Personnaliser LyX : fonctions pour l'utilisateur avancé

par l'équipe LyX*

Version 2.4.x

13 mai 2024

*Merci d'envoyer vos commentaires et corrections à la liste de diffusion de la documentation de LyX, lyx-docs@lists.lyx.org. Insérer « [Customization] » dans l'objet, et mettre en copie le responsable courant de ce fichier, Richard Kimberly Heck <rikiheck@lyx.org>.
Traduction en français par : P.-H. Boinnard, Adrien Rebollo, Siegfried Meunier-Guttin-Cluzel, Jean-Pierre Chrétien. Merci d'envoyer vos commentaires et corrections sur la traduction à la liste de diffusion de LyX en français <lyx-fr@lists.lyx.org>.

Table des matières

1. Introduction

Ce manuel couvre les fonctions de personnalisation de LyX. Nous y abordons des sujets comme les raccourcis clavier, les options d'aperçu à l'écran, les options d'impression, l'envoi de commandes à LyX via le Serveur LyX, l'internationalisation, l'installation de nouvelles classes LaTeX et de nouveaux formats LyX, etc. Nous n'espérons pas aborder tout ce que vous pouvez modifier – nos développeurs ajoutent de nouvelles fonctions plus vite que nous n'en écrivons la documentation – mais nous allons expliquer les personnalisations les plus courantes et au moins vous indiquer la bonne direction pour les plus obscures d'entre elles.

Avertissement : dans le document, on utilise « graphique » en tant que raccourci de « objet graphique » : graphique, diagramme, image, schéma, etc.

2. Les fichiers de configuration de LyX

Ce chapitre devrait vous aider à trouver votre chemin parmi les fichiers de configuration de LyX. Avant d'entreprendre sa lecture, consultez Aide ▷ À Propos de LyX pour connaître le répertoire de bibliothèques et le répertoire utilisateur de LyX. C'est dans celui-ci que LyX place ses fichiers de configuration système ; le répertoire utilisateur est celui dans lequel vous pouvez mettre vos versions modifiées. Le premier sera appelé RépLyX et le second `MonRép` dans la suite de ce document.

2.1. Qu'est-ce qu'il y a dans RépLyX ?

RépLyX et ses sous-répertoires contiennent un certain nombre de fichiers qui peuvent servir à personnaliser le comportement de LyX. Vous pouvez modifier ces fichiers depuis LyX lui-même avec la fenêtre Outils ▷ Préférences. La plupart des réglages personnels que vous voudrez apporter à LyX peuvent se faire par l'intermédiaire de cette fenêtre. Il y a cependant bien d'autres aspects du fonctionnement de LyX sur lesquels on peut agir en modifiant les fichiers de RépLyX. Ils entrent dans différentes catégories, qui sont décrites dans les sous-sections suivantes.

2.1.1. Les fichiers engendrés automatiquement

Ces fichiers sont créés au moment de la configuration de LyX. Ils définissent différentes valeurs implicites qui sont détectées automatiquement par l'inspection de votre système. Comme ils peuvent être réécrits à tout moment, il n'est pas conseillé de les modifier.

`lyxrc.defaults` contient les valeurs implicites pour différentes commandes.

`packages.lst` contient la liste des paquetages LaTeX que LyX a reconnu. LyX ne se sert pas directement de ce fichier, mais les informations obtenues sont disponibles en faisant Aide ▷ Configuration LaTeX.

`textclass.lst` donne la liste des classes de texte présentes dans le répertoire `layouts/`, les classes de document LaTeX correspondantes et leur description.

`lyxmodules.lst` donne la liste des modules figurant dans vos répertoires de format (`layouts`).

`*files.lst` donne la liste de différentes sortes de fichiers connexes à LaTeX figurant dans votre système.

`doc/LaTeXConfig.lyx` est généré automatiquement pendant la configuration de LyX, à partir du fichier `LaTeXConfig.lyx.in`.

2.1.2. Répertoires

Ces répertoires figurent à la fois dans `RépLyx` et dans `MonRép`. Si un fichier existe dans les deux répertoires, c'est celui de `MonRép` qui sera utilisé.

`bind/` contient les fichiers de définition des raccourcis clavier, qui ont le suffixe `.bind`. S'il y a une version « traduite » du fichier de raccourcis, appelée `bind/xx où «xx» est le code de langue ISO`, celle-ci sera détectée et utilisée en premier.

`citeengines/` contient les fichiers avec le suffixe `.citeengine` qui définissent les divers styles de citation (natbib, biblatex etc.). Voir 5.2.6 pour les détails.

`clipart/` contient quelques fichiers graphiques qui peuvent être inclus dans les documents.

`doc/` contient les fichiers de la documentation de LyX (dont celui que vous êtes en train de lire). `LaTeXConfig.lyx` est un cas à part, comme nous venons de le voir. S'il existe des version traduites des fichiers d'aide, avec le préfixe `$LANG`, celles-ci seront utilisées d'abord (voir le 4).

`examples/` contient les fichiers d'exemples qui illustrent la façon d'utiliser certaines fonctions. Vous pouvez y accéder en appuyant sur le bouton Exemples de la fenêtre Fichier ▷ Ouvrir.

`images/` contient les fichiers d'images utilisés dans l'interface Document. Il contient également les icônes utilisées dans la barre d'outils et les bannières affichées au démarrage de LyX.

`kbd/` contient les fichiers de réaffectation clavier. Voir le 4.2 pour des informations détaillées.

`layouts/` contient les classes de texte et les fichiers de modules décrits au 5.

`lyx2lyx` contient les scripts Python `lyx2lyx` permettant d'effectuer les conversions entre versions de LyX. Ils peuvent être exécutés depuis la ligne de commande, si vous désirez par exemple faire un traitement par lots.

`scripts/` contient quelques fichiers qui illustrent les possibilités offertes par l'insertion d'Objet Externe.

`templates/` contient les fichiers modèles de LyX décrits dans la 5.2.4.

`ui/` contient des fichiers avec le suffixe `.ui` qui définissent l'interface utilisateur de LyX. C'est-à-dire que ces fichiers définissent quels éléments apparaissent dans les menus et quels éléments apparaissent dans la barre d'outils.

`xtemplates/` contient les fichiers avec le suffixe `.xtemplate` qui définissent les modèles permettant l'insertion de contenu externe dans un document LyX, voir 6.

2.1.3. Fichiers que vous n'avez pas à modifier

Ce sont des fichiers internes à LyX, et ils ne doivent pas être modifiés, sauf par les développeurs.

`CREDITS` ce fichier contient la liste des développeurs de LyX. Son contenu est affiché en sélectionnant Aide ▷ Crédits.

`chkconfig.ltx` est un script LaTeX utilisé pendant le processus de configuration. Ne pas le lancer directement.

`configure` est le script qui permet de reconfigurer LyX. Il génère des fichiers de configuration dans le répertoire dans lequel il est exécuté.

2.1.4. Autres fichiers appelant un commentaire

`encodings` contient des tables de conversion faisant correspondre les différents encodages de caractères avec Unicode.

`languages` contient la liste de toutes les langues actuellement supportées par LyX.

`latexfonts` contient des informations sur les différentes polices.

`layouttranslations` contient les traductions pour les styles de paragraphe internationalisés (voir 5.3.8).

`unicodesymbols` contient des informations sur les glyphes encodés en Unicode et la manière dont LyX les traite via LaTeX.

2.2. Votre répertoire personnel de configuration

Même si vous utilisez LyX sans être le super-utilisateur, vous pouvez avoir envie de modifier la configuration de LyX pour votre usage personnel. Le répertoire MonRép contient les fichiers personnels de configuration. C'est le répertoire qui est décrit comme « Répertoire utilisateur » dans la fenêtre Aide ▷ À Propos de LyX. Il est traité comme un miroir du répertoire RépLyX, et chaque fichier qui se trouve dans MonRép remplace par conséquent le fichier correspondant de RépLyX. Chaque fichier de configuration décrit dans les sections précédentes peut se trouver soit dans le répertoire de configuration système, ce qui jouera pour tous les utilisateurs, soit dans votre répertoire personnel, pour votre propre configuration.

Pour éclaircir la situation, voici quelques exemples :

- les réglages effectués dans la fenêtre Outils ▷ Préférences sont enregistrés dans un fichier `preferences` dans MonRép ;
- quand vous reconfigurez LyX avec Outils ▷ Reconfigurer, LyX fait tourner le script `configure.py` et les fichiers résultants sont placés dans votre répertoire personnel de configuration. Ça signifie que les nouvelles classes de texte que vous avez pu ajouter dans `MonRép/layouts` seront ajoutées à la liste des classes de la fenêtre Document ▷ Paramètres.
- si vous récupérez sur le site ftp de LyX des fichiers de documentation mis à jour et que vous n'êtes pas administrateur sur votre système, vous pouvez placer ces fichiers dans `MonRép/doc/` et ils seront ouverts directement à partir du menu d'Aide !

2.3. Utiliser LyX avec plusieurs configurations

La liberté offerte pour le répertoire de configuration locale peut être insuffisante si vous avez besoin de plusieurs configurations différentes.

Par exemple, vous pouvez vouloir utiliser des raccourcis clavier ou des réglages d'imprimante différents selon les circonstances[1]. Vous pouvez y arriver en ayant plusieurs répertoires de configuration. Vous spécifiez alors lequel utiliser au moment de lancer LyX.

Si vous lancez LyX avec l'option de ligne de commande `-userdir` <*répertoire*>, le programme va lire la configuration qui se trouve dans ce répertoire, et non dans le répertoire implicite (en lançant LyX sans cette option vous pouvez déterminer le répertoire implicite). Si ce répertoire n'existe pas, LyX vous propose de le créer, exactement comme il le fait avec le répertoire implicite au premier lancement du programme. Vous pouvez modifier les options de configuration dans ce `MonNouveauRép` supplémentaire exactement comme vous le feriez pour le répertoire implicite. Ces répertoires sont complètement indépendants (mais lisez la suite). Notez que positionner la variable d'environnement `LYX_USERDIR_20x` a exactement le même effet.

Quand vous avez plusieurs configurations, vous devez faire plus attention : si vous voulez ajouter un nouveau format dans `MonNouveauRép/layouts`, pour qu'il soit disponible dans toutes vos configurations, vous devez l'ajouter dans chaque répertoire séparément. Vous pouvez contourner ceci avec l'astuce suivante : après que LyX a créé le répertoire supplémentaire, la plupart des sous-répertoires (voir plus haut) sont vides. Si vous voulez que la nouvelle configuration soit le reflet d'une déjà existante, remplacez le sous-répertoire vide par un lien symbolique au sous-répertoire correspondant dans la configuration existante. Faites toutefois attention avec le sous-répertoire `doc/`, car il contient un fichier écrit par le script de configuration (accessible avec Outils ▷ Reconfigurer) qui est propre à chaque configuration.

[1] NdT : ou encore vouloir écrire dans des langues différentes et adapter vos configurations en conséquence.

3. La fenêtre Préférences

Tous les options de la fenêtre Préférences sont expliquées dans l'annexe *La fenêtre Préférences* du *Guide de l'utilisateur*. Pour certaines options, vous trouverez ici plus de précisions.

3.1. Formats

La première étape consiste à définir vos formats de fichiers si ce n'est pas déjà le cas. Pour ce faire, ouvrez Outils ▷ Préférences : dans Gestion des fichiers ▷ Formats de fichier, appuyez sur le bouton Nouveau. Le champ Format contient le nom utilisé pour identifier le format de manière interne. Vous devrez également saisir un suffixe de nom de fichier. Toutes ces informations sont obligatoires. Le champ optionnel Raccourci permet de définir une séquence de touches du clavier pour un accès rapide aux menus (par exemple, appuyer sur Ctrl+D activera Document ▷ Visionner (autres formats) ▷ DVI).

Un Format peut se voir associer une Visionneuse et un Éditeur. Par exemple, vous pouvez vouloir utiliser Ghostview pour afficher les fichier PostScript. Vous pouvez saisir le nom de la commande permettant de lancer les programmes dans les champs correspondants. La visionneuse est lancée quand vous voulez voir un graphique dans LyX ou utiliser le menu Document ▷ Visionner. L'éditeur est lancé par exemple quand vous faites un clic-droit sur un graphique et que vous choisissez Modifier le fichier via une application externe dans le menu contextuel qui apparaît.

Le type MIME d'un format est facultatif, mais s'il est précisé, il doit être unique pour l'ensemble des formats. Il est utilisé pour reconnaître les fichiers de ce format à partir de leur contenu. Pour certains formats de fichier importants, il n'y a pas de type MIME officiellement répertorié dans la base IANA. De ce fait LyX utilise la liste plus étoffée spécifiée par freedesktop.org.

La case à cocher Format de document informe LyX que le format est approprié pour une exportation du document. Si la case est cochée que qu'une procédure de conversion est définie (voir la 3.3), ce format apparaîtra dans le menu Fichier ▷ Exporter. Le format apparaîtra également dans le menu Affichage si une visionneuse est définie pour lui. Les formats purement graphiques, comme png, ne doivent pas utiliser cette

option. Les formats adaptés à la fois aux graphiques et aux documents, comme pdf, doivent l'utiliser.

La case Format graphique vectoriel informe LyX qu'un format peut contenir des vecteurs graphiques. Cette information est utile pour déterminer le format cible des graphiques inclus lors de l'exportation via pdflatex. Les graphiques inclus dans le document peuvent nécessiter une conversion vers pdf, png ou jpg puisque pdflatex ne sait pas gérer d'autres formats graphiques. Si un graphique inclus n'est pas déjà dans un des ces trois formats, il est converti en pdf si la case est cochée, et en png sinon.

3.2. Copieurs

Du fait que toutes les conversions d'un format à une autre se font dans la répertoire temporaire de LyX, il peut être nécessaire de modifier un fichier avant de le copier dan le répertoire temporaire pour que la conversion se fasse correctement[1]. Ceci est effectué par un Copieur : il copie un fichier vers (ou depuis) le répertoire temporaire et peut le modifier à la volée.

La définition des copieurs peut utiliser huit variables :

`$$s`	le répertoire système de LyX (e. g. /usr/share/lyx)
`$$i`	le fichier en entrée
`$$o`	le fichier en sortie
`$$b`	la racine du nom (sans le suffixe) dans le répertoire temporaire LyX
`$$p`	la chemin d'accès complet du répertoire LyX temporaire
`$$r`	le chemin d'accès complet au ficher original LyX en cours de traitement
`$$f`	le nom du fichier LyX (sans chemin d'accès)
`$$l`	le « nom LaTeX »

Ce dernier doit être le nom du fichier tel qu'il serait spécifié dans la commande LaTeX **\include**. Il n'est pertinent que pour l'exportation des fichiers appropriés à une telle inclusion.

[1] Par exemple, le fichier peut référencer d'autres fichiers — e.g. une image — par un nom de fichier relatif, qui peut devenir incorrect lors de la copie du fichier dans le répertoire temporaire.

Les copieurs peuvent être utilisés pour à peu près n'importe quelle opération sur un fichier. Par exemple, supposons que vous vouliez que les fichiers pdf soient copiés dans un répertoire particulier, `/home/you/pdf/`. Alors vous pouvez écrire un script comme celui-ci :

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

Enregistrez-le dans votre répertoire LyX local — disons, `/home/you/.lyx/scripts/pd` — et rendez le exécutable, si c'est nécessaire sur votre plate-forme. Puis, dans Outils ▷ Préférences, sélectionnez sous Gestion des fichiers ▷ Formats de fichier le format PDF(pdflatex) — ou l'un des autres formats PDF — et saisissez `pdfcopier.sh $$i $$o` dans le champ Copieur.

Les copieurs sont utilisés par LyX dans plusieurs de ses convertisseurs internes. Par exemple, si les programmes appropriés sont trouvés lors de la configuration, LyX installera automatiquement des copieurs pour les formats HTML et HTML (MSWord). Quand le documente est exporté vers ces formats, le copieur considère que non seulement le fichier HTML, mais aussi les fichiers auxiliaires (fichiers de style, images, etc.) sont également copiés. Tous ces fichiers sont recopiés dans un sous-répertoire du répertoire dans lequel se trouvait le document LyX original.

3.3. Convertisseurs

Vous pouvez définir vos propres Convertisseurs pour activer les conversions entre différents formats. Ceci est défini dans Outils ▷ Préférences ▷ Gestion des fichiers ▷ Convertisseurs.

Pour définir un nouveau convertisseur, sélectionner les formats appropriés dans les menus déroulants Depuis le format et Vers le format, saisir le nom de la commande nécessaire à la conversion et appuyer sur le bouton Ajouter. Plusieurs variables peuvent être utilisées pour préciser les arguments des commandes de conversion :

`$$s`	le répertoire système de LyX
`$$i`	le fichier en entrée
`$$o`	le fichier en sortie
`$$b`	le nom de base du fichier en entrée (c'est-à-dire sans suffixe)
`$$p`	le chemin du fichier d'entrée

`$$r` le chemin du fichier d'entrée original (ceci peut différer de `$$p` lors de l'appel d'une séquence de convertisseurs)

`$$e` l'identificateur `iconv` désignant l'encodage du document

Dans le champ Autres options, vous pouvez saisir les mots-clés suivants, séparés par des virgules :

`latex=option` ce convertisseur exécute LaTeX ou une variante de LaTeX, le fichier journal sera disponible. `option` précise quelle forme de LaTeX sera exécutée (`latex, pdflatex, platex, xetex, luatex`). Sans `option, latex` est utilisé.

`needaux=option` la conversion utilise le fichier LaTeX .aux. `option` précise quelle forme de LaTeX sera exécutée pour créer le fichier .aux (`latex, pdflatex, platex, xetex, luatex`). Sans `option, latex` est utilisé.

`nice` demande une fichier LyX « nice » (facile), ce qui en pratique désigne un fichier similaire au résultat d'une exportation LaTeX, sans la directive `input@path`.

`xml` la sortie est au format XML

Les quatre mots-clés suivants ne sont pas vraiment des mots-clés, ils prennent un argument de la forme clé=valeur :

`hyperref-driver` est le nom du pilote qu'il faut charger pour ce convertisseur avec le paquetage hyperref, pour ce convertisseur. Le chargement du pilote convenable est nécessaire pour obtenir certaines fonctionnalités spécifiques du PDF. Voir le manuel hyperref pour les détails.

`parselog` s'il est utilisé, les messages d'erreur standard du convertisseur seront redirigés vers un fichier `infile.out`, et le script passé en valeur sera exécuté comme : `script < infile.out > infile.log`. L'argument peut contenir `$$s` ;

`resultdir` précise le nom du répertoire dans lequel le convertisseur placera les fichiers engendrés par la conversion. LyX ne créera pas ce répertoire, et ne copiera rien dedans, bien qu'il le copie dans la cible. La valeur peut utiliser `$$b`, qui sera remplacé par le nom sans suffixe des fichiers d'entrée et de sortie, respectivement, lors de la copie du répertoire. Noter que l'utilisation simultanée de resultdir et usetempdir n'a aucun sens. Ce dernier sera ignoré si le premier est employé ;

`resultfile` détermine le nom du fichier résultat et peut contenir `$$b`. N'a de sens qu'avec resultdir et est optionnel même dans ce cas ; la valeur « index » est implicite s'il n'est pas activé.

Un pilote `hyperref` approprié accompagne certains convertisseurs venant avec LyX. Par contre, aucun des trois derniers mots-clés n'est utilisé dans les convertisseurs pré-installés par LyX.

Il n'est pas nécessaire de définir des convertisseurs pour tous les formats entre lesquels vous désirez effectuer des conversions. Par exemple, vous remarquerez qu'il n'y a pas de convertisseur « LyX vers PostScript », mais LyX exportera bien vers PostScript. Ceci se fait parce que LyX crée d'abord un fichier LaTeX (pas besoin de convertisseur pour cela) qui est ensuite converti vers DVI en utilisant le convertisseur « LyX vers DVI », pour terminer par la conversion « DVI vers PostScript ». LyX trouve ainsi les séquences de convertisseurs automatiquement, et trouvera toujours le séquence la plus courte. Vous pouvez cependant définir plusieurs méthodes de conversion entre formats de fichier. Par exemple, la configuration LyX standard propose cinq façons différentes de convertir LaTeX vers PDF :

1. directement, via pdflatex
2. via (DVI et) PostScript, en utilisant ps2pdf
3. via DVI, en utilisant dvipdfm
4. directement, via XeTeX
5. directement, via LuaTeX

Pour définir de tels choix de séquences, vous devez définir plusieurs « formats de fichier » cible, comme décrit dans la 3.1. Par exemple, avec la configuration standard, les formats dénommés pdf (pour ps2pdf), pdf2 (pour pdflatex), pdf3 (pour dvipdfm), pdf4 (pour XeTeX), et pdf5 (pour LuaTeX) sont définis, qui correspondent tous au suffixe pdf et auxquels sont associées les trois méthodes décrites ci-dessus.

4. Internationaliser LyX

Il est possible de traduire l'interface utilisateur de LyX. La dernière fois que nous avons vérifié, LyX était disponible en 30 langues. La langue que vous avez choisie est appelée votre *locale* (pour plus de renseignements sur les réglages de localisation, voyez la documentation de votre système d'exploitation sur les locales. Pour Linux, la page de man de locale(5) est un bon point de départ).

Notez que ces traductions fonctionnent, mais présentent quelques failles. En particulier, toutes les fenêtres ont été tracées en fonction du texte anglais, et quelques-unes des traductions sont trop grandes pour rentrer dans l'espace alloué. Ce n'est qu'un problème d'affichage sans conséquence. Vous verrez aussi que certaines traductions n'ont pas de raccourcis définis pour tout. Parfois, c'est qu'il n'y a pas assez de lettres disponibles. Parfois c'est que le traducteur n'a tout simplement pas encore eu le temps de le faire. Nos équipes de traduction, que vous pouvez souhaiter rejoindre[1], essaieront de faire disparaître ces inconvénients dans les versions futures de LyX

4.1. Traduire LyX

4.1.1. Traduire l'interface graphique (messages textuels)

LyX utilise la bibliothèque GNU gettext pour gérer l'internationalisation de l'interface. Pour que LyX parle votre langue favorite dans tous les menus et fenêtres, vous avez besoin d'un fichier .po pour cette langue. Quand celui-ci est disponible, vous devez générer à partir de là un fichier .mo et installer ce dernier. Tout ce processus est expliqué dans la documentation de GNU gettext. Vous pouvez effectuer cette traduction pour votre propre besoin, mais si vous avez l'intention de la faire, vous pouvez aussi bien partager le résultat de votre travail avec la communauté LyX. Envoyez un message à la liste de messagerie des développeurs LyX pour plus d'information sur la procédure.

[1] Si vous parlez couramment une autre langue que l'anglais, rejoindre ces équipes est un des grands moyens de rendre service à la communauté des développeurs et utilisateurs de LyX.

En bref, voici un résumé de ce qu'il faut faire (**xx** est le code de la langue) :

- télécharger le code source de LyX (voir la page d'information sur le Web)
- copier lyx.pot dans po, le répertoire des fichiers **.po. Puis renommez-le en **xx.po** (si lyx.pot n'existe pas, il peut être recréé avec la commande **make lyx.pot** dans ce répertoire, ou bien vous pouvez utiliser le fichier po d'une autre langue comme modèle).
- Éditez **xx.po**.[2] Pour certains menus, il y a aussi des touches de raccourci qui doivent être traduites. Ces touches viennent après un «|», et doivent être adaptées au texte traduit. Vous devez aussi remplir les champs au début du nouveau fichier po avec votre adresse e-mail, etc., pour que les gens sachent comment vous joindre pour proposer des suggestions, ou pour démolir votre travail.

Si vous voulez juste traduire pour vous-même, alors :

- engendrer **xx.mo**, avec la commande
  msgfmt -o xx.mo < xx.po
- copier le fichier mo dans votre répertoire de locales, dans le répertoire correspondant aux messages pour la langue **xx**, sous le nom lyx.mo
  (par exemple /usr/local/share/locale/xx/LC_MESSAGES/lyx.mo)

Pour ajouter un nouveau fichier po à la *distribution* de LyX (ce qui serait mieux pour que d'autres puisent en profiter), il faut effectuer quelques changements dans LyX : envoyez donc une courrier à la liste de messagerie des développeurs pour ce faire.

4.1.1.1. Messages ambigus

Il arrive quelquefois qu'un message anglophone doive être traduit de différentes façons dans le langage cible. Un exemple est le message &New: qui se traduit en français par Nouvelle ou Nouvel, suivant le genre et l'initiale du mot suivant. GNU gettext ne sait pas gérer de telles traductions ambiguës, et vous devez donc ajouter une information

[2] C'est un fichier texte: il peut donc être modifié par n'importe quel éditeur de texte. Mais il existe des programmes spécifiques pour gérer les modifications, comme Poedit (toutes plate-formes) ou KBabel (KDE). Emacs est également doté d'un «mode» pour modifier les fichier .po, mais attention, les fichiers sont encodés en Unicode, voir https://www.gnu.org/software/gettext/manual/html_node/PO-Mode.html#PO-Mode.

contextuelle au message : au lieu de `&New:` il devient `&New:[[branch]]` et `&New:[[index]]`. De ce fait, les deux occurrences de `&New:` sont différentes pour `gettext` et peuvent être traduites correctement pour Nouvelle branche : et Nouvel index :, respectivement.

Bien entendu l'information contextuelle doit être supprimée de l'affichage lorsqu'aucune traduction n'est utilisée, c'est pourquoi il faut la mettre entre double crochets à la fin du message comme ci-dessus. Le mécanisme de traduction de LyX assure que tout ce qui est entre double crochets à la fin des messages est supprimé à l'affichage du message.

4.1.2. Traduire la documentation

La documentation en ligne (dans le menu Aide) peut (et doit !) être traduite. S'il existe une traduction de la documentation[3], et si la locale est correctement réglée, LyX affichera tout seul la version traduite. LyX cherche les versions traduites sous le nom RépLyX/doc/**xx**_NomDoc.lyx, où **xx** est le code pour la langue en cours d'utilisation. S'il n'y a pas de traduction disponible, c'est la version anglaise qui est affichée. Notez que les versions traduites doivent avoir le même nom de fichier (ici NomDoc) que l'original. Si vous vous sentez de traduire la documentation (ce qui est d'ailleurs un excellent moyen de corriger la documentation d'origine !), il y a un certain nombre de choses que vous devez faire tout de suite :

- faire un tour sur la page web de la traduction des documentations sur le site de LyX. De cette façon, vous pourrez voir quels documents ont déjà été traduits dans votre langue, s'il y en a. Vous y trouverez aussi qui s'occupe d'organiser l'effort de traduction dans votre langue. Si personne ne s'en occupe, faites-nous connaître votre intérêt.

Une fois que vous vous y mettez pour de bon, voilà quelques conseils qui pourront vous éviter des ennuis :

- joignez-vous à l'équipe de documentation ! `Intro.lyx` (Aide ▷ Introduction) explique comment faire. C'est d'ailleurs le premier document à traduire ;
- étudiez les conventions typographiques de la langue dans laquelle vous traduisez. La typographie est un art ancien et au cours des siècles, une grande variété de conventions ont vu le jour dans les différentes parties du globe. Apprenez aussi le vocabulaire spécifique des professionnels de la typographie dans votre pays. Si vous

[3] En mars 2008, au moins quelques documents avaient été traduit en14 langues, le manuel d'apprentissage étant disponible en quelques autres.

inventez votre propre terminologie, vous allez induire les lecteurs en erreur. (*Attention ! La typographie peut devenir une passion !*) ;

- faites une copie du document (dans les cas simples, sinon voyez la note de bas de page 4). Ce sera votre fichier de travail. Vous pouvez l'utiliser comme fichier d'aide personnel en le mettant dans votre répertoire `MonRép/doc/xx/`.
  Nota : pour un document complexe doté d'éléments externes (images, etc.), si vous voulez faire une copie dans un répertoire temporaire par exemple, faites attention aux liens vers les éléments externes qui peuvent être brisés quand le document est déplacé d'un endroit à un autre. La meilleure méthode est de télécharger l'arborescence LyX depuis le dépôt git (voir `https://www.lyx.org/WebFr.HowToUseGIT`) et de modifier le ficher de documentation en place.
- de temps à autre le document original (de l'équipe LyX) est mis à jour. Utilisez l'interface d'affichage des source pour voir ce qui a changé. De cette façon vous pouvez voir facilement les parties de votre document qui ont besoin d'être mises à jour[4].

Si vous avez trouvé une erreur dans le document original, corrigez-la et dites-le au reste de l'équipe de documentation (que vous n'avez pas manqué de rejoindre, n'est-ce pas ?)

4.2. Les Claviers Internationaux

Les deux sections suivantes décrivent en détail la syntaxe des fichiers `.kmap` et `.cdef`. Ces sections devraient vous aider à concevoir votre propre réaffectation clavier si celles fournies ne vous donnent pas satisfaction.

4.2.1. Le fichier `.kmap`

Un fichier .kmap fait correspondre des appuis clavier à des caractères ou à des chaînes de caractères. Il définit une réaffectation clavier[5]. Cette section décrit les mots clés `kmap`, `kmod`, `kxmod` et `kcomb` des fichiers `.kmap`.

[4] NdT : Je conseille très vivement de partir des fichiers disponibles sur Trac, et de ne pas perdre de temps à traduire la version contenue dans votre distribution pour s'apercevoir trop tard que tout est périmé. J'ajoute que la meilleure solution est de récupérer l'image courante des sources de LyX via git (cf. le site LyX) et de modifier directement le fichier concerné dans l'arborescence importée : de cette façon, les liens vers les images ne seront pas modifiés lors de la sauvegarde du fichier, et la version modifiée pourra être soumise sans modifications aux développeurs.

[5] NdT : Keyboard MAPping.

`kmap` fait correspondre un caractère à une chaîne

`\kmap` *caractère chaîne*

Ceci fait correspondre *chaîne* à *caractère*. Notez que dans *chaîne*, le guillemet double (") et l'antislash (\) doivent être protégés par un antislash (\) juste avant-++.

Voici un exemple de commande kmap qui fait s'afficher le symbole / quand on tape la touche & :

```
\kmap & /
```

`kmod` définit un caractère accentué

`\kmod` *caractère accent autorisés*

Le *caractère* devient alors un *accent* sur les caractères *autorisés*. C'est le principe de la touche morte[6].

Si vous tapez *caractère* puis une autre touche qui ne fait pas partie des caractères *autorisés*, vous obtiendrez un *caractère* suivi par l'autre touche non autorisée. Notez que Ret.Arr efface une touche morte, si vous tapez *caractère* Ret.Arr, le curseur ne recule pas mais annule l'effet que *caractère* aurait pu avoir sur la touche suivante.

Dans l'exemple suivant, il est spécifié que le caractère ' doit être un accent aigu, autorisé sur les caractères a, e, i, o, u, A, E, I, O et U :

```
\kmod ' acute aeiouAEIOU
```

`kxmod` définit une exception au caractère accentué

`\kxmod` *accent caractère résultat*

Ceci définit une exception pour l'*accent* sur le *caractère*. L'*accent* doit avoir été assigné auparavant par une déclaration `\kmod` et le *caractère* ne doit pas faire partie des caractères *autorisés* de l'*accent*. Quand vous tapez la séquence *accent caractère*, se produit le *résultat*. Si une telle déclaration n'existe pas dans le fichier `.kmap` et que vous tapez *accent caractère*, vous obtenez *touche_accent caractère* où *touche_accent* est le premier paramètre de la déclaration `\kmod`.

Avec la commande suivante, vous obtenez äi quand vous tapez acute-i ('i) :

[6] Le terme *touche morte* désigne une touche qui seule ne produit pas de caractère, mais qui produit le caractère accentué voulu quand elle est suivie par une autre touche autorisée. Par exemple, pour écrire un «ê» sur un clavier français, il faut d'abord taper sur «^» puis sur «e».

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

`kcomb` Combine deux caractères accentués

`\kcomb` *accent1 accent2 autorisés*

Celui-ci est assez ésotérique. Il vous permet de combiner les effets de l'*accent1* et de l'*accent2* (dans cet ordre !) sur les caractères *autorisés*. Les touches pour l'*accent1* et l'*accent2* doivent avoir été définies par une commande `\kmod` *plus haut* dans le fichier.

Voyez cet exemple extrait du fichier `greek.kmap` :

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

Ça vous permet de taper `;:i` et d'avoir l'effet de `\'{\"{i}}`. Dans ce cas un Ret.Arr annule la dernière touche morte, donc si vous tapez `;:Backspace i` vous obtenez `\'{i}`.

4.2.2. Le fichier `.cdef`

Après que la réaffectation .kmap a été effectuée, un fichier `.cdef` convertit les chaînes de symboles obtenues dans la police de caractères. Actuellement, la distribution de LyX comprend au moins les fichiers `iso8859-1.cdef` et `iso8859-2.cdef`.

En général le fichier `.cdef` est une suite de déclarations de la forme :

numéro_du_caractère chaîne

Par exemple, pour affecter `\'{e}` au caractère correspondant dans le jeu iso-8859-1 (233), il y a la déclaration suivante :

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

avec `\` et `"` protégés dans *chaîne*. Notez que le même caractère peut servir pour plus d'une chaîne. Dans le fichier `iso-8859-7.cdef` vous avez

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

Si LyX ne trouve pas de correspondance pour la chaîne produite par une touche ou une séquence avec des touches mortes, il va vérifier si elle ressemble à un caractère accentué et va essayer à l'écran de tracer un accent par dessus le caractère.

4.2.3. Touches mortes

Il y a une autre façon d'ajouter le support pour des caractères internationaux par l'intermédiaire des touches mortes. Une touche morte marche en combinaison avec une lettre pour produire un caractère accentué. Ici, nous allons expliquer comment créer une touche morte vraiment simple pour montrer comment elles fonctionnent.

Supposez que vous ayez besoin du caractère circonflexe,[7] «^». Vous devez faire correspondre la touche circonflexe à la commande LyX `accent-circumflex` dans votre fichier `lyxrc`. Maintenant, à chaque fois que vous taperez la touche circonflexe suivie par une lettre, il y aura un accent circonflexe dessus. Par exemple, la séquence «`^e`» produit la lettre «ê». Cependant, si vous essayez de taper «`^t`», LyX va se plaindre avec un bip, car un «`t`» ne prend jamais d'accent circonflexe. Taper Espace après une touche morte fait s'afficher l'accent seul. Notez bien ce dernier point ! Si vous faites correspondre une touche à une touche morte, vous devez faire correspondre le caractère sur cette touche à une touche différente. Faire correspondre la virgule à une cédille est une mauvaise idée, car vous n'aurez plus que des cédilles à la place des virgules.

Une façon courante de créer des touches mortes est d'utiliser Meta-, Ctrl- ou Maj- en combinaison avec un accent, comme «~» ou «^». Une autre méthode met en jeu `xmodmap` et `xkeycaps` (vus dans le *Guide de l'Utilisateur*) pour configurer la touche spéciale `Mode_Switch`. Elle agit un peu comme Maj et permet de faire correspondre des touches à des caractères accentués. Vous pouvez aussi transformer des touches en touches mortes en les affectant à quelque chose comme `usldead_cedilla` puis en affectant cette touche symbolique à la commande LyX correspondante.[8] Vous pouvez transformer à peu près n'importe quelle touche en touche `Mode_Switch` : une des touches `Ctrl-`, une touche de fonction inutilisée, etc. Quant aux commandes LyX qui produisent des accents, voyez à l'entrée `accent-acute` dans le *Manuel de Référence*. Là-bas il y a la liste complète.

4.2.4. Enregistrer votre configuration linguistique

Vous pouvez éditer vos préférences dans la fenêtre Outils ▷ Préférences pour que LyX démarre avec l'environnement linguistique que vous souhaitez, automatiquement configuré.

[7] NdT : déjà présent en français, mais nous allons conserver cet exemple.

[8] Note de John Weiss : C'est exactement ce que je fais dans mes fichiers `~/.lyx/lyxrc` et `~/.xmodmap`. Ma touche Arrêt Défil (ou Scroll Lock) est configurée comme `Mode_Switch` et j'ai affecté des choses comme Arrêt Défil-^ ou Arrêt Défil-~ à un tas de touches symboliques «`usldead_*`». C'est comme ça que j'obtiens tous mes caractères accentués.

5. Créer et installer de nouvelles classes de document, formats et modèles

Dans ce chapitre, nous décrivons comment créer et installer de nouveaux fichiers de format LyX ou de modèle, et nous vous offrons une révision des procédures correctes d'installation de nouvelles classes de document LaTeX.

D'abord, quelques mots pour décrire une bonne approche de la relation entre LyX et LaTeX. Ce qu'il faut bien comprendre, c'est qu'en certain sens, LyX ne sait rien de LaTeX. De fait, du point de vue de LyX, LaTeX est juste un « format de sortie » particulier parmi d'autres formats susceptibles de produire un résultat. D'autres formats sont DocBook, texte brut, et XHTML. LaTeX est bien entendu un format particulièrement important, mais une très petite quantité d'information concernant LaTeX est réellement contenue dans LyX.[1] Cette information, même pour des classes standard comme `article.cls`, est contenue dans les « fichiers de format ». De même, LyX ne connaît pas grand chose de DocBook ou XHTML, tout est décrit dans les fichiers de format.

Vous pouvez imaginer un fichier de format pour une classe de document donnée comme un traducteur entre les constructions LyX — paragraphes et leur style associé, certains types d'inserts, etc. — et les constructions LaTeX, DocBook ou XHTML correspondantes. Pratiquement tout ce que LyX sait de la classe `article.cls`, par exemple, est contenu dans le ficher `article.layout` et dans les fichiers inclus par celui-ci. Pour cette raison, un étude approfondie de ces fichiers est une bonne idée si vous avez l'intention d'écrire un fichier de format. Un bon point de départ est le fichier `stdsections.inc`, inclus par `article.layout`, `book.layout` et de nombreux autres fichiers de format pour de classes de document LaTeX. C'est dans ce fichier que sont définies les sections et les constructions similaires : `stdsections.lyx`

[1] Certaines commandes sont suffisamment compliquées pour être codées « en dur » dans les sources LyX, mais les développeurs considèrent que c'est une mauvaise chose.

décrit pour LyX comment les paragraphes marqués en style Section, SousSection, etc. peuvent être traduits en commandes ou balises LaTeX, DocBook et XHTML. Le fichier article.layout ne contient essentiellement que des inclusions de ces fichiers `std*.inc`.

Définir la correspondance LyX-LaTeX n'est cependant pas la seule fonction des fichiers de format. Leur autre but est de préciser comment les constructions LyX apparaîtront à l'écran. Le fait que ces fichiers assurent ces deux fonctions est souvent source de confusion, parce qu'elles sont tout à fait séparées : expliquer à LyX comment traduire un style de paragraphe en LaTeX ne lui dit pas comment l'afficher ; inversement, expliquer à LyX comment afficher un paragraphe ne lui dit pas comment le traduire en LaTeX (sans parler de lui dire comment LaTeX doit l'afficher). Par conséquent, en général, quand vous devez définir une nouvelle construction LyX, vous devez toujours effectuer deux tâches distinctes : (a) écrire comment la traduire en LaTeX et (b) écrire comment l'afficher.

La même distinction est vraie pour les autre formats de sortie, bien que XHTML soit différent sous divers aspects, du fait que dans ce cas LyX est capable d'utiliser au moins partiellement l'information concernant l'affichage à l'écran pour créer l'information (sous forme de CSS) nécessaire à l'affichage dans un navigateur. Même dans ce cas, cependant, la distinction entre les mécanismes internes de LyX et la façon dont les objets sont restitués à l'extérieur reste valide, et le contrôle séparé des deux facettes reste possible. Voir la 5.4 pour les détails.

5.1. Installer de nouveaux fichiers LaTeX

Certaines installations peuvent ne pas contenir le paquetage LaTeX que vous vouliez justement utiliser avec LyX. Par exemple, vous pouvez avoir besoin de FoilTeX, un paquetage pour préparer des transparents pour rétroprojecteurs. Les distributions LaTeX récentes comme TeXLive (depuis 2008) ou MiKTeX incluent une interface utilisateur pour installer de tels paquetages. Par exemple avec MiKTeX, vous démarrez le programme « Package Manager » pour afficher une liste des paquetages disponibles. Pour installer l'un d'entre eux, il suffit d'effectuer un clic droit sur son nom ou sur le bouton correspondant de la barre d'outil.

Si votre distribution LaTeX n'inclut pas un tel gestionnaire de paquetage, ou si le paquetage n'est pas disponible pour votre distribution, alors faites comme suit pour l'installer manuellement :

1. Récupérer le paquetage depuis CTAN ou ailleurs ;
2. Si le paquetage contient un fichier doté d'un suffixe « `.ins` » (c'est le cas pour FoilTeX), c'est qu'il est « relié » alors ouvrir une console,

aller dans le répertoire de ce fichier et exécuter la commande `latex foiltex.ins`. Vous avez ainsi « déplié » le paquetage et vous disposez des fichiers nécessaires à l'installation. De nombreux paquetages ne sont pas reliés et vous pouvez sauter cette étape ;

3. Vous devez alors décider si ce paquetage doit être disponible pour tous les utilisateurs ou seulement pour vous-même.

   a) Avec les systèmes *nix (Linux, OSX, etc.), si vous souhaitez que le paquetage soit disponible pour tous les utilisateurs, installez-le dans l'arborescence TeX « locale », sinon installez-le dans votre arborescence TeX « utilisateur ». La nécessité de créer ces arborescences, si elles n'existent pas déjà, dépend de votre système. Pour le savoir, examinez le fichier `texmf.cnf`.[2] L'emplacement de l'arborescence « locale » est définie par la variable `TEXMFLOCAL` ; elle a usuellement pour valeur /usr/local/share/texmf/ (installation système), /usr/local/share/texmf/ (installation utilisateur) ou /usr/local/texlive/texmf-local (installation TexLive). L'emplacement de l'arborescence « utilisateur » est définie par `TEXMFHOME` et vaut d'habitude `$HOME/-texmf/` ou
   `$HOME/.texliveXXXX` où XXXX est l'année de la distribution TeXLive (si ces variables ne sont pas prédéfinies, vous devrez le faire vous-même). Vous aurez besoin d'avoir les droits d'administration système pour créer ou modifier l'arborescence « locale », mais votre arborescence « utilisateur » ne devrait pas nécessiter de tels droits.
   D'une manière générale, il est recommandable d'installer dans le répertoire utilisateur parce que cette arborescence ne sera pas modifiée lors d'une mise à jour de votre système. Elle sera également sauvegardée avec tout le reste lors de le sauvegarde de votre répertoire utilisateur « home » (que vous effectuez régulièrement, bien entendu).

   b) Avec Windows, si vous souhaitez que le nouveau paquetage soit disponible pour tous les utilisateurs de votre système, allez dans le répertoire où LaTeX est installé, puis dans le sous-répertoire ~\tex\latex (pour MiKTeX, celui-ci sera implicitement ~ :\Programs\MiKTeX\tex\latex).[3] Créer dans ce répertoire un nouveau répertoire dénommé `foiltex` et copiez tous les fi-

[2] Ce fichier est habituellement dans le répertoire `$TEXMF/web2c`, bien que vous puissiez exécuter la commande `kpsewhich texmf.cnf` pour le localiser.

[3] Notez que ce chemin sera celui des installations anglophones. Pour une installation germanophone, ce serait `~:\Programme\MiKTeX\tex\latex`, de même pour d'autres langages.

chiers du paquetage dedans. Si le paquetage ne doit être accessible que par vous ou si vous n'avez pas les droits d'administration, faites de même, mais dans le répertoire LaTeX local, e.g. avec MiKTeX 2.8 sur WinXP le répertoire
`~:\Documents and Settings\<username>\Application Data\MiKTeX\2.8\tex\latex`.
Avec Vista ce sera le répertoire
~:\Users\<username>\AppData\Roaming\2.8\MiKTeX\tex\latex.

4. Vous devez alors informer LaTeX de la présence de nouveaux fichiers. Ceci dépend de nouveau de la distribution LaTeX :
   a) Avec TeXLive, exécuter la commande `texhash` dans une console. Si vous avez installé le paquetage pour tous les utilisateurs, vous aurez besoin des droits d'administration pour ce faire ;
   b) Avec MiKTeX, si vous avez installé le paquetage pour tous les utilisateurs, démarrez le programme « Settings (Admin) » et appuyez sur le bouton « Refresh FNDB ». Sinon démarrez le programme « Settings » et faites de même.
5. Enfin, vous devez informer LyX de l'existence de nouveaux paquetages : dans LyX, utilisez le menu Outils ▷ Reconfigurer, puis redémarrez LyX.

Le paquetage est maintenant installé. Dans notre exemple, la classe de document `Slides (FoilTeX)` est maintenant disponible dans Document ▷ Paramètres ▷ Classe du document.

Si vous souhaitez utiliser une classe de document qui ne figure pas dans la liste Document ▷ Paramètres ▷ Classe du document, vous devez créer une fichier de « format » (« layout ») pour celle-ci. Ceci est le sujet du paragraphe suivant.

5.2. Types de fichiers de formats

Cette section décrit les différentes variétés de fichiers contenant les informations de format : instructions pour l'affichage par LyX et pour la traduction en LaTeX, DocBook ou XHTML, ou autre format de sortie désiré, des styles de paragraphe et de caractère variés.

Nous essayons ici de donner une description complète de la procédure à suivre ; il y a cependant une telle variété de classes LaTeX supportant des types de document très différents que nous ne pouvons pas espérer couvrir tous les cas possibles ni tous les problèmes que vous pourrez rencontrer. La liste de messagerie des utilisateurs LyX est fréquentée par des personnes expérimentées dans l'écriture des formats,

qui voudront partager ce qu'elles ont appris, n'hésitez pas à poser des questions sur cette liste.

Quand vous prévoyez d'écrire un nouveau format, il est extrêmement utile de regarder les formats distribués avec LyX. Si vous écrivez un fichier de format pour une classe de document LaTeX qui pourrait intéresser d'autres personnes, ou si vous écrivez un module utile, vous pouvez envisager de l'insérer dans les pages dans le wiki LyX, voire de le soumettre à la liste des développeurs, pour qu'il figure dans la distribution de LyX proprement dite[4].

5.2.1. Modules de format

Nous avons mentionné jusqu'ici les « fichier de format ». mais il existe différentes sortes de fichiers qui contiennent des informations de format. Les fichiers de format au sens strict ont pour suffixe `.layout` et contiennent l'information nécessaire à LyX pour les classes de document. Depuis LyX 1.6, cependant, les informations de format peuvent être incluses dans les *modules* de format, qui ont pour suffixe `.module`. Les modules sont aux paquetages LaTeX à peu près ce que sont les fichiers de format aux classes de document, et certains modules — tels que le module « endnotes » — offrent la fonctionnalité d'un seul paquetage. En un sens, le modules de format sont aussi similaires aux fichiers d'inclusion[5] — des fichiers comme `stdsections.inc` — dans le fait que les modules ne sont pas spécifiques du format d'un document donné, mais peuvent être utilisés avec de nombreux formats différents. Cependant l'utilisation d'un fichier d'inclusion (dans `article.cls` par exemple) nécessite la modification du fichier de format `article.layout` ; au lieu de cette modification, les modules sont sélectionnés dans la page Document ▷ Paramètres

Rédiger des modules est la façon la plus simple de se familiariser avec l'écriture des formats, puisque ce peut être aussi simple que l'ajout d'un seul nouveau paragraphe ou insert. Mais en principe, les modules peuvent inclure tout ce que peuvent inclure les formats.

Après la création d'un nouveau module, vous devrez reconfigurer et redémarrer LyX pour que le module apparaisse dans le menu. Cependant, les modifications que vous faites dans le module seront immédiatement actives, si vous ouvrez Document ▷ Paramètres ▷ Modules, sélectionnez un module et appuyez sur « OK ». *Il vous est fortement recommandé de sauvegarder votre travail avant de procéder ainsi*. En fait,

[4] Notez bien que du fait que LyX est diffusé sous licence GPL (General Public License), toute contribution doit être également sous licence GPL.

[5] Ceux-ci peuvent avoir un suffixe quelconque, mais par convention il leur a été attribué le suffixe `.inc`.

il vous est fortement recommandé de ne pas essayer de modifier des modules durant une travail productif sur un document. Bien évidemment, les développeurs s'efforcent de conserver une bonne stabilité à LyX dans de telles situations, mais les erreurs de syntaxe et autres dans votre module pourraient déclencher des comportements bizarres.

5.2.1.1. Format local

Les modules sont à LyX ce que les paquetages sont à LaTeX. Cependant, il arrive que vous souhaitiez disposer d'un insert ou d'un style de caractère pour un seul document, et écrire un module qui sera universel n'a pas grand sens. Vous avez alors besoin du « format local » de LyX.

Vous le trouverez via Document ▷ Paramètres ▷ Format local. La grande zone textuelle vous permet de saisir ce que vous mettriez dans un fichier de format ou de module. Vous pouvez voir le format local comme un module qui n'appartient qu'au document en cours de saisie. De ce fait, vous devez saisir une balise `Format`. N'importe quel format est admissible, mais on utilise d'habitude le format de la version de LyX utilisée (pour LyX 2.3, le format porte le numéro 81).

Lorsque vous avez saisi quelque chose dans la fenêtre de `Format local`, LyX active le bouton « Valider » qui se trouve en bas. Cliquer sur ce bouton entraînera une vérification de la validité du code que vous aurez saisi. LyX vous donne cette information sans hélas l'accompagner d'un diagnostic d'erreurs explicite. Ces erreurs seront cependant affichées dans la fenêtre d'appel si vous avez activé LyX depuis un terminal. Vous ne pourrez pas appliquer ou enregistrer votre format tant qu'il n'est pas valide.

Les précautions indiquées à la fin de la section précédente s'appliquent également ici. Ne jouez pas avec un format local pendant un vrai travail, surtout si vous n'avez pas enregistré votre document. Ceci dit, utiliser un format local avec un document de test peut être très pratique pour essayer de nouvelles choses, voire pour aborder l'écriture d'un module.

5.2.2. Format pour un fichier `.sty`

Les deux situations que vous êtes probablement susceptibles de rencontrer quand vous voulez gérer une nouvelle classe de document LaTeX sont les fichiers LaTeX 2_ε de classe (`.cls`) et de style (`.sty`). Interfacer une nouveau fichier de style est assez facile, Interfacer une nouvelle classe est plus compliqué : nous allons décrire la première opération ici, et la seconde dans le paragraphe suivant.

La situation la plus simple se présente si votre nouvelle classe de document est fournie sous la forme d'un fichier de style à utiliser en com-

binaison avec une classe de document existante, déjà interfacée. Dans notre exemple, le fichier de style sera appelé `ma_classe.sty` et il doit être utilisé avec la classe standard `report.cls`.

Commencez par copier le fichier de format de la classe de document en question dans votre répertoire local.

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

Puis éditez `ma_classe.layout` et changez la ligne :

```
\DeclareLaTeXClass{report}
```

en :

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

Puis ajoutez :

```
Preamble
    \usepackage{ma_classe}
EndPreamble
```

près du début du fichier.

Démarrez LyX et faites Outils ▷ Reconfigurer. Redémarrez LyX et essayez de créer un nouveau document. Vous devriez voir « report (ma_classe) » parmi les classes de document dans la fenêtre Document ▷ Paramètres ▷ Classe de document. Il est probable que certaines des commandes de section devront être différentes de celles de la classe de base[6], vous pouvez donc jouer avec les réglages des différentes sections si vous le souhaitez. Les informations de format pour le sectionnement se trouvent dans `stdsections.inc`, mais il n'est pas nécessaire de recopier ce fichier pour le modifier. Il vous suffit en effet d'ajouter vos modifications au fichier de format, après la ligne `Input stdclass.inc`, qui implique l'inclusion de `stdsections.inc`. Par exemple, vous pourriez ajouter ces lignes :

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

[6] ici `report`

pour modifier la police des titres de chapitre en police sens empattements. Ceci écrasera (ou bien, en l'occurrence, ajoutera) la déclaration existante pour le style Chapitre.

Votre nouveau paquetage peut également fournir des commandes ou des environnements qui n'existent pas dans la classe de base. dans ce cas, il vous faudra ajouter ceux-ci au fichier de format. Voir 5.3 pour savoir comment faire.

Si `ma_classe.sty` peut être utilisé avec plusieurs classes de document différentes, vous trouverez sans doute plus approprié d'écrire un module que vous pourrez charger avec la classe de base. Le module le plus simple qu'il est possible d'écrire dans ce contexte est la suivant :

```
#\DeclareLyXModule{mon paquetage}
#DescriptionBegin
#Support for mypkg.sty.
#DescriptionEnd
Format 81
Preamble
    \usepackage{mypkg}
EndPreamble
```

Un module plus compliqué pourrait modifier le comportement de constructions existantes ou d'en définir de nouvelles. De nouveau, voir 5.3 pour les détails.

5.2.3. Format pour un fichier `.cls`

Deux possibilités se présentent.

Soit le fichier de classe est fondé sur une classe de document existante. Par exemple, de nombreuses classes pour une thèse sont des variantes de `book.cls`. Pour vérifier que c'est le cas pour la vôtre, cherchez une ligne comme

```
\LoadClass{book}
```

dans le fichier de classe. Si c'est la cas, vous pouvez essentiellement procéder comme dans la section précédente, bien que la ligne `DeclareLaTeXClass` soit différente. Si votre nouvelle classe est une thèse, et est fondée sur `book`, cette ligne doit être[7] :

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

[7] Et enregistrer le fichier comme `thesis.layout` facilitera les choses : LyX suppose que la classe de document a le même nom que le fichier de format.

Soit le fichier de classe est original. Dans ce cas, vous devrez probablement créer votre propre format. Nous vous suggérons fortement de copier un fichier de format existant qui utilise une classe LaTeX similaire et de le modifier si c'est possible. Utilisez au moins un fichier existant comme point de départ pour savoir quels sont les éléments dont vous devez vous occuper. Encore une fois, les spécificités sont décrites ci-après.

5.2.4. Création de modèles

Une fois qu'un fichier de format a été crée pour une classe de document, vous désirerez peut-être créer un *modèle* correspondant. Un modèle est une sorte de mode d'emploi pour votre format, montrant comment il peut être utilisé, mais avec un contenu fictif. Vous pouvez jeter un œil aux divers modèles distribués avec LyX pour illustration.

Les modèles sont créés comme les documents normaux. La seule différence est que les documents normaux contiennent tous les réglages possibles, y compris les polices et la taille du papier. Or on ne veut en général pas qu'un modèle modifie ces réglages implicites. Pour cette raison, l'auteur d'un modèle doit enlever les commandes correspondantes comme `\font_roman` ou `\papersize` du fichier modèle LyX. C'est faisable avec un simple éditeur de texte, comme `vi` ou `notepad`.

Mettez les fichiers modèles que vous avez créés et édités dans `MonRép/templates/`, copiez ceux que vous utilisez du répertoire global de modèles `RépLyX/templates/` vers le même endroit, et redéfinissez le répertoire de modèles dans la fenêtre Outils ▷ Préférences ▷ Répertoires.

Notez que le modèle `defaults.lyx` a un statut particulier. Il est chargé à chaque fois que vous créez un nouveau document avec Fichier ▷ Nouveau afin d'avoir les réglages implicites. Pour créer ce modèle depuis LyX, vous n'avez qu'à ouvrir un document ayant déjà les réglages voulus, et appuyer sur le bouton Enregistrer comme valeurs implicites.

5.2.5. Mise à jour des anciens fichiers de format

La syntaxe des fichiers de format change à chaque publication d'une nouvelle version de LyX, et les anciens fichiers de format doivent donc être convertis dans la nouvelle syntaxe. Si LyX lit un fichier de format écrit avec une syntaxe ancienne, il appelle le script `layout2layout.py` pour le convertir dans un fichier temporaire doté de la nouvelle syntaxe. Le fichier original reste intact. Si vous utilisez le fichier de format souvent, vous voudrez sans doute rendre les modifications permanentes, pour éviter des conversions répétitives de la part de LyX. Pour ce faire, vous pouvez appeler le convertisseur manuellement :

1. renommez le fichier `myclass.layout` comme `myclass.old`
2. exécutez la commande
   `python LyXDir/scripts/layout2layout.py myclass.old myclass.layout`
   dans laquelle `LyXDir` est le nom de votre répertoire LyX au niveau système.

Le convertisseur ne gère que les changements de syntaxe. Il ne peut pas gérer les modifications du contenu des fichiers inclus, ceux-ci doivent être convertis séparément.

5.2.6. Fichiers moteurCitation

Une espèce particulière de fichier de format est dénommée moteurCitation et se manifeste via des fichiers `*.citeengine` situés dans le sous-répertoire `citeengines/`. Ils sont destinés à spécifier les paquetages LaTeX destinés à la création de bibliographies, comme natbib, jurabib ou biblatex, mais le traitement par LyX des citations BibTeX ordinaires (sans paquetage additionnel) est également défini par de tels fichiers.

Plus précisément, on définit quels sont les paquetages à charger par LyX, quelles sont le commandes de citation disponibles, comment celles-ci seront affichées dans LyX (dans la fenêtre de travail, les dialogues, les menus contextuels), ainsi que dans les résultats XHTML et textuels. En outre, les fichiers spécifient les variantes de style disponibles (auteur-année, numérique, etc ;) et leurs particularités Les fichiers moteurCitation servent également à engendrer les options disponibles via Document ▷ Paramètres ▷ Bibliographie ▷ Processeur.

Bien qu'un fichier moteurCitation se présente comme un fichier de format ordinaire pouvant contenir n'importe quelle directive de format, il contient habituellement les directives particulières telles que `MaxCiteNames`, `CiteFramework`, `CiteEngine` et `CiteFormat`. La syntaxe des deux dernières est décrite ci-après dans 5.3.14 et 5.3.15, ainsi que dans les fichiers eux-mêmes.

5.3. Syntaxe des fichiers de format

Les sections suivantes vous expliquent à quoi vous vous attaquez quand vous décidez de mettre les mains dans le cambouis, et de créer ou d'éditer votre propre fichier de format. Notre conseil est d'aller doucement et d'enregistrer souvent pour faire des essais. Ce n'est pas si dur que ça, mais il y a une multitude d'options et vous pouvez vous laisser submerger si vous essayez d'en faire trop d'un coup. Il est plus facile d'utiliser

des formats existants comme référence ou modèle ou de modifier un fichier de format existant pour vos besoins.

Toutes les balises décrites dans ce chapitre sont insensibles à la casse : cela veut dire que `Style`, `style` et `StYlE` sont une seule et même balise. Les valeurs possibles sont inscrites entre crochets après le nom de la fonction. La valeur implicite d'une fonction quand elle n'est pas définie dans le fichier décrivant la classe de texte est `mise en évidence`. Si le paramètre est d'un type particulier la valeur implicite est indiquée ainsi : `float=`*`default`*.

5.3.1. Déclaration et classification d'une classe de document

Les lignes qui commencent par un # dans un fichier de format sont les commentaires. Il y a une exception à cette règle : tous les `.layout` doivent commencer par ceci :

```
#% Do not delete the line below; configure depends on this[8]
#  \DeclareLaTeXClass{Article (Standard Class)}
#  \DeclareCategory{Articles}
```

Les deuxième et troisième lignes servent lors de la (re)configuration de LyX. Le fichier de format est lu par le script LaTeX `chkconfig.ltx`, dans un mode spécial où # est ignoré. La première ligne est juste un commentaire LaTeX, la deuxième contient la déclaration obligatoire de la classe de texte et la troisième contient la classification facultative de la classe. Si ces lignes apparaissent dans un fichier appelé `article.layout`, elles définissent alors une classe de texte appelée `article` (le nom du fichier de format) qui utilise la classe de document LaTeX `article.cls` (implicitement, le même nom que le format). La chaîne « Article (Standard Class) » qui apparaît ci-dessus sert de description de la classe de texte dans la fenêtre Document ▷ Paramètres ▷ Classe de document. La catégorie (« Articles » dans l'exemple) est également utilisés dans la fenêtre Document ▷ Paramètres ▷ Classe de document : les classes sont regroupées par catégories (qui sont aussi des genres de documents, les catégories typiques sont « Articles », « Livres », « Rapports », « Présentations », « Lettres », « Curricula vitae », etc.). Si la troisième lignes est vide, la classe apparaîtra comme « Sans catégorie ».

Supposons que vous ayez écrit votre propre classe de texte qui utilise la classe de document `article.cls`, mais dans laquelle vous avez changé l'apparence des en-têtes de sections. Si vous la mettez dans un fichier `mon_article.layout`, l'en-tête de ce fichier doit être :

[8] Ne pas effacer la ligne ci-dessous ; configure compte dessus.

```
#% Do not delete the line below; configure depends on this
#  \DeclareLaTeXClass[article]{Article (avec Mes En-têtes)}
#  \DeclareCategory{Articles}
```

Ceci déclare une classe de texte `mon_article`, associée avec la classe de document LaTeX `article.cls` et décrite comme « Article (avec Mes En-têtes) ». Si votre classe de texte repose sur plusieurs paquetages, vous pouvez la déclarer ainsi :

```
#% Do not delete the line below; configure depends on this
#  \DeclareLaTeXClass[article,machin.sty]{Article (avec Mes En-têtes)}
#  \DeclareCategory{Articles}
```

Ceci indique que votre classe de texte utilise le paquetage `machin.sty`.

Notez que ces déclarations peuvent aussi recevoir un paramètre optionnel déclarant le nom de la classe de document (mais pas une liste).

Donc, de la manière la plus explicite, la forme d'une déclaration de format est la suivante :

```
#  \DeclareLaTeXClass[class,package.sty]{description du format}
#  \DeclareCategory{catégorie}
```

Il n'est nécessaire de déclarer la classe que si le nom de la classe LaTeX et le nom du fichier de format diffèrent, ou si des paquetages sont à spécifier. Si le nom de la classe n'est pas spécifiée, LyX suppose qu'elle est identique à celle du fichier de format.

Une fois que la classe de texte a été modifiée à votre goût, tout ce que vous avez à faire est de la copier soit dans `RépLyX/layouts/` soit dans `MonRép/layouts/` et de faire Outils ▷ Reconfigurer. Quittez LyX et redémarrez-le ; votre nouvelle classe de texte devrait alors être disponible avec les autres.

Une fois le fichier de format installé, vous pouvez le modifier et vérifier l'effet de ces modifications sans reconfigurer ou redémarrer LyX.[9] Vous pouvez imposer un rechargement du format en service en utilisant la fonction layout-reload. Cette fonction n'est implicitement liée à aucun raccourci — vous pouvez bien sûr la lier à une touche vous-même. Si vous voulez utiliser cette fonction, il vous suffit de la saisir dans le minitampon.

[9] Avec les versions de LyX antérieures à 1.6, il vous fallait redémarrer LyX pour valider les modifications effectuées dans les fichiers de format. De ce fait, les modifications pouvaient prendre beaucoup de temps.

Attention : layout-reload est une fonctionnalité « avancée ». Il vous est *fortement* recommandé de sauvegarder votre travail avant de procéder ainsi. En fait, il vous est *fortement* recommandé de ne pas essayer de modifier des formats durant une travail productif sur un document. Utilisez un document de test, les erreurs de syntaxe et autres dans votre module pourraient déclencher des comportements bizarres. En particulier, de telles erreurs pourraient inciter LyX à considérer les formats en service comme incorrects et à essayer de basculer sur un autre format.[10] Les développeurs s'efforcent de conserver une bonne stabilité à LyX dans de telles situations, mais il vaut mieux être sûr que regretter[11].

5.3.2. Déclaration d'un module

La première ligne d'un module doit être rédigée comme suit :

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

L'argument obligatoire de `\DeclareLyXModule`, entre accolades, est le nom du module, tel qu'il apparaîtra dans Document ▷ Paramètres ▷ Modules. L'argument entre crochets est facultatif : il déclare n'importe quel paquetage LaTeX dont dépend le module. Il est également possible d'utiliser la clé depuis->vers en argument optionnel : elle déclare que le module ne peut être employé que s'il existe un chemin de conversion entre les formats `depuis` et `vers`. La déclaration `\DeclareCategory` n'est pas strictement obligatoire, mais vous devriez l'ajouter, car elle facilite la recherche d'un module. Voyez les catégories de modules existantes et utilisez l'une d'entre elles si cela convient.

La déclaration et la catégorie du module devront être suivies par des lignes descriptives comme celles-ci :[12]

```
#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
```

[10] Les erreurs de syntaxe vraiment graves peuvent même stopper LyX. Ceci provient du fait que certaines erreurs rendent LyX incapable de lire la moindre information de format. Faites attention…

[11] Puisque nous en sommes au conseils : faites des sauvegardes régulières. Et soyez gentil avec votre maman.

[12] En anglais de préférence si le module doit être diffusé par LyX. Cette description apparaîtra dans la liste des messages à traduire et sera donc traduite lors de la mise à jour de l'interface.

Cette description est utilisée dans Document ▷ Paramètres… ▷ Modules pour informer l'utilisateur de la fonction du module. La ligne `Requires` permet d'identifier les modules qui doivent être activés avec celui-ci ; la ligne `Excludes` permet d'identifier les modules interdits avec celui-ci. Les deux lignes sont facultatives et, comme indiqué, les modules dans une liste doivent être séparés par une barre verticale : |. Noter que les modules requis sont traités sur le mode disjonctif : *au moins un* des modules requis doit être utilisé. De même, *aucun* des modules exclus ne doit être utilisé. Noter également que les les modules sont identifiés par leur nom de fichier, sans le suffixe `.module` . Ainsi `telmodule` est vraiment `telmodule.module`.

5.3.3. Déclaration d'un fichier moteurCitation

Un fichier moteurCitation doit commencer par une ligne comme celle-ci :

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

L'argument obligatoire entre accolades est le nom du module, comme il apparaîtra dans Document ▷ Paramètres ▷ Bibliographie. L'argument entre crochets est facultatif : il précise le ou les paquetages LaTeX dont dépend le format.

La déclaration doit ensuite être suivie d'une description suivant l'exemple ci-après[13] :

```
# 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
```

Cette description apparaît donc traduite dans Document ▷ Paramètres ▷ Bibliographie pour informer l'utilisateur.

[13] En anglais de préférence si le module doit faire l'objet d'une publication avec LyX. Cette description apparaîtra dans la liste ds messages qui seront traduits lors de la prochaine mise à jour de l'interface.

5.3.4. Numéro d'identification syntaxique

La première ligne non commentée doit contenir le numéro d'identification syntaxique du contenu du fichier de format :

Format [int] identification syntaxique du fichier de format

Ce balisage a été introduit dans LyX 1.4.0 Les fichiers de format des versions antérieures n'avaient pas de numéro de format explicite et sont affectés du `Format 1`. Le numéro de format du présent fichier est le numéro 620. mais chaque version de LyX peut lire les fichiers de format d'une version plus ancienne, comme elle peut lire les documents crées avec une version plus ancienne. Il n'y a cependant aucun moyen de revernir aux numéros d'identification antérieurs.

5.3.5. Paramètres généraux d'une classe de texte

Voici les paramètres généraux qui décrivent l'aspect du document pour une classe complète (ceci ne veut pas dire qu'il *doivent* apparaître dans le fichiers `.layout` plutôt que dans les modules. Un module peut contenir n'importe quelle balise de format) :

AddToCiteEngine <moteur> étend les possibilités d'affichage des références des citations. Voir 5.3.14 pour les détails. Doit être fermé par « `End` ».

AddToHTMLPreamble ajoute une information qui sera incluse dans le bloc `<head>` quand le document est exporté en XHTML. Typiquement, ce paramètre est utilisé pour exporter de l'information en style CSS, mais il peut être utilisé pour toute information valide entre les balises `<head>`. Doit être fermé par « `EndPreamble` ».

AddToPreamble ajoute une information au préambule du document. Doit être fermé par « `EndPreamble` ».

BibInToc [*0*, 1] doit être ajouté avec la valeur 1 (ou true) si la classe du document ajoute la bibliographie à la table des matières. Ceci évite à la bibliographie d'être insérée deux fois.

CiteEngine <moteur> définit le possibilités pour l'affichage des références de citation. Voir 5.3.14 pour les détails. Doit se terminer par « End ». Utilisé principalement dans les fichiers moteurCitation (voir 5.2.6). Notez que si vous spécifiez ceci dans un fichier de format ou un module, les définitions de moteurCitation seront écrasées. Voir aussi `AddToCiteEngine`.

CiteFormat définit le style à utiliser pour afficher les informations bibliographiques. Voir 5.3.15. Doit être fermé par « End ». Utilisé principalement dans les fichiers moteurCitation (voir 5.2.6). Un CiteFormat utilisé dans un fichier de format ou un module écrasera la définition de moteurCitation.

CiteFramework [*bibtex*,biblatex] précise si Biblatex ou BibTEX est utilié pour engendrer la bibliographie. Utilisé principalement dans les fichiers moteurCitation (voir 5.2.6).

ClassOptions décrit diverses options globales comprises par la classe du document. Voir la 5.3.6 pour les détails. Doit être fermé par « End ».

Columns [*1*, 2] fixe le nombre implicite de colonnes (une ou deux). Peut être changé dans la fenêtre Document ▷ Paramètres.

Counter [chaîne] définit les paramètres d'un nouveau compteur. Si le compteur n'existe pas, il est créé ; s'il existe, il est modifié. Doit être fermé par « End ».
Voir la 5.3.12 pour les détails.

DefaultFont décrit la police implicite dans le document. Voir la 5.3.13 pour une description. Doit être fermé par « EndFont ».

DefaultModule [<module>] spécifie un module inclus implicitement avec cette classe de document, donné par son nom de fichier sans suffixe .module. L'utilisateur peut toujours le retirer, mais il sera actif à l'ouverture du document (ceci s'applique pour un nouveau document, ou quand cette classe est choisie pour un document existant).

DefaultStyle [<style>] indique le style qui sera appliqué aux nouveaux paragraphes, habituellement Standard. Ce sera implicitement le premier style défini si ce paramètre n'est pas fixé, mais vous êtes vivement encouragé à utiliser cette directive.

DocBookRoot [chaîne] l'élément racine (en haut du document) à utiliser en exportant des documents en DocBook avec cette classe. La valeur implicite est « article ».

DocBookForceAbstract [booléen] l'élément racine aura toujours une balise <abstract> si la valeur est «true ». La valeur implicite est «false ».

ExcludesModule [<module>] indique que le module spécifié (donné par son nom de fichier sans le suffixe .module) ne peut pas être utilisé

avec cette classe de document. Ceci peut être utilisé par exemple dans un format pour une revue pour éviter e.g. l'utilisation du module `theorems-sec` (numérotation des théorèmes par sections). Ce paramètre *ne peut pas* être utilisé dans un module : les modules ont leur propre méthode d'exclusion (voir 5.2.1).

`Float` définit un nouveau flottant. Voir la 5.3.9. Doit être fermé par « `End` ».

`HTMLPreamble` définit l'information qui sera incluse dans le bloc `<head>` quand le document est exporté en XHTML. Notez bien que ceci écrasera toute déclaration `HTMLPreamble` ou `AddToHTMLPreamble` antérieure (utiliser
`AddToHTMLPreamble` si vous désirez simplement ajouter du contenu). Doit être fermé par « `End` ».

`HTMLTOCSection` [`<style>`] définit le style utilisé pour la table des matières, la bibliographie, et similaires, lorsque le document est exporté en HTML. Pour les articles, ceci devrait normalement valoir `Section` ; pour les livres `Chapter`. Sans précisions, LyX essaiera de deviner quel style utiliser.

`IfCounter` [`<compteur>`] modifie les propriétés d'un compteur donné. Si le compteur n'existe pas, la directive est ignorée. Doit être fermé par « `End` ».
Voir la 5.3.12 pour les détails au sujet des compteurs.

`Input` [`<nom de fichier>`] permet d'inclure un autre fichier de définition de format pour éviter de réécrire des commandes. C'est souvent le cas des fichiers de format standard, comme `stdclass.inc`, qui contient la plupart des réglages de base.

`InputGlobal` [`<nom de fichier>`] est une variante de la directive `Input` qui ne recherche pas de fichiers dans le répertoire utilisateur. Ceci permet de créer un fichier `nom.layout` ou `nom.inc` dans le répertoire utilisateur qui peut inclure un fichier global de même nom via `InputGlobal nom` or `InputGlobal nom.inc`, respectivement (avec `Input`, le fichier s'inclurait récursivement). De cette façon, vous pouvez modifier les fichiers globaux sans avoir à les copier complètement.

`InsetLayout` [`<type>`] cette section (re)définit le format d'un insert. Elle peut être appliquée à un insert existant pour obtenir un nouvel insert défini par l'utilisateur, e.g. un nouveau style de caractères. Doit être fermé par « `End` ».
Voir la 5.3.10 pour les détails.

LeftMargin [chaîne] indique la largeur de la marge gauche à l'écran, par exemple « MMMMM » (noter que ce n'est pas une « longueur », comme « 2ex. »).

MaxCiteNames [entier] cet entier définit le nombre maximum de noms affichés dans le style auteur-année avant que la citation bascule vers « PremierAuteur et al. ». Utilisé principalement dans les fichiers moteurCitation (voir 5.2.6).

ModifyInsetLayout [<type>] modifie le format d'un insert. Si le format n'existe pas, cette directive est ignorée. Doit être fermé par « End ».

ModifyStyle [<style>] modifies les propriétés de style d'un paragraphe donné. Si le style n'existe pas, le directive est ignorée. Doit être fermé par « End ».

NoCounter [<compteur>] efface un compteur existant, habituellement défini dans un fichier inclus.

NoFloat [<flottant>] efface un flottant existant. C'est très utile quand vous voulez supprimer un flottant qui a été défini dans un fichier inclus.

NoStyle [<style>] efface un style existant.

OutputFormat [<format>] indique quelle sorte de format de fichier (tel que défini dans les préférences de LyX) est produit par cette classe. Elle est surtout utile quand OutputType est literate et que l'on veut définir un nouvelle sorte de document en programmation littéraire. La chaîne est redéfinie comme « latex » quand la directive correspondante OutputType est trouvée.

OutputType [*latex*, literate] indique quelle sorte de document résultant sera obtenu par cette classe.

PackageOptions [chaîne chaîne] précise les options, données par le seconde chaîne de caractères, pour le paquetage nommé par le première chaîne. Par exemple, « PackageOptions natbib square » déclenchera le chargement de natbib avec l'option square (pour les TEXperts, ceci force LyX à exporter \PassOptionsToPackage{natbib}{square} avant le chargement de natbib).

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] définit la taille de page implicite. Cette directive est utilisée par certains convertisseurs.

PageStyle [plain, empty, headings] fixe la mise en page implicite. Peut être changée dans la fenêtre Document ▷ Paramètres….

Preamble fixe le préambule du document LaTeX. Notez bien que ceci remplacera complètement toute autre directive Preamble ou AddToPreamble (utiliser cette dernière directive si vous voulez seulement ajouter des éléments au contenu). Doit être fermé par « EndPreamble ».

ProvideInsetLayout [<type>] définit le format d'un insert s'il n'existe pas encore. Si le format existe, cette directive est ignorée. Doit être fermé par « End ».

Provides [chaîne] [*0*, 1] décrit si la classe fournit la fonctionnalité chaîne. Une fonctionnalité est en général le nom d'un paquetage (amsmath, makeidx, …) ou d'une commande LaTeX (url, boldsymbol,…). Voir A pour une liste des fonctionnalités.

ProvidesModule [chaîne] indique que ce format fournit la fonctionnalité du module indiqué, qui doit être spécifié par le nom du fichier dans le suffixe .module. Ceci sera utilisé typiquement si le format inclut directement le module, plutôt que via le paramètre DefaultModule pour indiquer qu'il doit être utilisé. Il pourrait être également utilisé dans un module fournissant une implantation équivalente de la même fonctionnalité.

ProvideStyle [<style>] crée une nouveau style de paragraphe s'il n'existe pas déjà. S'il existe, l'entrée est ignorée. Doit être fermé par « End ».

Requires [chaîne] indique que la classe nécessite la fonctionnalité chaîne. Plusieurs fonctionnalités doivent être séparées par des virgules. Noter que l'on ne peut requérir que des fonctionnalités reconnues (voir A pour accéder à la liste des fonctionnalités). Si vous demandez un paquetage doté d'options spécifiques, vous pouvez utiliser en plus PackageOptions.

RightMargin [chaîne] indique la largeur de la marge droite à l'écran, par exemple « MMMMM ».

SecNumDepth [int=3] fixe quels sectionnements doivent être numérotés. Correspond au compteur secnumdepth en LaTeX.

Sides [*1*, 2] fixe l'option implicite d'impression recto seul ou en recto verso. Peut être changé dans la fenêtre Document ▷ Paramètres.

Style [<nom>] définit un nouveau style de paragraphe. Si le style n'existe pas, il est créé ; s'il existe déjà, ses paramètres sont modifiés. Doit être fermé par « End ».
Voir la 5.3.7 pour les détails.

TableStyle [<nom>] définit le style de tableau implicite utilisé à la création d'un tableau. Les styles suivants sont disponibles :

- `Formal_with_Footline` : style formel (« booktabs ») avec uniquement des lignes horizontales, et des lignes haute et basse épaisses, la première et las dernière lignes étant éventuellement séparées du corps du tableau avec une ligne fine centrée ;
- `Formal_without_Footline` : identique àu précédent, mais la dernière ligne n'est pas séparée avec une ligne fine centrée ;
- `Simple_Grid` : lignes de tableau simples ;
- `Grid_with_Head` : comme `Simple_Grid`, mais avec la ligne des titres décalée par une seconde ligne horizontale. Ceci est également le style implicite de LyX ;
- `No_Borders` : tableau sans lignes.

TitleLatexName [`string="maketitle"`] définit nom de la commande ou de l'environnement mentionné ci-dessous.

TitleLatexType [*`CommandAfter`*, `Environment`] indique le genre de balisage utilisé pour définir le titre d'un document. `CommandAfter` signifie que la commande définie par `TitleLatexName` sera insérée après le dernier format avec «`InTitle 1`». `Environment` correspond au cas `TitleLatexName` est un environnement, dans lequel doit être inclus tous les formats avec «`InTitle 1`».

TocDepth [`int=3`] fixe quels sectionnements sont inclus dans la table des matières. Correspond au compteur `tocdepth` en LaTeX.

5.3.6. Section `ClassOptions`

La section `ClassOptions` peut contenir les directives suivantes :

FontSize [`string="10|11|12"`] liste les tailles de police disponibles comme police principale du document, séparées par un «|». Il est possible de saisir un nombre quelconque.

FontSizeFormat [`chaîne`] définit le format de l'option de taille de police. Implicitement :`$$spt`. `$$s` est mis à la place de la taille de police.

PageSize [`string="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"`] définit la liste des dimensions de page disponibles, séparées par «|». Actuellement, seules les dimensions proposées

sont reconnues. D'autres peuvent être saisies en tant qu'options de classe personnalisées.

PageSizeFormat [string] définit le format de l'option de dimension de page. Implicitement :$$spaper. $$s est mis à la place de la dimension de la page.

PageStyle [string="empty|plain|headings|fancy"] liste les mises en page disponibles, séparées par un «|».

Other [string=""] introduit certaines options de la classe de document, séparées par une virgule, qui seront ajoutées à la partie optionnelle de la commande\documentclass.

La section ClassOptions doit être fermée par « End ».

5.3.7. Styles de paragraphe

La description d'un style de paragraphe ressemble à ceci[14] :

```
Style nom
 ...
End
```

dans lequel on peut mettre les commandes suivantes :

AddToToc [string=""] paragraphe apparaissant dans la table des matières du type donné. Désactivé par une chaîne de caractères vide. Voir aussi les directives OutlinerName et IsTocCaption. Implicitement : désactivé.

Align [*block*, left, right, center] alignement de paragraphe.

AlignPossible [*block*, left, right, center] *liste* des alignements possibles séparés par une virgule. Certains styles LaTeX interdisent certains alignements, car ils n'auraient aucun sens. Par exemple une énumération alignée à droite ou centrée est impossible.

Argument [int] définit l'argument numéro <int> d'une commande ou environnement associé au style courant. La définition doit être fermée par EndArgument. Voir 5.3.11 pour plus d'informations.

AutoNests inclut une liste de formats (séparés par une virgule) devant être englobés dans et après le format courant. N'a de sens que pour des formats susceptibles d'englobement comme des environnements. Doit être terminé par « EndAutoNests ». Voir aussi IsAutoNestedBy.

[14] qui peut servir soit à définir un nouveau format soit à en modifier un déjà existant.

BabelPreamble noter que cette directive écrasera complètement toute déclaration `BabelPreamble` antérieure pour ce style. Doit être fermée par `EndBabelPreamble`. Voir la 5.3.8 pour les détails d'utilisation.

BottomSep [`float=0`][15] l'espacement vertical qui sépare du paragraphe suivant le dernier paragraphe d'une série doté de cette directive. Si le paragraphe suivant est doté d'un autre style, les séparations ne sont pas simplement additionnées, mais le maximum est pris en considération.La même chose que `TopSep` pour le dernier paragraphe.

Category [`chaîne`] fixe la catégorie pour ce style. Ceci est utilise pour regrouper les styles apparentés dans le menu déroulant des styles de la barre d'outils. N'importe quelle chaîne de caractères peut être utilisée, mais vous voudrez peut-être utiliser des catégories existantes pour vos propres styles.

CopyStyle [`chaîne`] sert à copier toutes les caractéristiques d'un style déjà existant dans un nouveau style. Notez bien que ceci copie le style tel qu'il est défini à l'instant de la copie. Des modifications ultérieures ne seront pas reportées là où il a été copié.

DependsOn [`<nom>`] nomme un style dont le préambule devra être placé *avant* celui-ci. Ceci permet d'assurer une relation d'ordre entre les morceaux de préambule quand les définitions de macros dépendent les unes des autres.[16]

DocBookGenerateTitle [`bool=false`] crée une étiquette `title` après l'étiquette d'encapsulation. Ce paramètre ne doit être utilisé qu'avec `DocBookWrapperTag`, sinon le titre sera émis *avant* les contenus de l'environnement. Le titre créé sera identique à l'étiquette LyXHTML : une combinaison entre le type d'environnement et son numéro. L'utilisation principale concerne les cas où DocBook n'a pas de dénomination voisine d'un environnement LaTeX et où les utilisateurs sont obligés de revenir à un conteneur générique comme `figure`, qui réclame un titre alors qu'il n'y en a pas en LaTeX. Cette fonctionnalité est utilisée largement pour les environnements de type théorème.

EndLabelType [*`No_Label`*, `Box`, `Filled_Box`, `Static`] fixe le type de marqueur qui se trouve à la fin du paragraphe (ou de la suite de pa-

[15] Noter que « float » ici désigne un nombre réel, e.g. 1.5

[16] Noter que, à part cette fonctionnalité, il n'y a aucun moyen de garantir un ordonnancement des préambules. L'ordre que vous constatez avec une version de LyX peut changer sans préavis dans les versions ultérieures.

ragraphes si `LatexType` est `Environment`, `Item_Environment` ou `List_Environment`). `No_Label` signifie « rien », `Box` (respectivement `Filled_Box`) est un carré blanc (respectivement noir) en général placé à la fin des démonstrations. `Static` est une chaîne de caractères explicitement donnée.

`EndLabelString` `[string=""]` fixe la chaîne utilisée pour le marqueur quand
`EndLabelType` vaut `Static`.

`Font` fixe la police utilisée à la fois pour le corps du texte *et* pour le marqueur. Voir la 5.3.13. Noter qu'en définissant cette police on définit aussi automatiquement `LabelFont` avec la même valeur. Il faut donc définir celle-ci d'abord pour fixer `LabelFont`.

`ForceLocal` `[int=0]` est utile pour forcer la portabilité de nouveaux styles vers des versions stables de LyX. La première version stable qui interprète cette directive est LyX 2.1.0. L'argument est un nombre qui peut être 0, -1 ou n'importe quel entier positif. Si `ForceLocal` est positif, il sera toujours émis vers le préambule du document. Lors de la lecture d'un fichier `.lyx`, les définitions de style du préambule du document sont ajoutées à la classe du document. De ce fait même les versions de LyX plus anciennes peuvent gérer ce style. L'argument de `ForceLocal` est un numéro de version : si le style est lu, et si le numéro de version est inférieur au numéro de version du style existant dans la classe de document, le nouveau style est ignoré. Si le numéro de version est supérieur, le nouveau style remplace l'ancien. La valeur -1 signifie un numéro de version infini, c'est-à-dire que le style est toujours utilisé.

`FreeSpacing` [*0*, 1] LyX ne permet pas d'habitude d'insérer plus d'une espace entre deux mots, car une espace est considérée comme un séparateur de mots, non comme un caractère ou un symbole en tant que tel. C'est très bien mais c'est parfois ennuyeux, par exemple pour taper un code source de programme ou du code LaTeX brut. C'est pourquoi on peut activer `FreeSpacing`. LyX créera les espaces insécables correspondants aux espaces additionnelles si `Passthru 1` n'est pas spécifié. Notez que `FreeSpacing` implique `KeepEmpty`.

`HTML*` sont utilisées avec l'exportation XHTML. Voir 5.4.1.

`InPreamble` [*0*, 1] indique avec 1 que que le style doit être inclus dans le préambule plutôt que dans le corps du document. Ceci est utile pour les classes de document qui désirent que les informations telles que le titre et l'auteur figurent dans le préambule. Notez bien

que ceci ne fonctionne que pour les styles pour lesquels `LateXType` est `Command` ou `Paragraph`.

`InTitle` [*0*, 1] indique avec 1 que le style fait partie d'un bloc de titre (voir aussi `TitleLatexType` et `TitleLatexName` dans les paramètres généraux).

`IsAutoNestedBy` inclut une liste de formats (séparés par une virgule) qui doivent englober celui-ci. N'a de sens que pour des formats englobables comme des environnements. Doit être terminé par « EndIsAutoNestedBy ». Voir aussi `AutoNests`.

`IsTocCaption` [*0*, 1] si positionné à 1 et AddToToc activé, le paragraphe ajoute un résumé de son contenu dans son élément dans la table des matières. Sinon, seule l'étiquette apparaît si elle existe.

`ItemCommand` [`string="item"`] est la commande LaTeX permettant de déclarer un élément d'une liste. La commande doit être définie sans la barre oblique inverse qui la précède habituellement (implicitement « item », qui est émis comme`\item` en LaTeX).

`ItemSep` [`float=0`] crée un espacement supplémentaire entre les paragraphes du même style dans un environnement. Si vous emboîtez des paragraphes d'autres formats dans un environnement, ils seront espacés de `ParSep`. Mais les éléments de l'environnement seront en plus espacés de `ItemSep`. Noter que c'est un *multiplicateur*.

`KeepEmpty` [*0*, 1] permet de laisser un paragraphe vide, ce que LyX ne permet pas d'habitude car il ne générerait rien en LaTeX. Il y a quelques cas où c'est pourtant utile : dans un modèle pour une lettre, les champs requis peuvent être laissés vides, pour que les gens ne les oublient pas ; dans certains cas particuliers, un style peut aussi servir de séparateur, et ne pas contenir de texte.

`LabelBottomSep` [`float=0`] fixe l'espacement vertical entre le marqueur et le corps du texte. Sert seulement pour les marqueurs qui sont au dessus du corps du texte (`Top_Environment` et `Centered_Top_Environment`).

`LabelCounter` [`string=""`] dénomme le compteur pour la numérotation automatique. Pour que le compteur soit associé à votre étiquette, il vous faut le référencer dans la directive `LabelString`. Ceci fonctionnera avec les `LabelTypes`, `Static`, `Above` and `Centered`, au moins.
Il *peut* être indiqué si `Labeltype` est `Enumerate`, bien que ce cas soit un peu compliqué. Supposez que vous déclariez «`LabelCounter`

monEnum ». Les compteurs effectivement utilisés sont alors monEnumi, monEnumii, monEnumiii et monEnumiv, tout à fait comme en LaTeX. Ces compteurs doivent être tous déclarés séparément.
Voir la 5.3.12 pour les détails sur les compteurs.

LabelFont fixe la police utilisée pour le marqueur. Voir la 5.3.13.

LabelIndent [string=""] fixe le texte indiquent l'importance de l'indentation d'un marqueur.

LabelSep [string=""] fixe le texte indiquent l'espacement horizontal entre le marqueur et le corps du texte. Sert seulement pour les marqueurs qui ne sont pas au dessus du corps du texte.

LabelString [string=""] fixe la chaîne utilisée comme marqueur avec le LabelType Static. Quand LabelCounter est indiqué, cette chaîne peut contenir les instructions de formatage particulières décrites dans la 5.3.12.

LabelStringAppendix [string=""] est utilisée dans une annexe à la place de LabelString. Noter que toute directive LabelString écrase LabelStringAppendix.

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

Manual veut dire que le marqueur est le premier mot (avant la première vraie espace). Utilisez des espaces insécables si vous voulez mettre plus d'un mot dans le marqueur.

Static veut dire qu'il est défini dans le style (voir LabelString). Ceci sera affiché « en ligne », au début du paragraphe.Si le LatexType est Environment, alors il ne sera affiché que dans le premier paragraphe de toutes séquence de paragraphes du même Style.

Above et Centered sont des cas particuliers de Static. Le marqueur sera imprimé au dessus du paragraphe, soit au début de la lignes, soit centré.

Sensitive est un cas particulier pour les marqueurs de légende « Figure » et « Tableau ». Sensitive signifie que le mot affiché (défini en dur) dépend du type de flottant : il est défini dans les sources comme « FloatType N » où N est la valeur du compteur associé au flottant. Au cas où une légende est insérée en dehors d'un flottant, le LabelString apparaîtra comme « Inapproprié ! ».

Enumerate produit le type habituel d'étiquettes d'énumération. The number type needs to be set in the Compteur, voir 5.3.12.

Itemize produit des symboles pour les différents niveaux. Les types de puces affichés peuvent être définis via Document ▷ Paramètres ▷ Puces.

Bibliography ne doit être utilisé qu'avec `LatexType BibEnvironment`.

LangPreamble écrasera complètement toute autre directive `LangPreamble` antérieure. Doit être fermée par `EndLangPreamble`.
Voir la 5.3.8 pour les détails d'utilisation.

LatexName [<nom>] fixe le nom de l'objet LaTeX correspondant, soit l'environnement, soit la commande.

LatexParam [<paramètre>] fixe le paramètre facultatif de l'objet `LatexName` correspondant. Ce paramètre ne peut pas être changé depuis LyX (utiliser `Argument` pour les paramètres personnalisables). Ceci sera émis tel quel après tous les `Arguments` LaTeX.

LatexType [*Paragraph*, `Command, Environment, Item_Environment, List_Environment, Bib_Environment`] décrit comment traduire l'environnement en LaTeX.[17]

Paragraph veut dire « rien de spécial ».

Command veut dire `\`*LatexName*`{…}`.

Environment veut dire `\begin{`*LatexName*`}…\end{`*LatexName*`}`.

Item_Environment est la même chose que `Environment`, sauf qu'il génère un `\item` pour chaque paragraphe de l'environnement.

List_Environment est la même chose que `Item_Environment`, sauf que
`LabelWidthString` est passé en paramètre de l'environnement. `LabelWidthString` peut être défini dans la fenêtre Édition ▷ Paramètres de Paragraphe.

Bib_Environment fonctionne comme `Environment`, mais ajoute l'argument obligatoire nécessaire (l'étiquette la plus longue) à la directive de début de l'environnement bibliographique : `\begin{thebibliography}{99}`. Cette directive n'est donc nécessaire que pour les environnements bibliographiques. L'étiquette implicite « 99 » peut être modifiée par l'utilisateur dans les réglages de paragraphe d'une entrée bibliographique.

[17] `LatexType` est peut-être trompeur, dans la mesure où ces règles s'appliquent également aux classes DocBook. Explorer les fichiers des classes DocBook (noms de fichiers `db_*.inc`) pour voir des exemples particuliers.

En collectant ces derniers éléments, l'exportation LaTeX sera ou bien :

```
\LatexName[LatexParam]{…}
```

ou bien :

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

en fonction du type LaTeX.

LeftDelim [chaîne] définit une chaîne de caractères insérée au début du contenu du style. Un passage à la ligne dans la sortie peut être indiquée par
.

LeftMargin [string=""] fixe la marge gauche : si vous mettez des styles dans un environnement, les LeftMargin ne seront pas ajoutées directement, mais avec un facteur $\frac{4}{\text{profondeur}+4}$. Notez que ce paramètre sert aussi quand Margin est définie comme Manual ou Dynamic. Il est alors ajouté à la marge manuelle ou dynamique.
Par exemple, « MM » signifie que le paragraphe est indenté avec la largeur de « MM » dans la police normale. Il est possible de créer une largeur négative avec « - ». Cette solution a été choisie pour que l'apparence soit la même quelle que soit la police d'écran.

Margin [*Static*, Manual, Dynamic, First_Dynamic, Right_Address_Box] fixe le type de marge à gauche du format.

Static veut dire une marge fixe.

Manual veut dire que la marge de gauche dépend de ce qu'il y a dans la fenêtre Édition ▷ Paramètres de Paragraphe. Ceci sert à obtenir des listes bien mises en page sans tabulations.

Dynamic veut dire que la marge dépend de la taille du marqueur. Ceci sert entre autres aux en-têtes numérotés. Il est évident que l'en-tête « 5.4.3.2.1 Très long en-tête » doit avoir une marge de gauche plus grande (autant que « 5.4.3.2.1 » plus l'espace) que « 3.2 Très long en-tête », même si les autres traitements de texte ne savent pas le faire.

First_Dynamic est similaire, mais seulement la toute première ligne du paragraphe est dynamique, les autres étant statiques ; ceci est utile par exemple pour les descriptions.

Right_Address_Box signifie que la marge est choisie pour que la ligne la plus longue du paragraphe touche la marge de droite. Ceci sert à typographier une adresse sur le bord droit de la page.

NeedProtect [*0*, 1] indique si les commandes fragiles doivent être protégées par `\protect` dans ce style (Note : ceci n'indique pas si cette commande elle-même doit être protégée).

NeedCProtect [-1, *0*, 1] avec la valeur 1 , protège si nécessaire les macros qui contiennent ce format avec `\cprotect` (cf. le paquetage cprotect) et par suite autorise du texte verbatim dans les macros. Avec la valeur implicite 0,`\cprotect` est utilisé si un élément imbriqué le demande. La valeur -1 empêche toute utilisation de `\cprotect` dans le format, même si un élément imbriqué le demande.

NeedMBoxProtect [*0*, 1] implique que des commandes spécifiques dans ce style (comme`\cite` et `\ref`) soient protégées dans une `\mbox`. Ceci est particulièrement requis pour les styles qui utilisent les commandes ulem ou soul, qui parcourent leur contenu de manière compliquée.

Newline [0, *1*] indique si les nouvelles lignes sont traduites ou non en sauts de ligne LaTeX (\\). La traduction peut être désactivée pour permettre d'éditer plus confortablement du LaTeX depuis LyX.

NextNoIndent [*0*, 1] indique si LyX indente ou non la première ligne du paragraphe suivant. 1 veut dire qu'il ne peut pas, 0 veut dire qu'il peut s'il le veut.

ObsoletedBy [<nom>] dénomme un style qui a remplacé ce style en cours. Ceci est utilisé pour renommer un style en conservant la rétrocompatibilité.

ParagraphGroup [0, 1] détermine si les paragraphes consécutifs du même type sont traités ensemble. Ceci a pour effet de déclencher une seule fois pour le groupe l'impression du `GuiLabel`. Implicitement, ceci est vrai pour les environnements `LaTeXType Environment` et `Bib_Environment` et faux pour tous les autres types.

ParbreakIsNewline [*0*, 1] indique que les paragraphes ne seront pas séparés par une ligne vide dans le résultat LaTeX, mais par un retour à la ligne ; combiné avec `PassThru 1`, ceci permet d'émuler une éditeur plein texte (comme l'insert code TeX).

ParIndent [string=""] fixe l'indentation de la toute première ligne d'un paragraphe. `Parindent` est fixé pour un format donné, à l'exception du format `Standard`, car l'indentation d'un paragraphe dans l'environnement `Standard` peut être empêchée par `NextNoIndent`. De plus, les paragraphes de style `Standard` emboîtés dans d'autres environnements utilisent le `ParIndent` de l'environnement, pas le

leur. Par exemple, les paragraphes `Standard` dans une énumération ne sont pas indentés.

ParSep [`float=0`] fixe l'espacement vertical entre deux paragraphes dans le style.

Parskip [`float=0`] fixe la valeur d'interligne entre paragraphes. LyX donne le choix entre `Indentation` et `Interligne` pour séparer les paragraphes. Quand on choisit `Indentation`, cette valeur n'est pas prise en compte. Quand on choisit `Interligne`, la valeur de `ParIndent` n'est pas prise en compte et tous les paragraphes sont séparés par ce paramètre `Parskip`, en plus de l'interligne normal. L'espacement vertical est calculé par `valeur*DefaultHeight()` où `valeur` est la valeur choisie pour `Parskip` et `DefaultHeight()` est la hauteur d'une ligne dans la police normale. De cette façon, l'aspect reste le même quelle que soit la police à l'écran.

PassThru [*0*, `1`] indique si le contenu du paragraphe doit être passé sous forme brute, c'est-à-dire sans traitements particuliers dont LaTeX aurait besoin.

PassThruChars [`chaîne`] indique les caractères isolés qui doivent être passés sous forme brute, c'est-à-dire sans traitements particuliers dont LaTeX aurait besoin.

Preamble fixe le contenu à inclure dans le préambule LaTeX quand le style est utilisé. Utile pour définir des macros, charger des paquetages, etc. requis pour ce style particulier. Doit être fermé par « `EndPreamble` ».

RefPrefix [`chaîne`] indique le préfixe à utiliser pour créer des étiquettes référant les paragraphes de ce type. Ceci permet l'utilisation de références mises en forme.

Requires [`chaîne`] indique que le style requiert la fonctionnalité chaîne (voir A pour la liste des fonctionnalités). Si vous demandez un paquetage doté d'options spécifiques, vous pouvez en outre utiliser `PackageOptions` en tant que paramètre général pour la classe de texte (voir 5.3.5).

ResetArgs [*0*,`1`] réinitialise les arguments LaTeX de ce style (tels que définis par la directive `Argument`). Ceci est utile si vous avez dupliqué un style en utilisant `CopyStyle`, et que vous ne voulez pas hériter de ses arguments (obligatoires et optionnels).

ResumeCounter [*0*,1] redémarre un compteur qui est normalement remis à zéro a chaque nouvelle séquence de formats. Ceci n'est utile pour le moment que lorsque`LabelType` vaut `Enumerate`.

RightDelim [chaîne] définit une chaîne de caractères insérée à la fin du contenu du style. Un passage à la ligne dans la sortie peut être indiquée par `<br/>`.

RightMargin [chaîne] similaire à `LeftMargin`.

Spacing [*single*, onehalf, double, other <valeur>] définit l'interligne implicite dans ce style. Les paramètres `single`, `onehalf` et `double` correspondent respectivement à un facteur multiplicatif de 1, 1,25 et 1,667. Si vous mettez le paramètre `other`, vous devez aussi mettre une valeur numérique qui servira de facteur multiplicatif. Notez que, contrairement aux autres paramètres, `Spacing` implique de générer du code LaTeX spécifique, qui utilise le paquetage LaTeX `setspace`.

Spellcheck [0, *1*] active la correction orthographique du style. Implicitement vrai.

StepParentCounter [*0*,1] incrémente le compteur parent d'un compteur donné au début d'une nouvelle séquence de formats. Ceci n'est utile pour le moment que lorsque `LabelType` vaut `Enumerate`.

TextFont fixe la police utilisée pour le corps du texte. Voir la 5.3.13.

TocLevel [int=3] fixe le niveau du style dans le table des matières. Ceci est utilisé pour la numérotation automatique des en-têtes.

ToggleIndent [*default*, always, never] détermine si l'indentation de la première ligne de ce paragraphe peut être réglée via le menu des réglages de paragraphe. Si *default* est choisi, l'indentation peut être réglée si les réglages de document utilisent un style de paragraphe avec « indentation » ; avec `always`, l'indentation est toujours réglable quels que soient les réglages du document ; avec `never`, l'indentation n'est jamais réglable.

TopSep [float=0] fixe L'espacement vertical qui sépare le premier paragraphe dans une série du même style, du paragraphe qui le précède. Si le paragraphe précédent est dans un autre style, les séparations ne s'ajoutent pas, mais LyX prend le maximum des deux.

5.3.8. Internationalisation des styles de paragraphes

LyX effectue depuis longtemps la traduction des informations de format, mais, jusqu'à la version 2.0, ceci ne s'appliquait qu'à l'interface utilisateur et non pas par exemple au résultat PDF. Ainsi les auteurs francophones devaient avoir recours à des astuces pénibles s'ils voulaient obtenir « Théorème 1 » au lieu de « Theorem 1 ». Grâce à Georg Baum, ce n'est plus le cas.

Si un `Style` définit un texte qui doit apparaître dans le le document typographique, il peut utiliser `LangPreamble` et `BabelPreamble` pour traiter correctement les documents non anglophones, et même multilingues. L'extrait suivant (du fichier `theorems-ams.inc`) illustre le fonctionnement :

```
Preamble

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

En principe, toute commande LaTeX autorisée peut apparaître dans les directives `LangPreamble` et `BabelPreamble`, mais dans la pratique elles se présenteront typiquement comme ce qui est montré ici. La clé de la traduction correcte du texte typographié est la définition de le commande LaTeX `\claimname` et son utilisation avec `\newtheorem`.

La directive `LangPreamble` fournit ce qu'il faut pour l'internationalisation fondée sur le langage global du document. Le contenu de la directive sera inclus dans le préambule, comme avec la directive `Preamble`. Ce qui la rend singulière est l'utilisation de la « fonction » `_()`, qui sera remplacée, lorsque LyX produira le résultat LaTeX, par la traduction de son argument dans la langue du document[18].

La directive `BabelPreamble` est plus compliquée, car elle fournit une fonctionnalité adaptée aux documents multilingues et propose par conséquent une interface pour le paquetage `babel` Son contenu sera ajouté

[18] Pour ceux qui s'intéressent au fonctionnement sous-jacent, la fonction « `_()` » cache la fonction `gettext()`, qui constitue le cœur du mécanisme de traduction de l'interface LyX, et qui est étendue ici à la traduction à la volée de code LaTeX.

au préambule une fois pour chaque langue utilisée par le document. Dans ce cas, l'argument de `_()` sera remplacé par sa traduction dans le langage en question : l'expression `$$lang` est remplacée par le nom de la langue (telle qu'il est connu de `babel`).

Un document en allemand qui inclut aussi un paragraphe en français aura donc ceci dans le préambule :

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

LaTeX et `babel` vont alors conspirer pour créer le texte correct dans le résultat imprimable.

Un point important à noter est que les traductions sont celles qui sont fournies par LyX lui-même, via le fichier `layouttranslations`. Ceci signifie que `LangPreamble` et `BabelPreamble` ne sont effectifs que dans les fichiers de format fournis avec LyX, puisque les fichiers de format créés par l'utilisateur ne seront pas explorés par les fonctions d'internationalisation, sauf si le fichier `layouttranslations` est modifié en conséquence. Ceci dit, tout fichier de format créé dans le but d'être distribué avec LyX doit inclure ces directives s'il y a lieu. Notez bien que les traductions des styles de paragraphe ne changeront jamais avec les publications de maintenance des versions de LyX (e.g. de la version 2.4.x à la version 2.4.y). Il est cependant très probable qu'à l'occasion d'une publication majeure (e.g. de 2.3.x à 2.4.0), de nouvelles traductions ou corrections soient introduites

5.3.9. Flottants

Il faut définir les flottants (`figure`, `tableau`...) dans la classe elle-même. Si vous cherchez à savoir comment mettre à jour une classe déjà existante, il suffit probablement que vous ajoutiez

```
Input stdfloats.inc
```

à un endroit raisonnable dans la classe[19]. Si vous voulez implémenter une classe de texte qui propose un nouveau type de flottant (comme la classe AGU fournie avec LyX), les explications ci-dessous vont sans doute vous être utiles :

`AllowedPlacement` [`string`= !htbpH] définit les options de placement pour ce type de flottant. La valeur est une chaîne de paramètres

[19] N'oubliez pas de jeter aussi un œil sur les compteurs dans la section suivante.

de placement, les caractères autorisés sont : *h* (*here*, « ici si possible »), *t* (*top* « haut de page »), *b* (*bottom*, « bas de page »), *p* (« page de flottants »), *H* (« forcément ici ») et *!* (« ignorer les règles LaTeX). L'ordre des caractères est indifférent. Si aucune option n'est autorisée, utiliser la chaîne de caractères *none*.

AllowsSideways [`0`, *1*] précise si le flottant peut subir une rotation grâce au paquetage LaTeX `rotfloat` (sidewaysfloat). Positionner à `0` si le flottant ne peut pas bénéficier de cette fonctionnalité.

AllowsWide [`0`, *1*] précise si le flottant possède une variante étoilée qui s'étend sur les colonnes dans un paragraphe à deux colonnes. Positionner à `0` si le flottant ne peut pas bénéficier de cette fonctionnalité.

Extension [chaîne] définit le suffixe d'un fichier auxiliaire contenant la liste des flottants de ce type. LaTeX écrit les légendes dans ce fichier.

GuiName [chaîne] définit la chaîne de caractères qui se trouvera dans les menus et également dans la légende. Ceci est traduit dans le langage courant si `babel` est utilisé.

HTML* sont utilisées avec l'exportation XHTML. Voir 5.4.

IsPredefined [*0*, `1`] indique si le flottant est déjà défini dans la classe de document ou s'il faut charger le paquetage LaTeX `float` pour le définir à la volée. La valeur implicite est `0` qui signifie : utiliser `float`. Elle doit être positionnée sur `1` si le flottant est déjà défini par la classe de document.

ListCommand [chaîne] détermine la commande utilisée pour engendrer une liste de flottants du type considéré ; le « \ » initial doit être omis. Cette directive *doit* être utilisée si `UsesFloatPkg` est faux, puisqu'il n'y a alors pas de procédure pour activer cette commande. La directive est ignorée si `UsesFloatPkg` est vrai, puisqu'il y a alors une procédure.

ListName [chaîne] fixe le titre utilisé pour une liste des flottants du type considéré (figures, tableaux, etc.). Il est utilisé pour l'étiquette à l'écran dans LyX ; il est passé à LaTeX pour être utilisé comme titre, et il est également utilisé comme titre dans l'exportation XHTML. Il sera traduit dans le langage du document.

NumberWithin [chaîne] (paramètre facultatif) détermine si les flottants de cette classe seront numérotés en suivant un type de section du

document. Par exemple, si `NumberWithin` es « `chapter` », les flottants seront numérotés chapitre par chapitre.

`Placement` [chaîne] définit le placement implicite pour cette catégorie de flottants. C'est comme en LaTeX standard : `t`, `b`, `p` et `h` pour haut, bas, page et ici respectivement[20]. En plus vous avez un nouveau type, `H`, qui ne correspond pas vraiment à un flottant, car il signifie de le positionner « ici » et nulle part ailleurs. Notez cependant que le type `H` est spécial et, à cause de détails d'implantation, ne peut pas être utilisé avec des flottants non définis dans la classe de document. Si vous n'avez pas compris tout ce baratin, mettez simplement «`tbp`».

`PrettyFormat` [`string=""`] définit le format utilisé pour les références mises en forme à ce compteur. Par exemple, on peut vouloir les références aux tableaux apparaître sous la forme « Tableau 2 ». La chaîne de caractères peut contenir « `##` » ou une spécification de compteur (voir la documentation de `LabelString` in 5.3.12.) La première forme sera remplacée par le numéro du compteur lui-même. Ainsi, pour les sections, ce serait *Section* `##`, ou peut-être *section* `\arabic{section}` (qui apparaîtra par exemple comme section 2.7).

`RefPrefix` [chaîne] détermine le préfixe à utiliser lors de la création d'étiquettes référant les flottants du type considéré. Ceci permet l'utilisation de références mises en forme. Notez que vous pouvez supprimer tout préfixe défini en copiant un style en utilisant la valeur spécifique « OFF », qui doit être en capitales.

`Requires` [`string`] s'utilise comme avec les styles de paragraphes, voir 5.3.7.

`Style` [chaîne] fixe le style utilisé quand on définit le flottant avec `\newfloat`.

`Type` [chaîne] définit le « type » de la nouvelle classe de flottants, comme « program » ou « algorithm ». Après l'instruction `\newfloat` appropriée, vous disposez de commandes comme `\begin{program}` ou `\end{algorithm*}`. Noter que définir un flottant de type `type` définit automatiquement les compteur correspondant, de nom `type`.

`UsesFloatPkg` [`0`,*`1`*] indique si ce flottant est défini en utilisant le paquetage LaTeX `float`, soit par la classe de document ou par un paquetage, soit à la volée par LyX.

Notez que la définition d'un flottant de type *`type`* implique la déclaration du compteur correspondant de nom *`type`*.

[20] Notez que l'ordre dans lequel vous mettez ces lettres est sans importance, comme en LaTeX.

5.3.10. Inserts flexibles et InsetLayout

Les inserts flexibles sont de deux sortes :

- styles de caractères (`CharStyle`) : ceux-ci définissent un balisage sémantique correspondant à des commandes LaTeX telles que `\noun` et `\code`.

- définis par l'utilisateur (`Custom`) : ceux-ci permettent de définir des inserts repliables personnalisés, similaires au code TeX, aux notes de bas de page, etc. Un exemple évident est l'insert de note en fin de document, défini dans le module `endnote`.

Les inserts flexibles sont définis en utilisant la directive `InsetLayout`, qui va être expliquée dans la suite.

La directive `InsetLayout` a une autre fonction : elle permet de personnaliser le style de différents types d'inserts. Actuellement, `InsetLayout` permet de personnaliser les paramètres de style pour les notes de bas de page, les notes en marge, les inserts de note, les inserts de code TeX (ERT), les branches, les listings, les index, les boîtes, les tables, les algorithmes, les URL et les légendes, aussi bien que de définir des inserts flexibles.

La définition d'un `InsetLayout` doit commencer par une lignes de la forme :

```
InsetLayout <type>
```

Ici `<type>` indique l'insert dont le style doit être défini, et il y a quatre cas.

1. Le style d'un insert préexistant doit être modifié. Dans ce cas, peuvent être `<type>` un quelconque des éléments suivants : `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`, ou `URL`.

2. Le style d'un insert flexible doit être défini. Dans ce cas, `<type>` doit être de la forme `Flex:<nom>`, où `nom` peut être n'importe quel identificateur valide qui n'est pas utilisé dans un insert existant. L'identificateur peut inclure des espaces, mais alors il faut enclore l'ensemble entre apostrophes. Noter que la définition d'un insert flexible *doit* aussi inclure une entrée `LyXType`, précisant quel est le type d'insert qu'il définit.

3. Le style d'une branche utilisateur doit être défini. Dans ce cas, `<type>` doit être de la forme « `Branch:<nom>` », où `nom` peut être un identificateur de branche valide défini dans le document utilisateur. L'identificateur peur contenir des espaces, mais dans ce cas la chaîne toute entière doit être enclose entre apostrophes. Le principal intérêt de cette fonctionnalité est est de permettre l'inclusion LaTeX de branches spécifiques suivant les besoins de l'utilisateur.

4. Le style d'une légende utilisateur (ou de classe) doit être défini. Dans ce cas, `<type>` doit être de la forme « `Caption:<nom>` », où `nom` spécifie le nom de la légende tel qu'il apparaît dans le menu. Voir la légende standard (`Caption:Standard`), les légendes spécifiques des classes KOMA-Script (`Caption:Above`, `Caption:Below`) ou le module Légendes multilingues (`Caption:Bicaption`) pour des applications.

La définition d'un `InsetLayout` peut inclure les entrées suivantes :

`AddToToc` [`string=""`] insert apparaissant dans la table des matières du type donné. Désactivé par une chaîne de caractères vide. Voir aussi les directives `OutlinerName` et `IsTocCaption`. Ceci ne fonctionne que pour les inserts flexibles. Implicitement : désactivé.

`AllowedInInsets` inclut une liste d'inserts (séparés par des virgules) au sein desquels cet insert peut être ajouté. Doit se clore par `EndAllowedInIns`. Si vous souhaitez également que l'insertion soit possible dans des arguments spécifiques des inserts ciblés, ajouter le nom de l'argument après une @ (e. g., `Mon_Insert@post:1`). Notez que ceci ne s'applique qu'au premier niveau d'insertion. Voir aussi `AllowedInLayouts`.

`AllowedInLayouts` inclut une liste d'inserts (séparés par des virgules) au sein desquels cet insert peut être ajouté. Doit se clore par `EndAllowedInLay`. Notez que ceci ne s'applique qu'au premier niveau d'insertion. Voir aussi `AllowedInsets`.

`AllowedOccurrences` [`int`] si `AllowedInInsets` ou `AllowedInLayouts` est défini, ceci peut être utilisé pour déterminer combien de fois au maximum l'insert peut être ajouté à un insert donné ou au paragraphe (groupe).

`AllowedOccurrencesPerItem` [*`0`*, `1`] si cette valeur est vraie, `AllowedOccurrences` s'applique aux paragraphes isolés dans un environnement de type liste (avec `\items`).

`Argument` [`int`] définit un numéro d'argument d'une commande ou d'un environnement associé au style courant. La définition doit être close par `EndArgument`. Voir 5.3.7 et 5.3.11 pour plus d'informations.

BabelPreamble, définit un préambule pour les modifications de commandes de langue ; voir 5.3.8.

BgColor [<nom>] définit la couleur d'arrière-plan de l'insert. See B pour une liste des noms de couleur disponibles.

ContentAsLabel [*0*, 1] indique s'il faut utiliser le contenu de l'insert comme une étiquette, quand l'insert est fermé. Valeur implicite : « faux ».

CopyStyle [<type>] fonctionne comme les styles de paragraphe, voir 5.3.7. Notez que vous devez spécifier le type complet, e. g. `CopyStyle Flex:<nom>`.

CustomPars [*0*, 1] indique si l'utilisateur peut utiliser le dialogue Édition ▷
Paramètres de paragraphe pour personnaliser le paragraphe.

Decoration peut être `Classic`, `Minimalistic`, ou `Conglomerate`, décrivant l'apparence du cadre et des boutons de l'insert. les notes de bas de page utilisent en général `Classic`, les inserts des code TEX `Minimalistic`, et les styles de caractères `Conglomerate`.

Display [0, *1*] n'est utilisé que lorsque `LatexType` vaut `Environment`. Indique si l'environnement se trouvera à part dans le résultat imprimable ou bien apparaîtra inclus dans le texte environnant. S'il est positionné à faux, on suppose que l'environnement LATEX ignore les espaces (avec un caractère retour ligne inclus) après les balises `\begin{`*LatexName*`}` et `\end{`*LatexName*`}`. Valeur implicite : « vrai »..

EditExternal [*0*,1] permet de modifier le contenu d'un insert via un éditeur externe (en utilisant n'importe quel éditeur défini pour le format d'exportation du document).

End est requis pour fermer la déclaration d'un `InsetLayout`.

Font définit la police utilisée pour le texte du corps de l'insert *et* pour le marqueur. Voir 5.3.13. Noter qu'en définissant cette police on définit aussi automatiquement `LabelFont` avec la même valeur. Il faut donc définir celle-ci d'abord pour fixer `LabelFont` différemment.

FixedWidthPreambleEncoding [*0*, 1] force un encodage à taille fixe pour les éléments traduits du code des préambules `BabelPreamble` et `LangPreamble` engendré par ce format. Ceci est nécessaire pour les paquetages LATEX spéciaux comme `listings` qui ne fonctionnent pas avec des encodages à taille variable comme utf8. Ce réglage

est ignoré lors de l'utilisation de moteurs admettant complètement Unicode comme XeTEX ou LuaTEX.

ForceLocalFontSwitch [*0*, 1] lors de l'utilisation de babel, force l'utilisation un commutateur de police *local* (`\foreignlanguage`), à la place d'un commutateur *global* (comme `\selectlanguage`).

ForceLTR [*0*, 1] impose le langage « latex », conduisant à un résultat de gauche à droite (Left-to-Right, latin), e. g. en code TEX ou dans un URL. Une verrue.

ForceOwnlines [*0*, 1] force un saut de ligne dans le résultat LATEX avant le début de l'insert et après sa fin. Ceci garantit que l'insert apparaît en lignes isolées, pour la lisibilité.

ForcePlain [*0*, 1] indique si `PlainLayout` doit être utilisé, ou bien si l'utilisateur peut modifier le style de paragraphe utilisé dans l'insert. Valeur implicite : « faux ».

FreeSpacing [*0*, 1] fonctionne comme les styles de paragraphe, voir 5.3.7.

HTML* sont utilisées avec l'exportation XHTML. Voir 5.4.

InheritFont [0, *1*]: la police dans l'insert est héritée du parent de l'export LATEX si ce paramètre vaut 1, et à l'écran également. Sinon, c'est la police implicite du document qui est utilisée.

InToc [*0*, 1] indique s'il faut inclure le contenu de cet insert dans les chaînes créées pour le panneau « Plan pour la table des matières complète, indépendamment du réglage de AddToToc ». L'utilisateur ne voudra pas, par exemple, que le contenu d'une note de bas de page dans un titre de section soit incluse dans la tables des matières affichée dans le plan, mais voudra habituellement voir affiché le contenu d'un style de caractère. Valeur implicite : « faux », pas d'inclusion.

IsTocCaption [*0*, 1] si positionné à 1 et AddToToc activé, le paragraphe ajoute un résumé de son contenu dans son élément dans la table des matières. Sinon, seule l'étiquette apparaît si elle existe.

KeepEmpty [*0*, 1] fonctionne comme les styles de paragraphe, voir 5.3.7.

LabelFont définit la police utilisée pour le marqueur. Voir la 5.3.13. Noter que cette directive ne peut jamais apparaître avant `Font`, sinon elle sera sans effet.

LabelString [chaîne] définit la chaîne de caractères qui apparaîtra sur le bouton ou ailleurs en tant que marquer d'insert. Quelques types d'inserts (`code TEX` et `Branche`) modifient ce marqueur à la volée.

LangPreamble définit un préambule dépendant de la langue, voir 5.3.8.

LatexName [<nom>] définit le nom du code LaTeX correspondant, soit un environnement, soit une commande.

LatexParam [<paramètre>] définit le paramètre facultatif pour le `LatexName` correspondant, y compris des paires de crochets `[]`. Ce paramètre ne peut pas être modifié depuis LyX (utiliser `Argument` pour les paramètres personnalisables). Il sera émis tel quel après tous les `Arguments` LaTeX.

LatexType [`Command, Environment, None , `] précise comment le style doit être traduit en LaTeX[21].

None signifie « rien de particulier » ;

Command signifie `\`*`LatexName`*`{…}` ;

Environment signifie `\begin{`*`LatexName`*`}…\end{`*`LatexName`*`}`.

En rassemblant ceci, la sortie LaTeX sera soit :

```
\LatexName[LatexParam]{…}
```

soit :

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

suivant le `LatexType`.

LeftDelim [chaîne] définit une chaîne de caractères insérée au début du contenu du style. Un passage à la ligne dans la sortie peut être indiquée par `<br/>`.

LyxType peut être `charstyle`, `custom`, `element`, ou `end` (indiquant une définition vide terminant les définitions de styles de caractères, etc). Cette entrée est requise dans les inserts flexibles et n'est significative que là. Parmi d'autres choses, elle détermine dans quel menu cet insert va apparaître. Fixer `LyxType` à `charstyle` positionnera `MultiPar` à « faux » et `ForcePlain` à « vrai ». `MultiPar` peut être positionné à « vrai » ou `ForcePlain`à « faux », pour les inserts `charstyle`, en le positionnant *après* avoir fixé `LyxType`.

[21] `LatexType` est peut-être trompeur, puisque ces règles s'appliquent aussi aux classes SGML. Voir les fichiers de classe SGML pour des exemples spécifiques.

MenuString [chaîne] définit une chaîne de caractères pour le menu. Vous pouvez associer un accélérateur en accolant le caractère voulu à la chaîne séparé par « | » (e. g. “Mon insert|M”). Cette spécification est facultative. Si indéfini, le nom utilisé dans le menu sera celui de l’insert dans la déclaration du type.

MultiPar [*0*, 1] indique si plusieurs paragraphes sont autorisés dans cet insert. Ceci positionnera aussi CustomPars à la même valeur et ForcePlain à la valeur opposée. Ceux-ci peuvent être repositionnés s’ils sont utilisés *après* MultiPar. Valeur implicite : « vrai ».

NeedProtect [*0*, 1] indique si les commandes fragiles de cet insert doivent êtres protégées par \protect (Note : ceci ne dit *rien* sur le fait que la commande elle-même doive être protégée). Valeur implicite : « faux ».

NeedCProtect [*0*, 1] protège si nécessaire les macros qui contiennent cet insert avec \cprotect (cf. le paquetage cprotect) et par suite autorise du texte verbatim dans les macros. Implicitement faux.

NeedMBoxProtect [*0*, 1] implique que des commandes spécifiques dans cet insert (comme\cite et \ref) soient protégées dans une \mbox. Ceci est particulièrement requis pour les styles qui utilisent les commandes ulem ou soul, qui parcourent leur contenu de manière compliquée. Implicitement faux.

NewlineCmd [string] définit une commande différente de la commande implicite (\\) pour les ruptures de ligne. La barre inversée initiale ne doit pas être saisie.

NoInsetLayout [<layout>] supprime un InsetLayout existant.

ObsoletedBy [<layout>] nomme un InsetLayout qui remplace cet InsetLayout. Ceci est utilisé pour renommer un InsetLayout en conservant la compatibilité.

ParbreakIgnored [*0*, 1] avec la valeur 1, les sauts de paragraphes seront ignorés dans le résultat imprimable. Ceci peut être utile pour les inserts dont le contenu doit être aligné dans la fenêtre LyX, sans que cela soit répercuté dans la sortie.

ParbreakIsNewline [*0*, 1] fonctionne comme avec les styles de paragraphe, voir 5.3.7.

PassThru [*0*, 1] fonctionne comme avec les styles de paragraphe, voir 5.3.7.

Preamble fonctionne comme avec les styles de paragraphe, voir 5.3.7.

RefPrefix [chaîne] indique le préfixe à utiliser pour créer des étiquettes référant les paragraphes de ce type. Ceci permet l'utilisation de références mises en forme.

Requires [chaîne] As with paragraph styles, see 5.3.7.

ResetArgs [*0*, 1] réinitialise les arguments LaTeX de ce style (définis via la directive Argument). Ceci est utile si vous avez dupliqué un style via CopyStyle, mais que vous ne voulez pas hériter de ses arguments (obligatoires et optionnels).

ResetsFont [0, *1*] avec la valeur 1, les changements de police sont réappliqués dans les inserts respectifs (dans la sortie) même si l'insert lui-même est t dans la portée de ces changements de police (e. g., `\textbf{Texte environnant \moninsert{\textbf{contenu}}…}` plutôt que `\textbf{Texte environnant \moninsert{contenu}…}`. Utiliser ceci a un sens pour le commandes qui reactualisent les réglages de police de manière interne (e. g. notes de bas de page). Notez que positionner incorrectement ceci peut conduire à des résultats non désirés (e. g., avec `\emph{Texte environnant \moninsert{\emph{con` le contenu est droit du fait que `\emph` bascule. Implicitement `0` : les changements de police ne sont pas réappliqués dans l'insert.

RightDelim [chaîne] définit une chaîne de caractères insérée à la fin du contenu du style. Un passage à la ligne dans la sortie peut être indiquée par `<br/>`.

Spellcheck [0, *1*] active le correcteur orthographique sur le contenu de cet insert. Valeur implicite : « vrai ».

5.3.11. Arguments

À la fois les styles de paragraphe et les formats d'insert admettent des *arguments* en plus du contenu principal. Ceci est particulièrement utile pour des éléments comme les en-têtes de section et n'a de sens qu'avec LaTeX. Chaque argument (facultatif ou obligatoire) d'une commande ou d'un environnement — sauf pour l'argument obligatoire correspondant au contenu — a une définition séparée, où le nombre spécifie l'ordre des arguments. La définition doit se terminer par EndArgument. Ainsi une commande avec deux arguments sera spécifiée comme suit :

```
Argument 1

...
```

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

Dans la définition de l'`Argument`, les spécifications suivantes sont possibles :

- `LabelString [chaîne]` définit la chaîne de caractères qui apparaîtra à la fois dans le menu (pour insérer cet argument) et dans le bouton d'insertion d'argument (sauf si vous spécifiez également un `MenuString`). Pour le menu, vous pouvez définir un raccourci en ajoutant le caractère désiré à la chaîne, séparé par « | » (e. g. « `Éléments personnalisables|é` »);

- `MenuString [chaîne]` définit une chaîne particulière pour le menu. Vous pouvez définir un raccourci en ajoutant le caractère désiré à la chaîne, séparé par « | » (e. g. « `Éléments personnalisables|é` »). Cette spécification est facultative, si elle n'existe pas, la `LabelString` sera utilisée pour le menu ;

- `Tooltip [chaîne]` définit un texte explicatif qui apparaît dans la bulle d'aide quand le curseur passe sur l'insertion d'argument ;

- `Mandatory [`*`0`*`, 1]` précise si l'argument est obligatoire (1) ou optionnel (0). Les arguments obligatoires ont émis comme une chaîne vide s'ils ne sont pas précisés, alors que les arguments optionnels ne sont pas émis du tout. Implicitement, les arguments obligatoires sont insérés entre `{...}`, alors que les arguments optionnels sont insérés entre `[...]` ;

- `NewlineCmd [string]` définit une commande différente de la commande implicite (`\\`) pour les ruptures de ligne. La barre inversée initiale ne doit pas être saisie.

- `Requires [int=0]` définit une autre argument (par son numéro d'ordre) dont cet argument demande l'émission s'il est lui-même émis. Par exemple en LaTeX, les arguments optionnels demandent l'émission d'autres arguments optionnels situés avant eux dans la liste (au moins vides), comme dans `\command[][argument]{text}`, ou ceci peut être précisé par la directive `Requires 1` dans l'`Argument 2`. Si plusiuers argumenst sont requis, séparez-les par des virgules, e.g. `Requires 1,2`.

- `LeftDelim [chaîne]` définit un délimiteur gauche personnalisé (au lieu de `{` ou `[`). Un passage à la ligne dans la sortie peut être indiqué par `<br/>` ;

- `RightDelim [chaîne]` définit un délimiteur droit personnalisé (au lieu de `}` ou `]`). Un passage à la ligne dans la sortie peut être indiqué par `<br/>` ;

- `DefaultArg [chaîne]` définit un argument qui sera inséré si et seulement si aucun argument n'est saisi par l'utilisateur, c'est-à-dire si aucun insert d'argument n'a été spécifié (notez qu'un insert d'argument vide masque le `DefaultArg`). Les arguments multiples doivent être séparés par des virgules ;

- `PresetArg [chaîne]` définit un argument inséré dans tous les cas (seul ou en plus des chaîne saisies par l'utilisateur). Les arguments multiples doivent être séparés par des virgules ;

- `Font` définit la police de caractères utilisée pour le contenu de l'argument, voir 5.3.13 ;

- `FreeSpacing` [*0*, `1`] similaire à la directive des styles de paragraphe, voir 5.3.7.

- `LabelFont` définit la police de caractères utilisée pour l'étiquette, voir 5.3.13 ;

- `Decoration` [*`Classic`*, `Minimalistic, Conglomerate`] précise le style de rendu utilisé pour les cadres d'insertion et les boutons ;

- `AutoInsert [int=0]` force l'insertion de l'argument lorsque le style concerné est sélectionné (si sa valeur est 1).

- `FreeSpacing` [*0*, `1`] similaire à la directive des styles de paragraphe, voir 5.3.7.

- `InsertCotext [int=0]` si positionné à `1`, insère l'argument avec une copie du co-texte, (soit du texte sélectionné, soit tout le paragraphe) comme contenu.

- `PassThru` [*`inherited`*, `true, false`] précise si le contenu de cet argument doit être émis sous forme brute, c'est-à-dire sans interprétation particulière requise par LaTeX. Implicitement, l'état `PassThru` est hérité par l'insert ou le paragraphe auquel l'argument est attaché, `true` et `false` changent le statut pour le seul argument donné.

- PassThruChars[chaîne de caractères] définit des caractères qui doivent être transmis tel quels à la sortie, c'est-à-dire sans traitements particuliers que LaTeX pourrait requérir. Notez bien que, contrairement à PassThru, ceci nécessite une définition explicite pour les arguments, ceux-ci n'héritant pas PassThruChars de leur insert ou format parent.
- IsTocCaption [*0*, 1] émettra le contenu de l'argument dans l'élément correspondant dans la table des matières si positionné à 1. Voir AddToToc.

Implicitement, le texte saisi dans la fenêtre de travail de LyX dans le style considéré est le dernier argument de la commande si le LatexType est Command. Cependant, les arguments préfixés par post: sont émis après cet argument de la fenêtre de travail. Notez que la numérotation des post-arguments repart de 1, de sorte que le premier argument suivant l'argument de la fenêtre de travail est post:1. Les post-arguments sont ignorés dans tout autre LatexType que Command.

Les arguments de liste \items (comme dans \item[toto]) sont préfixés par item: suivi par leur numéro (e. g. Argument item:1).

Enfin, il existe un type particulier d'argument doté du préfixe listpreamble:. Ce n'est pas vraiment un argument, mais il utilise l'interface des arguments (le préfixe est aussi suivi par un nombre, e. g. Argument listpreamble:1). Comme son nom l'indique, il s'applique aux listes comme ListePuces, Énumération, Description, ou Bibliographie. Son contenu sera positionné au début de la liste, avant le premier \item, sur une ligne à part (un endroit normalement inaccessible en LyX). De cette façon, les utilisateurs peuvent insérer des redéfinitions (de longueurs, etc.) pour des listes particulières. Implicitement, ces arguments n'ont pas de délimiteur.

5.3.12. Compteurs

Il faut définir les compteurs (chapitre, figure…) dans la classe elle-même. Les compteurs standard sont définis dans le fichier stdcounters.inc. Si vous cherchez à savoir comment mettre à jour une classe déjà existante, il suffit probablement que vous ajoutiez

```
Input stdcounters.inc
```

à un endroit raisonnable dans la classe.

Mais si vous voulez définir des compteurs à votre guise, vous pouvez le faire. La déclaration d'un compteur doit commencer par

```
Counter <nom>
```

où <nom> est le nom du compteur. Et elle doit se terminer par `End`.
Les paramètres suivants peuvent également être utilisés :

`InitialValue` [`int=1`] positionne la valeur initiale du compteur, valeur à laquelle il sera réinitialisé quand cela sera nécessaire. Normalement, la valeur désirée est la valeur implicite, 1.

`LabelString` [`string=""`] définit comment le compteur s'affiche. Fixer ce paramètre positionne également `LabelStringAppendix` à la même valeur. Les arguments spécifiques suivants peuvent être utilisés :

- `\thecounter` sera remplacé par l'expansion de `LabelString` (ou `LabelStringAppendix`) du compteur `counter`.
- les valeurs du compteur peuvent être exprimées par des macros à la LaTeX `\`*`numbertype`*`{`*`counter`*`}`, où *`numbertype`* peut être : `arabic` : 1, 2, 3,... ; `alph` pour les lettres en bas de casse : a, b, c, ... ; `Alph` pour les lettres en capitales : A, B, C, ... ; `roman` pour la numérotation en bas de casse romaine : i, ii, iii, ... ; `Roman` pour la numérotation capitale romaine : I, II, III.

Si `LabelString` est indéfini, une valeur implicite est construite comme suit : si le compteur a un compteur parent `parent` (défini via `Within`), la chaîne `\theparent.\arabic{counter}` est utilisée ; sinon la chaîne `\arabic{counter}` est utilisée.

`LabelStringAppendix` [`string=""`] est identique à `LabelString`, mais pour les annexes.

`LaTeXName` [`string = ""`] définit le nom du compteur utilisé en LaTeX. (e.g., en LyX, il existe un compteur dénommé 'theorem', mais son nom dans l'exportation LaTeX est 'thm'.)

`PrettyFormat` [`string=""`] définit un format à utiliser pour les références mises en forme utilisant ce compteur. Par exemple, on peut vouloir que les références aux numéros de section apparaissent comme « Section 2.4 ». La chaîne devra contenir « `##` » ou une spécifcation de compteur comme dans `LabelString`. Ceci sera remplacé par la valeur courante du compteur. Ainsi, pour les sections, ce sera : Section `##`, ou peut-être, `\S\arabic{section}` (qui pourrait être affiché comme §2.7).

`RefFormat` [`string`, `string`] est à utiliser avec les références mises en forme, en particulier quand un seul compteur est utilisé avec plusiurs styles. Par exemple le compteur `theorem` est souvent utilisé pour tous les environnements de type théorème : Théorème,

Lemme, etc. Le premier argument donne un préfixe utilisé dans les étiquettes (e.g. « thm » ou « lem »), et le second un chaîne de mise en forme comme pour `LabelString` ou `PrettyFormat`. S'il n'est pas spécifié, alors c'est `PrettyFormat` qui est utilisé.

Within [`string=""`] contient le nom d'un autre compteur : le compteur actuel sera remis à zéro à chaque fois que l'autre augmentera. Par exemple, `sous-section` est numéroté dans `section`.

5.3.13. Description de police

Une description de police ressemble à :

```
Font ou LabelFont ou DefaultFont
 ...
EndFont
```

et les commandes suivantes sont disponibles :

Color [chaîne] voir l'annexe B pour les arguments valides.

Family [*Roman*, Sans, Typewriter]

Misc [chaîne] avec les arguments valides suivants : emph, noun, strikeout, underbar, uuline, uwave, no_emph, no_noun, no_strikeout, no_bar, no_uuline et no_uwave. Chacun de ceux-ci (dés)active l'attribut correspondant. Par exemple, emph active la mise en évidence, et no_emph la désactive. Si ce dernier point vous intrigue, souvenez-vous que le réglage de police du contexte courant est généralement hérité du contexte environnant. De ce fait, no_emph désactiverait la mise en évidence qui était active de toutes façons, par exemple dans un environnement de théorème.

Series [*Medium*, Bold]

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

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

5.3.14. Description du moteur de citation

Les blocs `MoteurCitation`, tels qu'ils ont utilisés dans les fichiers éponymes (voir 5.2.6), définissent les commandes de citation reconnues par un « moteur de citation » spécifique. Un moteur de citation, en LyX, désigne une façon de spécifier le format des citations en utilisant des

nombres, des noms d'auteurs ou des années. Actuellement, LyX reconnaît trois types de moteurs :

1. `default` : la méthode implicite de BibTeX pour mettre en forme les citations, un style numérique simple (e. g., "[1]") ;
2. `authoryear` : les citations style Harvard combinant noms d'auteurs et année de publication (e. g., "Smith et Miller (2017b)")
3. `numerical` : citations numériques étendues incluant l'auteur ou le titre à côté du numéro (e. g., "Smith et Miller [1]")

Les blocs `MoteurCitation` se présentent comme ceci :

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

L'élément suivant `CiteEngine` désigne le moteur. Les lignes suivantes définissent respectivement une commande de citation ou un paradigme de commande de citation reconnu par le moteur. La ligne peut être simplement une commande de citation utilisée à la fois pour désigner la commande LyX et la sortie LaTeX ou peut être plus compliquée pour éclaircir les choses. La syntaxe complète est :

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

- `LyXName` : le nom utilisé dans le fichier `*.lyx`.

  Pour des raisons de portabilité, nous essayons de choisir le même nom pour le commandes de format similaires dans différents paquetages de citation (de ce fait de nombreux noms découlent de natbib, et nous devons choisir un `latexcmd` différent, si le nom de la commande LaTeX est différent).

- `alias` : une liste de commandes (séparées par des virgules) synonymes du `LyXName` donné dans le moteur courant. Ceci facilite le changement de paquetages de citations et de moteurs. `alias` peut être comparé à `ObsoletedBy` dans les définitions de format.

- `latexcmd` : la commande LaTeX effectivement émise.

`Alias` et `latexcmd` sont facultatifs. S'il n'y a pas de`latexcmd`, le `LyXName` sera émis vers LaTeX.

Notez également :

- une capitale indique que la commande a également une forme capitalisée (`\Latexcmd` vs. `\latexcmd`). Ceci force en général la capitalisation des préfixes de nom (*von Goethe* ⇒ *Von Goethe*).
- les crochets `[]` précisent le nombre d'arguments facultatifs (de 0 à 2).
- une étoile * indique qu'il, existe une version étoilée de la commande (`\latexcmd*` vs. `\latexcmd`).

  Implicitement, la version étoilée signifie qu'il faut afficher tous les auteurs, même si la liste devrait être raccourcie par « et al. » du fait de la limite `MaxCiteNames` .

  Si l'étoile a une signification différente pour une commande donnée, ceci peut être spécifié par des crochets anguleux : `<!_stardesc!_stardes`
  On peut préciser au maximum deux mots-clés (sujets à traduction) marqués par le préfixe `!_`. Le premier désigne la chaîne de caractères qui remplace la case à cocher « Tous les auteurs » dans le menu de citation, le second désigne une bulle d'aide optionnelle pour cette case.

  Notez que ces deux mots-clés doivent être définis dans un `CiteFormat` (voir le paragraphe suivant), en enlevant le point d'exclamation, comme ceci :

  ```
  _stardesc Légende d'une commande étoilée
  _stardesctooltip Bulle d'aide pour la case à cocher d'une comm
  ```

- un dollar $ indique que cette commande supporte les « listes de citation qualifiées ». Ceci est une fonctionnalité particulière à Biblatex pour les citations à références multiples, où une pré- ou post-note peut être associée à chaque référence de la liste. Veuillez vous reporter au manuel Biblatex pour les détails.

Si vous souhaitez ajouter une commande `cite` à un moteur de citation (e.g. ajouter une commande particulière fournie par une classe), vous pouvez utiliser `AddToCiteEngine <type de moteur> … End`. Notez que seules les commandes de citation qui n'existe pas encore sont ajoutées.

5.3.15. Description d'une insertion de citation

Les blocs `CiteFormat` servent à décrire comment les citations bibliographiques doivent être affichées, à la fois dans LyX (dans la fenêtre de citation et dans les bulles d'aide, par exemple) et dans le résultat XHTML. Un tel bloc se présente comme suit :

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

ou

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

Dans le premier cas, les différentes lignes définissent l'affichage de l'information correspondant à une article ou à un livre, respectivement, et une telle définition peut être créée pour tout type d'entrée apparaissant dans un fichier BibTEX. LYX définit un format implicite dans le code source qui sera utilisé si aucune définition spécifique n'est fournie. LYX prédéfinit plusieurs formats dans le fichier `stdciteformats.inc`, qui est inclus dans la plupart des classes de document LYX.

Dans le second cas, les lignes définissent comment une commande de citation particulière (dans l'exemple `\cite`, `\citet`) doit être affichée dans la légende d'insert de citation, dans le dialogue de citation ou dans le résultat XHTML. LYX définit de tels formats pour les variantes de style de citation qu'il reconnaît via Document ▷ Paramètres ▷ Bibliographie… dans les fichiers `*.citeengine` qui accompagnent LYX (voir 5.2.6).

Les définitions utilisent un langage simple qui permet de remplacer les clés BibTEX par leurs valeurs. les clés doivent être encloses entre caractères `%`, e.g. : `%author%`. Une définition simple serait par exemple :

```
misc %author%, “%title%”.
```

Ceci imprimerait le nom de l'auteur, suivi d'une virgule, suivi du titre, entre double guillemets, suivi d'un point.

Bien entendu, vous voudrez parfois n'imprimer une clé que si elle existe. Ceci est obtenu par une construction conditionnelle, comme : `{%volume%[[vol. %volume%]]}`. Ceci signifie : si la clé `volume` existe, alors imprimer « vol. » suivi de la clé volume. On peut aussi introduire une clause `else` dans l'expression conditionnelle, comme dans :
`{%author%[[%author%]][[%editor%, ed.]]}`
Ici, la clé `author` est imprimée si elle existe ; sinon la clé `editor` est imprimée, suivi de « , ed. ». Noter que la clé est encore enclose entre caractères `%` ; la clause conditionnelle entière est enclose entre accolades, et les clauses `if` et `else` sont encloses entre double crochets « [[» et «]] ». Il ne peut avoir d'espaces entre ces marqueurs.

En plus des clés d'entrée, quelques clés spéciales peuvent être utilisée pour ces conditions :

- {%dialog%[[true]][[false]]} : traite « true » pour les dialogues et les menus, « false » dans d'autres contextes (fenêtre LyX, export) ;
- {%export%[[true]][[false]]} : traite « true » pour exports et menus, « false » dans d'autres contextes (fenêtre LyX, dialogues) ;
- {%next%[[true]]} : traite « true » si un autre élément suit (e. g., dans une citation à clés multiples), « false » sinon » ;
- {%second%[[true]][[false]]} : traite « true » si c'est le second élément d'une liste, « false » sinon ;
- {%ifstar%[[true]][[false]]} : traite « true » pour une commande de citation étoilée (comme \cite*), « false » si la commande est non étoilée ;
- {%ifentrytype:<type>%[[true]][[false]]} : traite « true » si le type d'entrée courant correspond à <type>, sinon « false » (e.g., dans une définition de citation {%ifentrytype:book%[[ceci est un livre]][[ceci n'est pas un livre]]});
- {%ifmultiple:<authortype>%[[true]][[false]]} : traite « true » si le type d'auteur courant (author, editor etc.) a plusieurs auteurs, « false » sinon (e.g., dans la définition d'une bibliographie : {%ifmultiple:editor%[[eds.]][[ed.]]});
- {%ifqualified%[[true]][[false]]} : traite « true » si la citation courante est est une liste de citation qualifiée (un format spécifique Biblatex pour les citations à références multiples), « false » sinon.

Il a été dit que %author% imprime la clé auteur telle qu'elle est décrite dans le fichier bibliographique. Ce n'est pas nécessairement ce qui est désiré, puisque le résultat pourrait être « Miller, Peter and Smith, Mary and White, Jane » (du fait que « and » est utilisé par BibTeX pour séparer les auteurs). LyX propose donc quelques méthodes pour afficher correctement des listes de noms (qui seront également traduites). Les solutions suivantes sont disponibles :

1. Pour les listes de noms avec prénom et nom, appropriées pour les auteurs/éditeurs d'une entrée bibliographique. La partie <nametype> précise le type de liste requis (e.g. <nametype:author>) :
   - %abbrvnames:<nametype>% : crée une liste abrégée (avec « et al. ») quand MaxCiteNames est atteint.
   - %fullnames:<nametype>% : crée une liste complète (jamais abréviée avec « et al. »).

    - `%forceabbrvnames:<nametype>%` : crée une liste toujours abrégée (avec « et al. ») quel que soit `MaxCiteNames`.

2. Autres listes avec prénom et nom, si l'ordre des nom et prénom diffère dans l'entrée bibliographique (comme avec : « Miller, John : <texte>, in : Mary Smith, ed. : A volume ») :
    - `%abbrvbynames:<nametype>%` : crée une liste abrégée (avec « et al. ») quand `MaxCiteNames` est atteint.
    - `%fullbynames:<nametype>%` : crée une liste complète (jamais abréviée avec « et al. »).
    - `%forceabbrvbynames:<nametype>%` : crée une liste toujours abrégée (avec « et al. ») quel que soit `MaxCiteNames`.

3. Et enfin listes constituées uniquement de noms de famille, comme utilisé dans les étiquettes de citation auteur-année. Elles ne prennent pas de `<nametype>`, mais retournent toujours soit une liste d'auteurs ou, si c'est impossible, une liste d'éditeurs (comme il est fréquent dans les étiquettes auteur-année) :
    - `%abbrvciteauthor%` : crée une liste abrégée (avec « et al. ») quand `MaxCiteNames` est atteint.
    - `%fullciteauthor%` : crée une liste complète (jamais abréviée avec « et al. »).
    - `%forceabbrvciteauthor%` : crée une liste toujours abrégée (avec « et al. ») quel que soit `MaxCiteNames`. .

L'ordre prénom/nom dans les deux premières listes peut être ajusté via ces macros :

- `!firstnameform %surname%, %prename%` (premier auteur dans les listes de type 1)
- `!othernameform %surname%, %prename%` (autres auteurs dans les listes de type 1)
- `!firstbynameform %prename% %surname%` (premier auteur dans les listes de type 2)
- `!otherbynameform %prename% %surname%` (autres auteurs dans les listes de type 2)

Ceci vous permet de créer des nommages tels que « Miller, Peter and Mary Smith : …, in : John Doe and Pat Green, eds. :… ».

Il existe enfin une autre syntaxe possible dans ces définitions, qui se présente comme suit :`{!<i>!}`. Ceci définit une entité d'information utilisée pour créer du « texte enrichi ». De manière évidente, nous ne souhaitons pas exporter des balises HTML en écrivant du texte normal, aussi doivent elles être enclose entre «{!» et «!}».

Deux définitions spéciales sont également disponibles dans un bloc `CiteFormat` Un exemple de la première définition est celle-ci :

```
!quotetitle “%title%”
```

Ceci est une abréviation, ou macro-instruction, et peut être employée comme une clé : `%!quotetitle%`. LyX considérera `%!quotetitle%` exactement comme s'il s'agissait de sa définition. Exprimons par conséquent un *avertissement* évident : n'utilisez pas :

```
!funfun %funfun%
```

ou similaire. LyX ne devrait pas se perdre dans une boucle infinie, mais cela peut prendre un moment avant qu'il s'en sorte.

Le second type de définitions particulières se présente comme ceci :

```
B_pptext pp.
```

Ceci définit un segment de texte susceptible de traduction, ce qui permet de traduire les parties appropriées de la bibliographie ou de la citation. Cette syntaxe peut être incluse dans une définition normale en la considérant comme une clé : `%B_pptext%`. Notez qu'il existe deux façons de traduire : toutes les définitions commençant par `B_`, comme dans l'exemple ci-dessus, seront traduites dans le tampon actif courant (la traduction sera faite dans la langue du document) ; toutes les définitions commençant par un caractère « souligné » seulement seront traduites dans la langue de l'interface. Ceci est la traduction correcte pour les chaînes de caractères qui apparaissent seulement dans les menus ou les boutons, comme celle-ci :

```
_addtobib Add to bibliography only.
```

Plusieurs d'entre elles sont prédéfinies dans `stdciteformats.inc` et les différents fichiers `*.citeengine`. Notez bien que ce ne sont pas des macro-instructions, au sens défini ci-dessus : elles ne seront pas interprétées.

Voici donc un exemple utilisant ces fonctionnalités :

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

Ceci défini une macro qui imprime le nom de l'auteur, suivi d'un virgule, si la clé `author` existe, ou bien imprime le nom de l'éditeur, suivi de `B_edtext` ou de sa traduction (implicitement, apparaîtra « ed. »), si la clé `editor` existe. Notez que ceci est déjà défini dans `stdciteformats.inc`, vous pouvez donc l'utiliser dans vos propres définitions, ou redéfinitions, si vous chargez ce fichier d'abord.

5.4. Directives pour l'exportation XHTML

Comme pour LaTeX ou DocBook, la présentation du résultat HTML créé par LyX est déterminé par les informations de format. En général, LyX produit une présentation implicite raisonnable, et comme indiqué plus haut, il construira même des règles implicites CSS à partir des autres directives de format. Par exemple, LyX essaiera d'utiliser l'information du bloc `Font` du style Chapter pour écrire une CSS qui mettra en forme les titres de chapitres de manière appropriée.

Dans de nombreux cas, vous n'aurez donc sans doute rien à faire du tout pour obtenir un résultat XHTML acceptable pour vos environnements propres, vos inserts personnalisés, et ainsi de suite. Mais dans certains cas vous devrez faire quelque chose, et LyX fournit par conséquent un certain nombre de directives de format qui peuvent être utilisées pour personnaliser le XHTML et les CSS qui sont créés.

Notez qu'il existe deux directives,`HTMLPreamble` et `AddToHTMLPreamble`, qui peuvent apparaître en dehors des déclarations de style et d'insert. Voir 5.3.5 pour les détails sur celles-ci.

5.4.1. Styles de paragraphe

Le type de XHTML que LyX crée pour un paragraphe dépend s'il s'agit d'un paragraphe normal, d'une commande ou d'un environnement, ce qui est déterminé par le contenu de la directive correspondante `LaTeXType`.

Pour une commande ou un paragraphe normal, le résultat XHTML a la forme suivante :

```
<tag attr="value">
<labeltag attr="value">Étiquette</labeltag>
Contenu du paragraphe
</tag>
```

Les balises d'étiquette sont bien entendu omises si le paragraphe n'est pas étiqueté.

Pour un environnement qui n'est pas du type liste, le XHTML prend la forme :

```
<tag attr="value">
<itemtag attr="value"><labeltag at-
tr="value">Étiquette d'environnement</labeltag>Premier pa-
ragraphe.</itemtag>
<itemtag>Second paragraphe.</itemtag>
</tag>
```

Noter que l'étiquette n'est émise que pour le premier paragraphe, comme ce serait le cas pour une théorème, par exemple.

Pour une liste, nous avons une de ces formes :

```
<tag attr="value">
<itemtag attr="value"><labeltag at-
tr="value">Étiquette de liste</labeltag>Premier élé-
ment.</itemtag>
<itemtag attr="value"><labeltag at-
tr="value">Étiquette de liste</labeltag>Second élé-
ment.</itemtag>
</tag>
<tag attr="value">
<labeltag at-
tr="value">Étiquette de liste</labeltag><itemtag at-
tr="value">Premier élément.</itemtag>
<labeltag at-
tr="value">Étiquette de liste</labeltag><itemtag at-
tr="value">Second élément.</itemtag>
</tag>
```

Noter la différence en ce qui concerne l'ordre des balises `labeltag` et `itemtag`. L'ordre obtenu dépend du positionnement de `HTMLLabelFirst` : si la valeur de `HTMLLabelFirst` est fausse (valeur implicite), vous obtiendrez la première forme, avec l'étiquette dans l'élément ; s'il la valeur est vraie, vous obtiendrez la seconde forme, avec l'étiquette en dehors de l'élément.

Les balises et les attributs spécifiques émis pour chaque type de paragraphe peuvent être contrôlés par les directives de format que nous allons décrire. Comme indiqué ci-dessus, cependant, LyX utilise des valeurs implicites raisonnables pour la plupart d'entre elles, vous n'aurez donc pas grand-chose à faire pour produire un résultat XHTML satisfaisant. Pensez aux directives disponibles comme un supplément pour obtenir un réglage à votre convenance.

`HTMLAttr` [chaîne] précise les informations d'attribut à émettre avec la balise principale. Par exemple «`class=`madiv'». Implicitement, LyX émettra

«`class=`nomstyle'», où `nomstyle` est le nom du style en bas de casse, par exemple : chapter. Ceci ne doit contenir *aucune* information de style : utiliser `HTMLStyle` pour cela.

`HTMLClass` [chaîne] définit la calsse CSS à utiliser pour ce paragraphe. Notez que, si le type du paragraphe est une énumération ou une liste à puces, la valeur implicite sera « lyxenum » ou « lyxitem », plus « i », « ii », « iii », ou « iv », en fonction de la profondeur. Ceci peut être surchargé ici, cependant le suffixe ne sera pas ajouté dans ce cas, c'est-à-dire que la classe CSS sera toujours exactement ce qui sera déclaré ici.

`HTMLForceCSS` [*0*, 1] indique s'il faut émettre l'information CSS implicite engendrée par LyX pour ce style, même si une information complémentaire est explicitement émise par `HTMLStyle`. Positionner cette directive à `1` vous permet de modifier ou de compléter la CSS créée, plutôt que de l'écraser complètement. Implicitement `0`.

`HTMLInToc` [`0`,*1*] précise si ce paragraphe (d'habitude une section ou similaire) doit être ajouté à la table des matières. Implicitement vrai, donc il faut le mettre à faux pour les sections étoilées par exemple.

`HTMLItem` [chaîne] définit la balise utilisée pour les paragraphes ou environnements isolés, qui remplace `itemtag` dans les exemples ci-dessus. Vaut implicitement `div`.

`HTMLItemAttr` [chaîne] définit les attributs des balises `itemtag`. Vaut implicitement «`class=`nomstyle_item'». Ceci ne doit contenir *aucune* information de style : utiliser `HTMLStyle` pour cela.

`HTMLLabel` [chaîne] définit la balise utilisée pour les étiquettes de paragraphe et d'élément, qui remplace `labeltag` dans les exemples ci-dessus. Vaut implicitement `span`, à moins que `LabelType` soit `Top_Environment` ou `Centered_Top_Environment`, auquel cas elle vaut implicitement `div`.

`HTMLLabelAttr` [chaîne] définit les attributs de la balise `labeltag`. Vaut implicitement ". Ceci ne doit contenir *aucune* information de style : utiliser `HTMLStyle` pour cela.

`HTMLLabelFirst` [*0*, 1], significatif uniquement pour les environnements de liste, contrôle si la balise d'étiquetage est émise avant ou dans la balise d'élément. Ceci est utilisé par exemple dans l'environnement de description, où l'on veut «`<dt>…</dt><dd>…</dd>`». Vaut implicitement `0` : la balise d'étiquetage est dans la balise d'élément.

HTMLPreamble définit l'information à émettre dans la section <head> quand ce style est utilisé. Ceci pourrait être utilisé par exemple pour inclure un bloc <script> définissant un gestionnaire onclick.

HTMLStyle définit l'information à émettre dans la section <head> quand ce style est utilisé. Ceci pourrait être utilisé par exemple pour inclure un bloc <script> définissant un gestionnaire onclick. Doit être fermé par EndHTMLStyle.

HTMLTag [chaîne] définit la balise utilisé pour l'étiquette principale, qui remplace tag dans les exemples ci-dessus. Vaut implicitement div.

HTMLTitle [*0*, 1] identifie ce style comme celui à utiliser pour créer la balise <title> dans le fichier XHTML. Implicitement fausse. Le fichier stdtitle.inc la positionne à vraie pour l'environnement title.

5.4.2. InsetLayout XHTML

L'exportation XHTML des inserts peut également être contrôlée par l'information résidant dans les fichiers de format[22]. Là encore, LyX essaie de produire implicitement un résultat raisonnable, et il crée des règles CSS implicites. Mais tout peut être personnalisé.

Le résultat XHTML produit par LyX pour un insert a la forme suivante :

```
<tag attr="value">
<labeltag>Étiquette</labeltag>
<innertag attr="value">Contenu de l'insert.</innertag>
</tag>
```

Si l'insert permet de saisir plusieurs paragraphes — c'est-à-dire, si MultiPar est vrai — alors le contenu de l'insert sera exporté également en paragraphes formatés en fonction des styles utilisés pour ces paragraphes (standard, citation et similaire). La balise d'étiquetage est bien entendu omise si la paragraphe n'a pas d'étiquette et, pour le moment, est toujours span. La balise interne est facultative et, implicitement, n'apparaît pas.

Les balises et attributs spécifiques émis pour chaque insert peuvent être contrôlés au moyen des directives de format suivantes :

HTMLAttr [chaîne] précise les informations d'attribut à émettre avec la balise principale. Par exemple «class=`moninsert' onclick=`…'».

[22] Pour le moment, ceci n'est exact que pour les inserts de « texte » (les inserts dans lesquels vous pouvez effectuer une saisie), mais n'est pas exact pour les inserts de « commande » (inserts associés à des boîtes de dialogue).

Implicitement, LyX exportera «`class=`nominsert'», où `nominsert` est le nom LyX de l'insert en bas de casse et avec les caractères non alphanumériques remplacés par des caractères « souligné », par exemple : footnote.

`HTMLForceCSS` [*`0`*, `1`] indique s'il faut émettre l'information CSS implicite engendrée par LyX pour cet insert, même si une information complémentaire est explicitement émise par `HTMLStyle`. Positionner cette directive à `1` vous permet de modifier ou de compléter la CSS créée, plutôt que de l'écraser complètement. Implicitement `0`.

`HTMLInnerAttr` [`chaîne`] fixe l'attribut de la balise interne. Vaut implicitement
«`class=`nominsert_inner'».

`HTMLInnerTag` [`chaîne`] détermine la balise interne, et remplace `innertag` dans les exemples ci-dessus. Implicitement absente.

`HTMLIsBlock` [`0`, *`1`*] indique si cet insert représente un bloc autonome de texte (comme une note de bas de page) ou bien représente du contenu inclus dans le texte environnant (comme une branche). Vaut implicitement `1`.

`HTMLLabel` [`chaîne`] définit une étiquette pour cet insert, pouvant inclure une référence à un compteur. Par exemple, pour une note de bas de page, ce peut être `\arabic{footnote}`. Cette directive est facultative, et n'a pas de valeur implicite.

`HTMLPreamble` définit l'information à émettre dans la section `<head>` quand ce style est utilisé. Ceci pourrait être utilisé par exemple pour inclure un bloc `<script>` définissant un gestionnaire `onclick`.

`HTMLStyle` définit l'information CSS à inclure quand ce style est utilisé. Noter que le contenu sera automatiquement enclos dans un bloc `<style>` créé par la directive, il n'est donc besoin d'inclure que la CSS elle-même.

`HTMLTag` [`chaîne`] définit la balise utilisé pour l'étiquette principale, qui remplace `tag` dans les exemples ci-dessus. La valeur implicite dépend du réglage de `MultiPar` : si `MultiPar` est vrai, la valeur implicite est `div` ; s'il est faux, la valeur implicite est `span`.

5.4.3. Flottants XHTML

Le résultat XHTML pour les flottants peut être contrôlé par les informations de format. Le résultat a la forme suivante :

```
<tag attr="value">
Contenu du flottant.
</tag>
```

La légende, si elle est présente, est un insert séparé et sera émis en tant que tel. Son apparence pourra être contrôlée par la directive InsetLayout pour les inserts de légende.

HTMLAttr [chaîne] précise les informations d'attribut à émettre avec la balise principale. Par exemple «`class=`monflottant' onclick=`…'`». Implicitement, LyX exportera «`class=`float float-typefloat'`», où `typefloat` est le nom LyX de l'insert (tel qu'il est déterminé par la déclaration de flottant, voir 5.3.9) en bas de casse et avec les caractères non alphanumériques remplacés par des caractères « souligné », par exemple : `float-table`.

HTMLStyle définit l'information CSS à inclure quand ce flottant est utilisé. Noter que le contenu sera automatiquement enclos dans un bloc `<style>` créé par la directive, il n'est donc besoin d'inclure que la CSS elle-même.

HTMLTag [chaîne] définit la balise utilisé pour l'étiquette principale, qui remplace `tag` dans les exemples ci-dessus. Vaut implicitement `div`, cette valeur devra rarement être modifiée.

5.4.4. Mise en page de la bibliographie

La bibliographie peut être mise en forme via les blocs `CiteFormat`. Voir 5.3.15 pour les détails.

5.4.5. CSS créés par LyX

Nous avons mentionné plusieurs fois que LyX créera des règles CSS implicites pour les paragraphes et les inserts, fondées sur les autres informations de format fournies. Ici, nous préciserons quelle information est utilisée par LyX et comment.

Pour le moment, LyX engendre tout seul des CSS seulement pour les informations de police, en utilisant les directives `Family`, `Series`, `Shape`, et `Size` spécifiées dans la déclaration `Font` (voir 5.3.13.) La traduction est essentiellement directe et évidente, par exemple «`Family Sans`» devient «`font-family: sans-serif;`». La correspondance entre les tailles en LyX et les tailles en CSS est un peu moins évidente main néanmoins intuitive. Voir la fonction `getSizeCSS()` dans `src/FontInfo.cpp` pour les détails.

5.5. Balisage pour l'exportation DocBook

Comme pour LaTeX ou XHTML, le format de l'exportation DocBook de LyX est contrôlé par les informations de format (*layout*). En général, LyX fournit des données implicites raisonnables ; cependant, la plus grande partie du style est perdu pendant la conversion, du fait que DocBook a une sémantique stricte et ne permet pas la mise en forme. Quand c'est possible, l'information provenant de LyX sera traduite dans les attributs de `role`.

Dans de nombreux cas, vous pouvez n'avoir rien à faire du tout pour obtenir une exportation DocBook pour vos propres environnements, inserts personnels et autres. Mais dans certains cas, vous devrez, et donc LyX fournit un certains nombre de balises que vous pourrez utiliser pour personnaliser le document DocBook exporté.

Les étiquettes sont rarement exportées, du fait qu'elles sont redondantes en DocBook : l'information est incluse dans les balises elles-mêmes, et l'apparition des étiquettes dans le document final (après traitement des fichiers DocBook) est contrôlé par les feuilles de style. Cependant, les étiquettes peuvent ne pas être redondantes, comme dans les listes de définitions : dans ce cas, le terme défini sera l'étiquette.

5.5.1. Styles de paragraphe

Le type de LyX DocBook export pour un paragraphe est différent suivant qu'il s'agit d'un paragraphe normal, d'une commande ou d'un environnement, et est lui-même déterminé par le contenu de la balise LaTeXType.

Pour une commande ou un paragraphe normal, l'exportation DocBook est de la forma suivante :

```
<tag attr>
Contenu du paragraphe.
</tag>
```

Pour un environnement qui n'est pas du type liste, l'exportation DocBook prend cette forme :

```
<tag attr>
<itemtag>Premier paragraphe.</itemtag>
<itemtag>Second paragraphe.</itemtag>
</tag>
```

Pour une liste, le résultat est le suivant :

```
<tag attr>
<itemtag attr>Premier item.</itemtag>
<itemtag attr>Second item.</itemtag>
</tag>
```

Les balises et rôles émis pour chaque type de paragraphe peuvent être fixés via les balises de format qui vont être décrites ci-après. Veuillez noter que, du fait de la nature même de DocBook, il n'existe pas de valeurs implicites raisonnables, et que les valeurs doivent être toujours soigneusement choisies.

DocBookAttr [chaîne] spécifie les informations d'attribut à émettre avec la balise principale, en replaçant « attr » dans l'exemple ci-dessus. Cette information peut être utilisées dans le traitement ultérieur des fichiers DocBook.

DocBookTag [chaîne] définit la balise à utiliser pour cet insert, en remplaçant « tag » dans l'exemple ci-dessus. La valeur implicite est le nom du flottant et doit toujours être redéfinie, du fait que DocBook ne propose pas de balise générique.

DocBookTagType [block, paragraph, inline] définit la politique de passage à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

5.5.2. Politique de passage à la ligne

Pour toutes les balises, il y a trois politiques possibles pour émettre un passage à la ligne (en fonction de l'attribut DocBook*TagType) :

- « block » : les balises ouvrante et fermante sont sur leur propre ligne (i.e. un saut de ligne avant et après la balise ouvrante et fermante). Les éléments typiques sont les flottants. Par exemple :

  ```
  Contenu avant
  <blocktag>
    Contenu du bloc.
  </blocktag>
  Contenu après
  ```

- « paragraph » : les balises ouvrante et fermante sont sur une même nouvelle ligne et un saut de ligne est émis avant la balise ouvrante et après la balise fermante. Les éléments typiques sont les paragraphes et les items de liste. Par exemple :

```
Contenu avant
<paratag>Contenu du paragraphe.</paratag>
Content après
```

- « `inline` » : les balises ouvrante et fermante sont sur le même ligne que le reste du contenu. Aucun saut de ligne n'est émis. Les éléments typique sont les polices. Par exemple :

```
Contenu avant<inlinetag>Contenu du paragraphe.</inlinetag>Contenu apr
```

La valeur implicite est toujours « `block` ».

5.5.3. InsetLayout DocBook

L'exportation DocBook des inserts peut aussi être contrôlée par des informations dans les fichiers de format.

L'exportation LyX DocBook pour un insert a la forme suivante :

```
<wrappertag wrapperattr>
  <tag attr>
    <innertag innerattr>
      Contenu de l'insert.
    </innertag>
  </tag>
</wrappertag>
```

Pour un insert à éléments, elle ressemble plutôt à ceci :

```
<wrappertag wrapperattr>
  <tag attr>
    <innertag innerattr>
      <itemwrappertag itemwrapperattr>
        <itemlabeltag itemattr>
          Étiquette du premier élément.
        </itemtag>
        <itemtag itemattr>
          <itemtag itemattr>
            Contenu du premier élément.
          </itemtag>
        </itemtag>
      </itemwrappertag>
      <itemwrappertag itemwrapperattr>
        <itemlabeltag itemattr>
```

```
            Étiquette du deuxième élément.
          </itemtag>
          <itemtag itemattr>
            <itemtag itemattr>
              Contenu du deuxième élément.
            </itemtag>
          </itemtag>
        </itemwrappertag>
        ...
      </innertag>
    </tag>
  </wrappertag>
```

Si l'insert autorise plusieurs paragraphes — c'est à dire, si `MultiPar` est vrai — alors les contenus de l'insert seront eux-mêmes exportés comme des paragraphes mis en forme en fonction des styles utilisés pour ces paragraphes (standard, citation et autres). La balise interne est facultative et implicitement n'apparaît pas.

Les balises et attributs spécifiques pour chaque insert peuvent être contrôlés via les balises de format qui suivent.

`DocBookAttr` [`chaîne`] précise l'information attribut à émettre avec la balise principale, en remplaçant « `attr` » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

`DocBookInInfo` [`never, always, maybe`] précise si cette balise se retrouve dans la balise `<info>` au commencement du format parent.`never` indique que ce n'est jamais le cas (c'est la valeur implicite, et elle correspond au contenu habituel).`always` indique que c'est toujours le cas (ceci correspond aux métadonnées usuelles) : s'il n'y a pas de balise `<info>` pour le parent, il en **sera** créée **une**. `maybe` indique que la balise pourra peut-être aller dans `<info>` (ce n'est le cas que pour les titres) : s'il n'y a pas de balise `<info>` pour le parent, **aucune** ne sera créée, la balise correspondante sera exportée directement en tant que contenu.

`DocBookItemAttr` [chaîne] précise l'information attribut à émettre avec la balise élément, en remplaçant « `itemattr` » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

`DocBookItemInnerAttr` [chaîne] précise l'information attribut à émettre avec la balise élément interne, en remplaçant « `iteminnerattr` » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

DocBookItemInnerTag [chaîne] définit la balise à utiliser pour la balise d'élément interne à l'intérieur de l'insert, en remplaçant « `iteminnertag` » dans l'exemple ci-dessus. La valeur implicite est `NONE`, indiquant qu'il n'y a pas de balise interne d'élément : le contenu est directement exporté sans elle pour chaque élément. Ce paramètre n'a de sens que lorsque des formats à éléments sont utilisés, comme les listes. La valeur la plus probable est « `para` ».
Quand un élément de liste est divisé par un saut de ligne, la balise interne d'élément sera répétée pour chaque partie du paragraphe, les parties étant séparées par des retours à la ligne.

DocBookItemInnerTagType [`block, paragraph, inline`] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

DocBookItemLabelAttr [chaîne] précise l'information attribut à émettre avec la balise étiquette d'élément à l'intérieur de l'insert, en remplaçant « `itemlabelattr` » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

DocBookItemLabelTag [chaîne] définit la balise à exporter pour la balise étiquette d'élément à l'intérieur de l'insert, en remplaçant « `itemlabeltag` » dans l'exemple ci-dessus. Ce paramètre n'a de sens que lorsque des formats à éléments sont utilisé avec la notion d'étiquette, comme les listes descriptives.

DocBookItemLabelTagType [`block, paragraph, inline`] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

DocBookItemTag [chaîne] définit la balise à utiliser pour la balise d'élément à l'intérieur de l'insert, en remplaçant « `itemtag` » dans l'exemple ci-dessus. La valeur implicite est `NONE`, indiquant qu'il n'y a pas de balise élément. Ce paramètre n'a de sens que lorsque des formats à éléments sont utilisés, comme les listes.

DocBookItemTagType [`block, paragraph, inline`] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

DocBookItemWrapperAttr [chaîne] précise l'information attribut à émettre avec la balise d'encapsulation d'élément, en remplaçant « `itemwrapperattr` » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

DocBookItemWrapperTag [chaîne] définit la valeur à utiliser pour la balise d'encapsulation d'élément à l'intérieur de l'insert, en remplaçant « itemwrappertag » dans l'exemple ci-dessus. La valeur implicite est NONE, indiquant qu'il n'y a pas de balise d'encapsulation d'élément : balise et contenu sont exportés directement pour chaque élément. Ce paramètre n'a de sens que lorsque des formats à éléments sont utilisés, comme les listes.

DocBookItemWrapperTagType [block, paragraph, inline] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

DocBookInnerAttr [chaîne] précise l'information attribut à émettre avec la balise intérieure, en remplaçant « innerattr » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

DocBookInnerTag [chaîne] définit la balise à utiliser pour l'exportation de la balise intérieure dans l'insert, en remplaçant « innertag » dans l'exemple ci-dessus. La valeur implicite est NONE, indiquent qu'il n'y a pas de balise intérieure : le contenu est exporté sans.

DocBookInnerTagType [block, paragraph, inline] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

DocBookSectionTag [chaîne] définit la balise qui correspond à ce type de section. Ce paramètre n'a de sens que pour les éléments de sectionnement (partie, chapitre, section, etc.). La valeur implicite est section, et n'est écrasée que si DocBook utilise quelque chose d'autre pour le sectionnement (typiquement parties et chapitres d'un livre).

DocBookTag [chaîne] définit la balise à utiliser pour cet insert, en remplaçant « tag » dans l'exemple ci-dessus. La valeur implicite est le nom de l'insert et doit toujours être redéfinie, puisque DocBook ne propose pas de balise d'insert générique.

DocBookTagType [block, paragraph, inline] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

DocBookWrapperAttr [chaîne] précise l'information attribut à émettre avec la balise d'encapsulation externe, en remplaçant « wrapperattr » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

DocBookWrapperTag [chaîne] définit la valeur à utiliser pour la balise d'encapsulation de l'insert, en remplaçant « wrappertag » dans l'exemple ci-dessus. La valeur implicite est NONE, indiquant qu'il n'y a pas de balise d'encapsulation globale : balise et contenu sont exportés directement sans elle.

DocBookWrapperTagType [block, paragraph, inline] définit la politique de retour à la ligne pour cette balise, voir la section 5.5.2 pour les détails.

5.5.4. Flottants DocBook

L'exportation DocBook des flottants peut être également contrôlée par les informations de format. L'exportation a la forme suivante :

```
<tag attr>
  Contenu du flottant en tant que  DocBook.
</tag>
```

La légende, s'il y en a une, est un insert à part et sera exportée comme un titre.

DocBookAttr [chaîne] précise l'information attribut à émettre avec la balise principale, en replaçant « attr » dans l'exemple ci-dessus. Cette information peut être utilisée dans le traitement ultérieur des fichiers DocBook.

DocBookTag [chaîne] définit la balise à utiliser pour le flottant, en remplaçant « tag » dans l'exemple ci-dessus. La valeur implicite est le nom du flottant et doit toujours être redéfinie, du fait que DocBook ne propose pas de balise générique pour les flottants.

5.5.5. Mise en forme de la bibliographie

Les bibliographies incluses ne peuvent pas être mises en forme : tous les champs sont toujours exportés dans le format DocBook similaire à une base de données (équivalente à un fichier BibTeX), en utilisant la balise biblioentry.

Lorsque les entrées bibliographiques sont saisies dans le document LyX en tant qu'éléments de Bibliographie, l'utilisateur gère la mise en forme lui-même : il n'y a aucune tentative pour décoder ce que l'utilisateur a écrit, la chaîne de caractères est utilisée telle quelle (via la balise bibliomixed).

6. Insérer un objet externe

Avertissement : cette partie des la documentation n'a pas été mise à jour depuis un certain temps. Nous espérons qu'elle est toujours correcte, mais ce n'est pas garanti.

L'utilisation d'éléments créés par des logiciels extérieurs à LyX est couverte en détail dans le manuel *Objets insérés*. Cette partie du manuel couvre ce qui se passe derrière la scène pour permettre d'écrire un mécanisme d'inclusion pour un nouvel élément.

6.1. Comment fonctionne-t-il ?

L'insertion d'objet externe repose sur le concept de *cadre*[1]. Un cadre définit comment LyX doit s'interfacer avec un type d'objet donné. Tel qu'il est distribué, LyX possède des cadres prédéfinis pour les figures XFig, pour les diagrammes Dia, pour différents formats d'images rastérisées, pour gnuplot, et d'autres. Vous pouvez en consulter la liste avec Insertion ▷ Objet Externe. En outre, il est possible de créer votre propre cadre pour supporter un type donné d'objet. Nous décrirons plus loin en détail ce qu'il faut faire, et nous espérons que vous nous enverrez tous les cadres que vous créerez pour que nous puissions les inclure dans de futures versions de LyX.

Un autre concept de base est qu'il faut faire la distinction entre le fichier d'origine qui sert de point de départ et le fichier transformé qui est inclus avec votre document exporté ou imprimé. Voyons par exemple une figure produite avec `XFig`. L'application XFig elle-même travaille sur un fichier avec l'extension `.fig`. Dans XFig, vous créez et modifiez votre figure. Quand c'est fini, vous enregistrez le fichier `fig`. Quand vous voulez inclure la figure dans votre document, vous invoquez `transfig` pour créer un fichier PostScript qui sera aisément inclus dans votre fichier LaTeX. Dans ce cas, le fichier `.fig` est le fichier d'origine, et le fichier PostScript est le fichier transformé.

Cette distinction est importante pour pouvoir mettre à jour l'objet pendant l'écriture de votre document. En outre, elle vous donne la flexibilité requise pour supporter des formats d'exportation différents. Par

[1] NdT : Traduction de « template » dans ce contexte.

exemple, dans le cas de l'exportation en Ascii, ce n'est pas vraiment une super idée d'inclure la figure en PostScript brut. À la place, vous préférerez soit inclure une référence à la figure, soit essayer un convertisseur graphique vers Ascii pour rendre un résultat final approchant du graphique d'origine. L'insertion d'objet externe vous permet de le faire, car il est paramétré avec les différents formats d'exportation supportés par LyX.

En plus de supporter la génération de fichiers transformés différents selon le format d'exportation, l'insertion d'objet externe travaille en étroite collaboration avec les applications d'édition et de visualisation. Dans le cas d'une figure XFig, vous pouvez invoquer Xfig sur le fichier d'origine d'un simple clic depuis la fenêtre d'objet externe de LyX, et aussi visualiser le fichier transformé PostScript avec Ghostview d'un autre clic. Il n'y a plus à se bagarrer avec la ligne de commande ou avec des explorateurs de fichiers pour localiser et manipuler le fichier d'origine et le fichier transformé. De cette façon, vous pouvez enfin profiter à plein des nombreuses applications différentes qui servent à la production de documents, et serez finalement plus efficaces.

6.2. Le fichier de configuration d'un cadre externe

Il est assez facile de définir de nouveaux cadres externes dans LyX. Cependant, sachez que si vous le faites de façon négligente, vous introduirez *sûrement* une faille de sécurité facilement exploitable. Avant de commencer, lisez donc plus bas ce qui concerne la sécurité (97).

Ceci dit, nous vous encourageons à créer des cadres intéressants et à nous les proposer.

Les cadres externes sont définis dans les fichiers`*.xtemplate` qui se trouvent dans le répertoire `LyXDir/lib/xtemplates/`. Chaque cadre est défini dans un fichier spécifique[2]. Vous pouvez mettre votre propre version dans `MonRép/xtemplates/` ou bien copier un fichier existant dans ce répertoire et le modifier.

Un fichier de configuration de cadre typique se présente comme ceci :

```
Template XFig
GuiName "XFig: $$AbsOrRelPathParent$$Basename"
HelpText
An XFig figure.
```

[2] NdT : Nous rappelons que «cadre» est la traduction de «template» dans ce contexte de l'objet externe.

```
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
```

Comme vous pouvez le constater, le cadre est inséré entre `Template` … `TemplateEnd`. Il contient un en-tête spécifiant quelques réglages généraux et, pour chaque format primaire de document reconnu, une section `Format` … `FormatEnd`.

6.2.1. L'en-tête de cadre

`AutomaticProduction true|false` indique si le fichier représenté par le cadre doit être créé par LyX. Cette directive doit apparaître une et une seule fois.

`FileFilter <patron>` précise un patron global utilisé dans l'échange dans la spécification des fichiers pour filtrer les fichiers désignés. S'il y a plus d'un suffixe possible (e.g. pour tgif, `.obj` et `.tgo`), utiliser une syntaxe comme `"*.{obj,tgo}"`. Cette directive doit apparaître une et une seule fois.

`GuiName <nom_interface>` fixe le texte apparaissant dans le bouton. Cette directive doit apparaître une et une seule fois.

`HelpText <texte> HelpTextEnd` décrit le texte d'aide utilisé dans la fenêtre de dialogue « Objet externe ». Fournir suffisamment d'informations pour que l'utilisateur comprenne ce que le cadre peut faire pour lui. Cette directive doit apparaître une et une seule fois.

`InputFormat <format>` indique le format du fichier original. Ce doit être le nom d'un format reconnu par LyX (voir 3.1). Utiliser "*"si le cadre peut gérer des fichiers originaux dans plus d'un format. LyX tentera d'analyser le fichier pour en déduire le format dans ce cas. Cette directive doit apparaître une et une seule fois.

`Template <id>` est un nom original pour le cadre. Il ne soit pas contenir de macros de substitution (voir ci-après).

`Transform Rotate|Resize|Clip|Extra` spécifie quelles transformations sont reconnues par ce cadre. Elle peut apparaître zéro ou plusieurs fois. Elle active les options correspondantes dans la fenêtre de dialogue. Chaque directive `Transform` doit avoir soit une directive `TransformCommand` correspondante, soit une directive `TransformOption` dans la section `Format`. Sinon, la transformation ne sera pas reconnue par ce cadre.

6.2.2. La section Format

`Format LaTeX|PDFLaTeX|PlainText|DocBook|XHTML` décrit le format de document primaire relevant de cette définition de format. Tous les cadres n'ont pas de représentation raisonnable dans tous les formats de fichier documentaire. Définissez cependant une section `Format` pour tous les formats, et utilisez un texte bidon lorsqu'il n'existe pas de représentation. Ainsi vous verrez au minimum une référence à l'objet externe dans le document exporté.

`Option <nom> <valeur>` définit une macro additionnelle `$$<nom>` pour la substitution par `Product`. `<valeur>` peut elle-même contenir des macros de substitution. L'avantage par rapport à l'usage direct de `<valeur>` dans `Product` est que la valeur substituée par `$$<nom>` est nettoyée de sorte qu'elle est un argument facultatif valide dans le format documentaire. Cette directive doit apparaître une et une seule fois.

`Product <texte>` spécifie le texte inséré dans le document exporté. C'est en fait la directive la plus importante et elle peut être assez compliquée. Cette directive doit apparaître une et une seule fois.

`Preamble <nom>` spécifie un élément de préambule qui sera inclus dans le préambule LaTeX. Il doit être défini par `PreambleDef` … `PreambleDefEnd`. Cette directive doit apparaître une et une seule fois.

`ReferencedFile <format> <nom_fichier>` désigne les fichiers créés par la procédure de conversion et requis par un format d'exportation particulier. Si le nom d'un fichier est relatif, il est interprété relativement au document maître. Cette directive peut apparaître zéro ou plusieurs fois.

`Requirement <paquetage>` désigne le nom d'un paquetage LaTeX requis. Le paquetage est inclus via `\usepackage{}` dans le préambule LaTeX. Cette directive peut apparaître zéro ou plusieurs fois.

`TransformCommand Rotate RotationLatexCommand` spécifie que la commande LaTeX native doit être utilisée pour effectuer les rotations. Cette directive peut apparaître une fois ou pas du tout.

`TransformCommand Resize ResizeLatexCommand` spécifie que la commande LaTeX native doit être utilisée pour effectuer les changements de taille. Cette directive peut apparaître une fois ou pas du tout.

TransformOption Rotate RotationLatexOption spécifie que les rotations sont effectuées via un paramètre facultatif. Cette directive peut apparaître une fois ou pas du tout.

TransformOption Resize ResizeLatexOption spécifie que les changements de taille sont effectuées via un paramètre facultatif. Cette directive peut apparaître une fois ou pas du tout

TransformOption Clip ClipLatexOption spécifie que les recadrages sont effectuées via un paramètre facultatif. Cette directive peut apparaître une fois ou pas du tout.

TransformOption Extra ExtraLatexOption spécifie qu'un paramètre supplémentaire facultatif est utilisé. Cette directive peut apparaître une fois ou pas du tout.

UpdateFormat <format> spécifie le format de fichier du fichier converti. Ce doit être le nom d'un format reconnu par LyX (voir la fenêtre de dialogue Outils ▷ Préférences ▷ Gestion des fichiers ▷ Format de fichier). Cette directive doit apparaître une et une seule fois. Si le format du fichier résultant est PDF, vous devez spécifier le format `pdf6`, qui est le format PDF utilisé pour insérer des graphiques. Les autres formats PDF définis servent aux exportations du document.

UpdateResult <nom_fichier> spécifie le nom du fichier converti. Le nom de fichier doit être absolu. Cette directive doit apparaître une et une seule fois.

6.2.3. Définitions du préambule

La configuration du cadre externe peut contenir des définitions additionnelles de préambule incluses dans `PreambleDef` … `PreambleDefEnd`. Elle peuvent être utilisées par les cadres dans la section `Format`.

6.3. Le mécanisme de substitution

Quand la fenêtre d'objet externe invoque un programme externe, il le fait selon la commande définie dans le fichier de configuration des cadres. Ces commandes peuvent contenir diverses macros qui sont interprétées avant l'exécution. Les commandes sont toujours exécutées dans le répertoire du document.

Ainsi, quand un objet externe est affiché, un mécanisme de substitution génère son nom, et la plupart des autres directives reconnaissent la substitution de manière similaire.

Les macros disponibles sont les suivantes :

$$AbsOrRelPathMaster est le est le chemin d'accès, absolu ou relatif au document maître LyX.

$$AbsOrRelPathParent est le est le chemin d'accès, absolu ou relatif au document LyX.

$$AbsPath est le chemin d'accès absolu.

$$Basename est le nom du fichier sans chemin d'accès et sans suffixe.

$$Contents(«nom_fichier.ext») la macro dépliera le contenu du fichier de nom `nom_fichier.ext`.

$$Extension est le suffixe (point inclus).

$$pngOrjpg sera la chaîne « jpg » si le fichier est en format JPEG, sinon sera la chaîne « png ». Ceci est utile pour éviter des conversions inutiles pour les formats de sortie admettant à la fois les types PNG et JPEG. Le modèle prédéfini « Image tramée » utilise cette macro avec le moteur pdfTeX.

$$FName est le nom du fichier spécifié dans la fenêtre d'objet externe. Il est soit absolu, soit relatif au document LyX.

$$FPath la partie « chemin d'accès » de `$$FName` (absolu ou relatif au document LyX).

$$RelPathMaster est le chemin d'accès relatif au document maître LyX.

$$RelPathParent est le chemin d'accès relatif au document LyX.

$$Sysdir désigne le chemin absolu du répertoire système. Typiquement utilisé pour pouvoir trouver les différents scripts auxiliaires fournis avec LyX.

$$Tempname est le nom et le chemin absolu d'un fichier temporaire qui sera automatiquement effacé quand le document sera fermé, ou l'objet externe effacé.

Toutes la macros de chemin contiennent un séparateur de répertoire final, vous pouvez donc construire par exemple le nom de fichier absolu avec `$$AbsPath$$Basename$$Extension`.

Les macros ci-dessus sont substituées dans toutes les directives sauf indiqué. La directive `Product` reconnaît également les substitutions suivantes si elles sont autorisées par les directives `Transform` et `TransformCommand` :

$$ResizeFront partie initiale de la commande de changement de taille.

$$ResizeBack partie finale de la commande de changement de taille.

$$RotateFront partie initiale de la commande de rotation.

$$RotateBack partie finale de la commande de rotation.

La valeur de la chaîne de la directive `Option` reconnaît également les substitutions suivantes si elles sont autorisées par les directives `Transform` et `TransformOption` :

$$Clip option de recadrage.

$$Extra option supplémentaire.

$$Resize option de changement de taille.

$$Rotate option de rotation.

Vous pouvez vous demander pourquoi les macros de chemin d'accès sont aussi nombreuses. Il y a deux raisons principales :

1. les chemins d'accès relatifs et absolus doivent rester relatifs et absolus, respectivement. Les utilisateurs peuvent avoir des raisons de préférer l'un ou l'autre. Les chemins relatifs sont utiles pour des documents portables sur différentes machines, par exemple. Les chemins absolus peuvent être requis par certains logiciels ;

2. LaTeX traite les chemins d'accès relatifs différemment de LyX et d'autres logiciels dans des fichiers inclus encapsulés. Pour LyX, par exemple, un chemin relatif est toujours relatif au document qui contient le nom de fichier. Pour LaTeX, il est toujours relatif au document maître. Ces deux définitions sont identiques si vous n'avez qu'un seul document., mais diffèrent si vous avez un document maître qui inclut des sous-documents. Ceci signifie que les chemins relatifs doivent être transformés pour être transmis à LaTeX. Heureusement LyX fait ceci automatiquement si vous choisissez les macros de substitution correctes.

Donc, quelle macro faut-il choisir dans une nouvelle définition de cadre ? La règle n'est pas difficile :

- utiliser `$$AbsPath` si un chemin absolu est requis ;
- utiliser `$$AbsOrRelPathMaster` si la chaîne substituée est du type entrée pour LaTeX ;
- sinon utiliser `$$AbsOrRelPathParent` pour préserver le choix de l'utilisateur.

Des cas spéciaux existent pour lesquels cette règle n'est pas valable, et par exemple lorsque des chemins relatifs sont requis, mais elle fonctionne correctement d'habitude. Un exemple de cas tordu est la directive `ReferencedFile latex "$$AbsOrRelPathMaster$$Basename.pstex_t"` dans le cadre Xfig indiqué ci-dessus : il n'est pas possible d'utiliser le nom absolu parce que la copieur pour les fichiers `.pstex_t` nécessite le nom relatif pour réécrire le contenu du fichier.

6.4. La question de la sécurité

L'insertion d'objet externe crée une interface avec un paquet de programmes externes et le fait de manière automatique, nous devons donc en examiner les conséquences en matière de sécurité. En particulier, comme vous avez la possibilité d'inclure vos propres noms de fichier ou de paramètres et qu'ils sont interprétés pour former une commande, il semble possible de créer un document malveillant qui exécute des commandes indésirables quand un utilisateur le visualise ou l'imprime. C'est quelque chose que nous voulons absolument éviter.

Cependant, comme les commandes du programme externe sont définies seulement dans le fichier de configuration des cadres, il n'y a pas de problème de sécurité si LyX est configuré correctement avec seulement des cadres sûrs. Ceci parce que les programmes externes sont invoqués via l'appel système `execvp` et non via l'appel système `system`. Il n'est donc pas possible d'exécuter des commandes arbitraires via le shell à partir du nom de fichier ou des paramètres.

Ceci implique aussi qu'il y a des restrictions dans les commandes définissables dans un cadre d'objet externe. En particulier, les pipes et les redirections ne sont pas disponibles. C'était obligatoire pour que LyX reste sûr. Si vous voulez utiliser quelques-unes des fonctions du shell, vous devez écrire un script sûr pour le faire de manière contrôlée, puis définir ce script comme commande.

Il est possible de créer un cadre qui interagit directement avec le shell, mais comme cela permettrait à un utilisateur malveillant d'exécuter des commandes arbitraires en écrivant des noms de fichier ou des paramètres bien choisis, nous vous recommandons d'utiliser en général des scripts sûrs qui fonctionnent avec l'appel système `execvp` d'une manière contrôlée. Bien sûr, si vous êtes dans un environnement dans lequel vous avez confiance, il est tentant de ne pas se fouler et d'utiliser des scripts shell ordinaires. Si vous faites cela, sachez que vous allez *vraiment* créer dans votre système une faille de sécurité facile à exploiter. Il est évident que de tels cadres non sécurisés ne seront jamais inclus dans la distribution standard de LyX, même si nous encourageons les gens à nous proposer de nouveaux cadres dans la tradition

du logiciel libre. Mais LyX tel que vous vous le procurez par les voies officielles de distribution ne contiendra jamais de cadres non sécurisés.

L'insertion d'objet externe est un outil très puissant, et vous devez faire attention à ne pas mettre en péril la sécurité avec cette puissance. Une légère erreur dans une seule ligne d'un script apparemment inoffensif peut ouvrir la voie à d'énormes problèmes. Si vous n'avez pas une compréhension claire de ce qui est en jeu, nous vous recommandons de consulter un professionnel de la sécurité, ou de contacter l'équipe de développement de LyX si vous vous demandez si votre cadre est sûr ou non. Faites le avant de l'utiliser dans un environnement que vous ne contrôlez pas.

A. Liste des fonctions utilisables dans les styles

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

B. Noms des couleurs disponibles utilisables dans les styles

Les couleurs énumérées ci-après sont les couleurs standard et celles que vous pouvez régler dans les préférences LyX.

B.1. Fonctions couleurs

Les mots-clés suivants ne sont pas des couleurs, mais plutôt des actions sur la définition des couleurs.

ignore couleur ignorée

inherit couleur héritée

none pas de couleur particulière – réinitialisation ou couleur implicite

B.2. Couleurs statiques

Ces couleurs ne peuvent pas être personnalisées. Merci de ne pas utiliser ces couleurs dans les définitions de format, du fait qu'elles fonctionneront mal avec certains thèmes (comme les thèmes sombres).

black

white

blue

brown

cyan

darkgray

gray

green

lightgray

lime

magenta

olive

orange

pink

purple

red

teal

violet

yellow

B.3. Couleurs dynamiques

Ces couleurs sont affectées à différents éléments dans Outils ▷ Préférences :

added_space espace ajoutée

addedtext texte ajouté

appendix marque d'annexe

background fond

bottomarea zone du bas

branchlabel étiquette de branche

buttonbg fond de bouton d'insert

buttonframe bordure de bouton d'insert

buttonhoverbg fond du bouton d'insert (pointé)

changebar barre de modification

changedtextauthor1 texte modifié auteur 1

changedtextauthor2 texte modifié auteur 2

changedtextauthor3 texte modifié auteur 3

changedtextauthor4 texte modifié auteur 4

changedtextauthor5 texte modifié auteur 5

collapsibletext texte d'insert repliable

collapsibleframe cadre d'insert repliable

command insert de commande

commandbg fond d'insert de commande

commandframe cadre d'insert de commande

comment étiquette de commentaire

commentbg fond de commentaire

cursor curseur

deletedtext texte supprimé

deletedtextmodifier modificateur de texte supprimé

depthbar barre de profondeur

eolmarker marqueur de fin de ligne

error erreur LaTeX

footlabel étiquette de note de bas de page

foreground foreground color

graphicsbg fond graphique

greyedoutbg fond d'insert grisé

greyedoutlabel étiquette d'insert grisé

greyedouttext texte d'insert grisé

indexlabel étiquette d'index

inlinecompletion complétion en ligne

insetbg fond d'insert

insetframe cadre d'insert

language langue étrangère

latex texte LaTeX

listingsbg fond de listing

marginlabel étiquette de note en marge

math texte mathématique

mathbg fond mathématique

mathcorners cadre mathématique (non pointé)

mathframe cadre mathématique (pointé)

mathline ligne mathématique

mathmacrobg fond de macro mathématique

mathmacroblend macro mathématique désactivée

mathmacroframe cadre de macro mathématique

mathmacrohoverbg fond dynamique de macro mathématique

mathmacrolabel étiquette de macro mathématique

mathmacronewarg macro mathématique : nouveau paramètre

mathmacrooldarg macro mathématique : ancien paramètre

newpage saut de page

nonunique_inlinecompletion complétion en ligne (choix multiples)

note étiquette de note

notebg fond de note

pagebreak saut de page / saut de ligne

paragraphmarker marqueur de paragraphe

phantomtext texte d'insert fantôme

preview aperçu

previewframe cadre d'aperçu

regexpframe cadre d'expression régulière

scroll color that indicates when a row can be scrolled

selection sélection (fond)

selectiontext sélection (texte)

shadedbg boîte ombrée

special caractère spécial

tabularline ligne de tableau

tabularonoffline ligne on/off de tableau

textlabel1 couleur 1 des étiquettes de format et d'inserts personnalisés

textlabel2 couleur 2 des étiquettes de format et d'inserts personnalisés

textlabel3 couleur 3 des étiquettes de format et d'inserts personnalisés

urllabel étiquette d'URL

urltext texte d'URL