <itemtag itemattr>
          <itemtag itemattr>
            第２項目の内容
          </itemtag>
        </itemtag>
      </itemwrappertag>
      ...
    </innertag>
  </tag>
</wrappertag>
```

差込枠が複数の段落を許容する場合——つまり MultiPar が真である場合——には，差込枠の中身自身は，段落に使用される様式（標準，引用等）にしたがって整形された段落として出力されます．内側のタグは必須ではなく，既定では表示されません．

各差込枠に出力される特定のタグと属性は，下記のようなレイアウトタグによって制御されます．

DocBookAttr [文字列] メインタグに出力される属性情報を，上記の例の「attr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookInInfo [never，always，maybe] このタグを親レイアウトの最初にある<info>タグの中に入れるかどうかを指定します．never は，このタグを<info>の中に決して入れないことを示します（これは既定値であり通常の内容に対応します）．always は，このタグを<info>の中につねに入れることを示します（これは通常のメタデータに対応します）．親に<info>タグがない場合には生成されます．maybe は，このタグを<info>の中に入れるかもしれないことを示します（これは表題の場合にのみ該当します）．親に<info>タグがない場合には生成されることはありません．対応するタグは，内容として直接出力されます．

DocBookItemAttr [文字列] 項目タグに出力される属性情報を，上記の例の「itemattr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookItemInnerAttr [文字列] 項目内部タグに出力される属性情報を，上記の例の「iteminnerattr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookItemInnerTag [文字列] 差込枠内部の項目内部タグに使用するタグ．上記の例の「iteminnertag」の位置に出力します．既定値は NONE で，項目内部タグ

がないことを示します．すなわち，各項目立て要素について，項目内部タグなしに中身が直接出力されます．このパラメーターは，箇条書きのように，項目立てレイアウトが使われているときのみ意味を持ちます．最もあり得る値は「para」です．
箇条書き項目が新規行を使って分割される場合，項目内部タグは，新規行で分割される段落の各部分に対して個別に出力されます．

DocBookItemInnerTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第 5.5.2 節を参照のこと．

DocBookItemLabelAttr [文字列] 項目ラベルタグに出力される属性情報を，上記の例の「itemlabelattr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookItemLabelTag [文字列] 差込枠内部の項目ラベルタグに使用するタグ．上記の例の「itemlabeltag」の位置に出力します．このパラメーターは，定義リストのように，ラベルの概念を伴う項目立てレイアウトが使われているときのみ意味を持ちます．

DocBookItemLabelTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第 5.5.2 節を参照のこと．

DocBookItemTag [文字列] 差込枠内部の項目タグに使用するタグ．上記の例の「itemtag」の位置に出力します．既定値は NONE で，項目タグがないことを示します．このパラメーターは，箇条書きのように，項目立てレイアウトが使われているときのみ意味を持ちます．

DocBookItemTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第 5.5.2 節を参照のこと．

DocBookItemWrapperAttr [文字列] 項目ラッパータグに出力される属性情報を，上記の例の「itemwrapperattr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookItemWrapperTag [文字列] 差込枠内部の項目ラッパータグに使用するタグ．上記の例の「itemwrappertag」の位置に出力します．既定値は NONE で，項目ラッパータグがないことを示します．すなわち，各項目立て要素について，項目ラッパータグなしにタグと中身が直接出力されます．このパラメーターは，箇条書きのように，項目立てレイアウトが使われているときのみ意味を持ちます．

DocBookItemWrapperTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第 5.5.2 節を参照のこと．

DocBookInnerAttr [文字列] 内部タグに出力される属性情報を，上記の例の「innerattr」の位置に指定します．この情報は，DocBook ファイルの後処理に使用することができます．

DocBookInnerTag [文字列] 差込枠内部の内部タグに使用するタグ．上記の例の「innertag」の位置に出力します．既定値はNONEで，内部タグがないことを示します．すなわち，内部タグなしに中身が直接出力されます．

DocBookInnerTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第5.5.2節を参照のこと．

DocBookSectionTag [文字列] このタイプのセクションに対応するタグを指定します．このパラメーターは，セクショニング要素（部・章・節等）にのみ意味を持ちます．既定値はsectionで，DocBookがセクショニングに他のものを使うときのみ上書きされます（典型的にはbookの部や章）．

DocBookTag [文字列] この差込枠に使用するタグ．上記の例の「tag」の位置に出力します．既定値はフロート名で，DocBookには一般的なタグはないので，つねに変更する必要があります．

DocBookTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第5.5.2節を参照のこと．

DocBookWrapperAttr [文字列] 外部ラッパータグに出力される属性情報を，上記の例の「wrapperattr」の位置に指定します．この情報は，DocBookファイルの後処理に使用することができます．

DocBookWrapperTag [文字列] 差込枠の周りのラッパータグに使用するタグ．上記の例の「wrappertag」の位置に出力します．既定値はNONEで，ラッパータグがないことを示します．すなわち，ラッパータグなしにタグと中身が直接出力されます．

DocBookWrapperTagType [block, paragraph, inline] このタグの新規行ポリシー．詳細は第5.5.2節を参照のこと．

5.5.4. Float DocBook

フロートのDocBook出力もレイアウトファイルの情報によって制御されます．出力は以下の形を取ります．

```
<tag attr>
 DocBookとしてのフロートの内容
</tag>
```

キャプションは，存在するならば別の差込枠となり，表題とともに出力されます．

DocBookAttr [文字列] メインタグに出力される属性情報を，上記の例の「attr」の位置に指定します．この情報は，DocBookファイルの後処理に使用することができます．

DocBookTag [文字列] このフロートに使用するタグ．上記の例の「tag」の位置に出力します．既定値はフロート名で，DocBookには一般的なタグはないので，つねに変更する必要があります．

5.5.5. 書誌情報の組版

読み込んだ書誌情報は整形することができません．すなわち，フィールドはすべて，bibentry タグを使用して（BibTeX ファイルと同等な）データベース型の DocBook 形式でつねに出力されます．

書誌情報項目が，LyX 文書に書誌情報項目として手動で挿入されると，ユーザーはその整形を担当します．すなわち，ユーザーが書いたことを解析することは為されず，文字列は（bibliomixed タグを用いて）直接に使用されます．

6. 外部素材を取り込む

【警告】本説明書のこの部分は，しばらく更新されていません．もちろんまだ正確であることを期待していますが，保証の限りではありません．

LyX 外部のソースから素材を使用する方法は，取扱説明書『**埋込オブジェクト篇**』で詳細にカバーされています．本章は，新種の素材を取り込む際に，舞台裏で何をする必要があるかをカバーします．

6.1. どのように機能するのか

外部素材の機能は，ひな型の概念に基づいています．ひな型は，LyX がある型の素材とどのように橋渡しをするべきかを指定するものです．同梱物として，LyX は，Xfig の図や，様々なラスター形式画像，チェス棋譜，LilyPond 楽譜用のひな型を事前に定義されたものとして含んでいます．実際に何が入っているかは，挿入 ▷ ファイル ▷ 外部素材メニューで見ることができます．さらに，特定の型の素材をサポートするのに，自分自身のひな型を作成することも可能です．後でどのようなことをすればいいか詳細に説明しますが，できればあなたが作ったすべてのひな型を投稿して，我々が LyX の後の版に取り込むことができるようにしてくださることを希望します．

外部素材の機能におけるもう一つの基本的な発想は，最終素材の元となるオリジナルファイルと，書き出された文書や印刷された文書に取り込むための生成ファイルとを区別していることです．たとえば，`Xfig` で作成した図の場合を考えてみましょう．Xfig アプリケーション自体は，`.fig` 拡張子を持つオリジナルファイルを操作します．Xfig で図を作成したり変更したりして，作業が終われば `fig` ファイルに保存します．この図をお使いの文書に取り込みたいときには，LaTeX ファイルにそのままインクルードできるように，`transfig` を呼び出して PostScript ファイルを生成します．この場合には，`.fig` ファイルがオリジナルファイルであり，PostScript ファイルが生成ファイルになります．

この区別は，文書を執筆している最中に，素材を更新することができるようにするために重要です．さらに，これによって，複数の書き出し書式をサポートするために必要な柔軟性が提供されます．たとえば，平文テキストファイルの場合には，図を生の PostScript ファイルとして取り込むのは，とても褒められた発想とはいえません．むしろ，その図への参照だけを含めるか，最終出力が実際の画像に近いものとなるように画像から ASCII への変換子を呼び出したいと考えることでしょう．LyX の外部素材マネジメントは，LyX がサポートする各書き出し書式別に仕分けしているので，ユーザがこれを行うことが可能となっています．

LyX の外部素材マネジメントは，書き出し書式によって異なる生成物をサポートすることの他に，編集・閲覧アプリケーションを緊密に統合することもサポートし

ます．Xfig の図の場合には，LyX の外部素材ダイアログからシングルクリックでオリジナルファイルを Xfig で開くことができ，ダブルクリックすることで生成された PostScript ファイルを Ghostview で閲覧することができます．もうコマンドラインをもてあそんだり，オリジナルファイルや生成ファイルがどこにあるか探したり変更を加えるためにファイルブラウザをいじくり回す必要はないのです．このようにして，文書を執筆する際に必要となる多くのアプリケーションを最大限に利用し，最終的により生産性を上げることができるようになるのです．

6.2. 外用ひな型設定ファイル

LyX に自製の外用ひな型を付け加えるのは，比較的簡単です．しかしながら，これを不用心に行ってしまうと，たいていの場合，簡単に濫用されてしまうようなセキュリティホールを作ってしまいがちであることを心に留めておいてください．したがって，これを実行に移す前に，6.4 のセキュリティに関する議論を読んでおいてください．

このことに言及した上で，あなたが作成した面白いひな型は，ぜひ投稿してください．

外用ひな型ファイルは，LyXDir/lib/xtemplates/ディレクトリ中にある*.xtemplate ファイルに定義されています．各ひな型は，そのファイル完結で定義されています．自分専用のひな型を UserDir/xtemplates/に置くこともできますし，既存のひな型をそのデイレク取りにコピーして修正を加えることもできます．

典型的なひな型は以下のようになります．

```
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
```

ご覧の通り，ひな型は `Template` ... `TemplateEnd` で閉じられます．ひな型には，一般的な設定を行うヘッダ部と，サポートされている主要な文書ファイル形式の設定を行う `Format` ... `FormatEnd` 部があります．

6.2.1. ひな型のヘッダ

`AutomaticProduction true|false` このひな型で扱うファイルをLyXが生成しなくてはならないか否か．このコマンドは，一度だけ必ず現れなくてはなりません．

`FileFilter <パターン>` 望むファイル群を表示するために，ファイルダイアログで使用するフィルタ用globパターン．2つ以上のファイル拡張子があり得る場合（たとえば，tgifには`.obj`と`.tgo`があります），「`*.{obj,tgo}`」の様なパターンを使用してください．このコマンドは，一度だけ必ず現れなくてはなりません．

GuiName <GUI 名> この文字列はボタン上に表示されます．このコマンドは，一度だけ必ず現れなくてはなりません．

HelpText <文章> HelpTextEnd 外部素材ダイアログで使用されるヘルプ文．このひな型がユーザに何を提供できるのか，ユーザに説明するのに十分な情報を盛り込んでください．このコマンドは，一度だけ必ず現れなくてはなりません．

InputFormat <書式> オリジナルファイルのファイル形式．これは，LyX が知っている書式名でなくてはなりません（3.1 参照）．このひな型が，2 つ以上の書式のオリジナルファイルを取り扱える場合は,「*」を使用してください．この場合，LyX はファイル形式を推定するために，ファイル自体に詮索を試みます．このコマンドは，一度だけ必ず現れなくてはなりません．

Template <ID> このひな型の（他と重複しない）名称．代入マクロを含めてはなりません（下記参照）．

Transform Rotate|Resize|Clip|Extra このコマンドは，このひな型がどのような変換をサポートしているかを指定します．これは全く登場しなくても 1 回以上現れても構いません．このコマンドは，外部素材ダイアログ中の対応するタブを使用可能にします．Transform コマンド一つずつに応じて，Format 部に，対応する TransformCommand コマンドか TransformOption コマンドを置かなくてはなりません．これを行わないと，この書式での変換はサポートされません．

6.2.2. Format 部

Format LaTeX|PDFLaTeX|PlainText|DocBook|XHTML この書式定義が定める主要な文書ファイル形式．すべてのひな型が，全文書ファイル形式に対して意味のある表示ができるわけではありません．それでも，全書式に対して Format 部を定義してください．表示する方法がないときは，ダミーテキストを使用してください．これによって，書き出した文書内で，少なくとも外部素材への参照を見ることができるようになります．

Option <名称> <値> このコマンドは，Product での代入に使うマクロ$$<名称>を新たに定義します．<値>自体にも代入マクロを使うことができます．Product で<値>を直接使用するよりも優れた点は，$$<名称>に代入された値が，その文書書式で有効な非必須引数となるように健全化されることです．このコマンドは全く登場しなくても 1 回以上現れても構いません．

Product <文> 書き出された文書に挿入される文．実のところ，これが最も重要なコマンドであり，とても複雑になることがあります．このコマンドは，一度だけ必ず現れなくてはなりません．

Preamble <名称> このコマンドは，LaTeX プリアンブルに入れるプリアンブル片を指定します．これは PreambleDef ... PreambleDefEnd を使用して定義しなくてはなりません．このコマンドは全く登場しなくても 1 回以上現れても構いません．

ReferencedFile <書式> <ファイル名> このコマンドは，変換過程で生成され，特定の書き出し書式に必要とされるファイルを示します．ファイル名が相対パスである場合には，親文書に対する相対パスとして解釈されます．このコマンドは全く登場しなくても 1 回以上現れても構いません．

Requirement <package> 必要とされる LaTeX パッケージ名．パッケージは，LaTeX プリアンブル中で`\usepackage{}`を使って取り込まれます．このコマンドは全く登場しなくても 1 回以上現れても構いません．

TransformCommand Rotate RotationLatexCommand このコマンドは，回転用に，組み込みの LaTeX コマンドを使用するように指定します．このコマンドは，1 回現れても全く現れなくても構いません．

TransformCommand Resize ResizeLatexCommand このコマンドは，伸縮用に，組み込みの LaTeX コマンドを使用するように指定します．このコマンドは，1 回現れても全く現れなくても構いません．

TransformOption Rotate RotationLatexOption このコマンドは，回転が非必須引数を通じて行われるように指定します．このコマンドは，1 回現れても全く現れなくても構いません．

TransformOption Resize ResizeLatexOption このコマンドは，伸縮が非必須引数を通じて行われるように指定します．このコマンドは，1 回現れても全く現れなくても構いません．

TransformOption Clip ClipLatexOption このコマンドは，切り抜きが非必須引数を通じて行われるように指定します．このコマンドは，1 回現れても全く現れなくても構いません．

TransformOption Extra ExtraLatexOption このコマンドは，追加の非必須引数を使用することを指定します．このコマンドは，1 回現れても全く現れなくても構いません．

UpdateFormat <書式> 変換されたファイルのファイル形式．これは，LyX が知っている書式名でなくてはなりません（ツール▷設定▷ファイル処理▷ファイル書式ダイアログを参照）．このコマンドは，一度だけ必ず現れなくてはなりません．得られるファイル形式が PDF の場合，書式 `pdf6` を指定する必要があります．これは画像取り込みに用いられる PDF 計四機です．他の定義済み PDF 形式は文書書き出し用のものです．

UpdateResult <ファイル名> 変換されたファイルのファイル名．ファイル名は絶対パスでなくてはなりません．このコマンドは，一度だけ必ず現れなくてはなりません．

6.2.3. プリアンブルの定義

外用ひな型設定ファイルには，PreambleDef ... PreambleDefEnd で囲んだプリアンブル定義を追加することができます．これらの定義は，ひな型の Format 部で使用することができます．

6.3. 代入機構

外部素材機構が外部プログラムを呼び出すときには，ひな型設定ファイルで定義されたコマンドにしたがって行われます．これらのコマンドには，実行前に展開されるマクロをいろいろ入れることができます．実行は，つねに元の文書があるディレクトリで行われます．

また，外部素材が表示されるときにはいつでも，その名称は代入機構によって組み立てられ，ひな型定義中の他のほとんどのコマンドも代入をサポートしています．

使用できるマクロは以下の通りです．

$$AbsOrRelPathMaster LyX 親文書への絶対ファイルパスないしは相対ファイルパス

$$AbsOrRelPathParent LyX 文書への絶対ファイルパスないしは相対ファイルパス

$$AbsPath 絶対ファイルパス

$$Basename パスおよび拡張子を除いたファイル名

$$Contents("filename.ext") このマクロは，filename.ext と云う名のファイルの中身を展開します．

$$Extension ファイル拡張子（点を含む）

$$pngOrjpg これは，ファイルが JPEG 形式の場合は，文字列「jpg」となり，それ以外では文字列「png」となります．これは，PNG 形式と JPEG 形式の両方をサポートする出力形式に対して，不必要な変換を行うことを避けるのに役立ちます．事前設定されているラスター画像ひな形は，pdfTeX 出力形式に対してこのマクロを使用します．

$$FName 外部素材ダイアログで指定されたファイルのファイル名．これは LyX 文書への絶対パスでも良いですし，相対パスでも構いません．

$$FPath $$FName のパス部分（LyX 文書への絶対パス名か相対パス名）

$$RelPathMaster LyX 親文書への相対ファイルパス

$$RelPathParent LyX 文書への相対ファイルパス

$$Sysdir このマクロは，システムディレクトリの絶対パスを展開します．これは，典型的には，LyX に同梱されているヘルパースクリプト群を示したりするのに使用されます．

$$Tempname 元の文書が閉じられたり，挿入されていた外部素材が削除されたりすると自動的に削除される一時ファイルのフルパスとファイル名.

パスを示すマクロはすべて最後のディレクトリ区切りも含んでいますので，たとえば絶対パスのファイル名を `$$AbsPath$$Basename$$Extension` のようにして作ることができます.

上記マクロは，特記しない限りはすべてのコマンドで代入が行われます．`Transform` コマンドと `TransformCommand` コマンドが有効にされている場合，`Product` コマンドは，これらに加えて以下の代入もサポートします.

$$ResizeFront 伸縮コマンドの前置部.

$$ResizeBack 伸縮コマンドの後置部.

$$RotateFront 回転コマンドの前置部.

$$RotateBack 回転コマンドの後置部.

`Option` コマンドの値に入れる文字列では，`Transform` コマンドと `TransformOption` コマンドが有効にされていれば，以下の代入もサポートされます.

$$Clip 切り抜きオプション.

$$Extra 追加オプション.

$$Resize 伸縮オプション.

$$Rotate 回転オプション.

どうしてこんなに多くのパス関連マクロがあるのか不思議に思われるかもしれません．主に以下の二つの理由があります.

1. 相対ファイル名と絶対ファイル名は，それぞれ相対的あるいは絶対的なままで維持されなくてはなりません．ユーザにはどちらかの形を好む理由があるのかもしれません．たとえば相対名は，いろいろなマシンで作業をする持ち運び用の文書で役立ちます．絶対名は，プログラムによっては必要とされることがあり得ます.

2. LaTeX は，相対ファイル名に関して，LyX や入れ子にした取り込みファイル中の他のプログラムとは異なった取り扱いを行います．LyX にとって相対ファイル名とは，常にこのファイル名が書かれている文書に対して相対的なものになります．LaTeX にとっての相対ファイル名は，常に親文書に対するものになります．これら二つの定義は，一つの文書しかないときには同じですが，部分文書を含む親文書があるときには異なったものとなってきます．つまり，相対ファイル名は，LaTeX に提示されるときに変換されなくてはならないのです．幸い，正しいマクロを選びさえすれば，これは LyX が自動的に行ってくれます.

すると，新しく作ったひな型定義では，どのパス関連マクロを使うべきでしょうか．このルールは難しくありません．つまり，

- 絶対パスが必要とされるときには$$AbsPath を使う.
- 代入された文字列が, LaTeX インプットの一種である場合には, $$AbsOrRelPathMaster を使う.
- それ以外ならば, ユーザの選択を尊重するために$$AbsOrRelPathParent を使う.

このルールが機能せずに, たとえば相対名が必要となる特殊な場合もありますが, 通常, 上記でうまく動作します. 特殊例の例としては, 上述の XFig ひな型での ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pstex_t" というコマンドがあります. この場合, .pstex_t ファイルの複写子は, ファイル内容を書き換えるのに相対名を必要とするために, 絶対名を使用することができないのです.

6.4. セキュリティに関する論点

外部素材機能は, 多くの外部プログラムとの橋渡しをし, しかもそれを自動的に行うので, そのセキュリティ面での帰結を考慮しなくてはなりません. 特に, ユーザは好きなファイル名やパラメータ文字列を含めることが許されていて, しかもそれらがコマンドに展開されるので, ユーザが文書を閲覧したり印刷したりしたときに, 任意のコマンドを実行することができるような悪意ある文書を作成することが可能となりましょう. これは, 我々がぜひとも避けたいことなのです.

しかしながら, 外部プログラムコマンドはひな型設定ファイルでのみ指定されているので, LyX が安全なひな型でのみ適切に設定されているならば, セキュリティ上の問題は発生しません. これは, 外部プログラムが system システムコールではなく, execvp システムコールで呼び出されているため, ファイル名やパラメータ部からシェル経由で任意のコマンドを実行することはできないためです.

これは, 外部素材ひな型でどのようなコマンド文字列を使用することのできるかについて, 制限があることを意味します. 特に, パイプやリダイレクトはそのまま使用することはできません. これは, LyX の安全性を維持するためにそうしなくてはならないのです. もしシェル機能の一部を使用したいとすると, これを完全に統御の下においたまま行う安全なスクリプトを書いた上で, このスクリプトをコマンド文字列から呼び出すようにしなくてはなりません.

シェルと直接やりとりするひな型を設計することは可能ではありますが, 悪意のあるユーザが狡猾なファイル名やパラメータを書くことによって, 任意のコマンドを実行できるようになるため, 一般的には, 統御下に置いた状態で execvp システムコールを使用する安全なスクリプトのみを使用することをお勧めします. 確かに, 管理された環境下で使用する分には, 通常のシェルスクリプトを使用する方に流れる誘惑はあります. そうした場合には, お使いのシステムに簡単に濫用することのできるセキュリティホールを, **間違いなく**導入することを理解しておいてください. オープンソースの伝統に従って, 私たちは人々に新しいひな型を投稿してくれるよう促していますが, そのような安全でないひな型は LyX の標準頒布版には取り入れるべきでないとすることには当然の正当性があります. 公式の頒布チャンネルから出荷されている LyX には, 安全でないひな型は決して入っていません.

外部素材を含めることで強力な力を手に入れることができますが，この力とともにセキュリティ上の危険を導入してしまわないように気をつける必要があります．無防備なスクリプトのたった一行に入り込んだ，ちょっとしたエラーが，巨大なセキュリティ上の問題に扉を開きうるのです．したがって，もしこの問題を完全に理解していないならば，特定のひな型が安全であるかどうか疑問がある際には，知識豊富なセキュリティの専門家か，LyX 開発チームに相談してみてください．そしてこの相談は，管理されていない環境下でこれを使用する前に，行うようにしてください．

A. サポートされているレイアウト用 LyX 関数一覧

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. レイアウトで使用できる色名

ここに列挙されている色は標準色であり，LyX 設定で調整できるものです.

B.1. 色関数

以下は，色そのものではありませんが，色定義に作用します：

ignore この色を無視します

inherit この色を継承します

none 特定の色ではありません – 透明か既定色です

B.2. 静的色名

これらは固定された色名で変更することはできません．これらの色は,（ダークテーマ等）一部の色テーマでうまく動作しないので，レイアウト定義の中では用いないでください：

black

white

blue

brown

cyan

darkgray

gray

green

lightgray

lime

magenta

olive

orange

pink

purple

red

teal

violet

yellow

B.3. 動的色名

これらはの特定要素に割り当てられる色です：

added_space 追加空白色

addedtext 追加文字色

appendix 付録マーカ色

background 背景色

bookmark しおり標識色

bottomarea 下部領域色

branchlabel 派生枝ラベル色

buttonbg ボタンの背景色

buttonframe 差込枠の縁色

buttonhoverbg フォーカスを得ているボタンの背景色

buttonhoverbg_broken フォーカスを得ている破損差込枠ボタンの色

changebar 変更バー色

changedtextauthor1 変更された文章: 第 1 著者の色

changedtextauthor2 変更された文章: 第 2 著者の色

changedtextauthor3 変更された文章: 第 3 著者の色

changedtextauthor4 変更された文章: 第 4 著者の色

changedtextauthor5 変更された文章: 第 5 著者の色

changedtextcomparison 変更された文章: 文書比較（作業領域）

collapsible 畳み込み可能枠の文字色

collapsibleframe 畳み込み可能枠の縁色

command コマンド差込枠の文字色

commandbg コマンド差込枠の背景色

commandframe コマンド差込枠の縁色

command_broken 破損（参照）差込枠の文字色

commandbg_broken 破損差込枠の背景色

commandframe_broken 破損差込枠の縁色

comment コメント色

commentbg コメントの背景色

cursor カーソル色

deletedtext 削除された文章の色

deletedtextmodifier 削除された文章の修飾子の色（輝度調整用）

depthbar 余白部の階層表示線の色

eolmarker 行末標色

error LaTeX エラーボックス色

footlabel 脚註ラベル色

foreground 前景色

graphicsbg 画像差込枠の背景色

greyedoutbg 淡色表示差込枠の背景色

greyedoutlabel 淡色表示差込枠のラベル色

greyedouttext 淡色表示差込枠の文字色

indexlabel 索引差込枠のラベル色

inlinecompletion 行内補完色

insetbg 差込枠標の背景色

insetframe 差込枠標の縁色

language 外国語の単語を標識するための色

latex LaTeX モードの文字色

listingsbg プログラムリスト差込枠の背景色

marginlabel 傍註のラベル色

math 数式差込枠の文字色

mathbg 数式差込枠の背景色

mathcorners フォーカスを得ていない数式差込枠の縁色

mathframe フォーカスを得ている数式差込枠の縁色

mathline 数式行色

mathmacrobg 数式マクロ差込枠の背景色

mathmacroblend 数式マクロ差込枠の混ぜ合わせ色

mathmacroframe 数式マクロ差込枠の縁色

mathmacrohoverbg マウスを置いたときの数式マクロ差込枠の背景色

mathmacrolabel 数式マクロ差込枠のラベル色

mathmacronewarg 数式マクロ新パラメーターのひな型色

mathmacrooldarg 数式マクロ旧パラメーターのひな型色

newpage 新規頁色

nonunique_inlinecompletion 行内補完の一意でない部分の色

note 註釈のラベル色

notebg 註釈の背景色

pagebreak 改頁/改行色

paragraphmarker 段落末を標識するための段落標の色

phantomtext 埋め草差込枠の文字色

preview プレビューに用いられる色

previewframe プレビュー縁色

regexpframe 正規表現枠の色

scroll 行をスクロールすることができることを示す色

selection 文章選択部の背景色

selectiontext 文章選択部の前景色

shadedbg 影付きボックスの背景色

special 特殊文字の色

tabularline 罫線色

tabularonoffline 罫線色

textlabel1 レイアウトと特殊差込枠ラベルの色 1

textlabel2 レイアウトと特殊差込枠ラベルの色 2

textlabel3 レイアウトと特殊差込枠ラベルの色 3

urllabel URL 差込枠のラベル色

urltext URL 差込枠の文字色