paquet conditext

Jean-Louis Brabant

Version 1.5 du 1 juin 2020

Résumé

Ce paquet permet de définir et gérer du contenu conditionnel (multilingue ou autre) dans un document source LaTeX, sous la forme de commandes d'emploi aisé, et selon un paramétrage minimal.

Table des matières

I Guide de l'utilisateur/référence des commandes 3

1 Généralités 3

1.1 Principe général 3

1.2 Déclaration du paquet/option 3

1.3 Modalités d'utilisation des commandes 4

2 Commandes relatives à la gestion de contenu conditionnel 5

2.1 Saisie des formes substituables 5

2.2 Définition d'un espace de condition 8

2.3 Création d'un domaine de condition 11

2.4 Création d'une propriété de condition 13

2.5 Définition d'un domaine de condition implicite 15

3 Commandes relatives à la mini-gestion conditionnelle 16

3.1 Saisie des formes substituables pour une mini-gestion conditionnelle 16

3.2 Redéfinition des propriétés de condition pour une mini-gestion conditionnelle 18

3.3 Définition d'une espace de condition pour une mini-gestion conditionnelle 21

II Tutoriel : exemples commentés 22

1 Exemple commenté n° 1 : un document en version multilingue 23

2 Exemple commenté d'utilisation n° 2 : un document à gestion conditionnelle multiple 31

3 Exemple commenté d'utilisation n° 3 : une mini-gestion conditionnelle 37

3.1 37

3.2 41

Introduction

Ce paquet fournit quelques facilités, sous la forme d'une petite dizaine de commandes, pour définir et gérer du contenu conditionnel dans un document source LaTeX.

Un *contenu conditionnel*, au sens où ce paquet l'entend et le gère, est un texte (y compris sous forme de formules mathématiques ou autres) et/ou un élément graphique (diagramme, figure, image...) se présentant sous des formes pouvant se substituer les unes aux autres – appelées *formes substituables* dans cette documentation – qui, en fonction d'un test de condition, figurent ou non dans le document généré.

L'une des gestions de contenu conditionnel les plus courantes est d'ordre multilingue, mais il peut aussi s'agir d'éditions « versionnées », de niveaux de services, de niveaux de confidentialité etc.

Comment exploiter cette documentation. Pour une toute première utilisation du paquet conditext, il est fortement conseillé, après un survol de cette documentation, de se reporter directement à la partie *Tutoriel : exemples commentés*. L'on y apprend à utiliser les différentes commandes disponibles, dans des contextes variés, avec des conseils de bonne pratique qui en permettront par la suite une exploitation optimisée.

Une fois en mesure d'utiliser les principales ressources fonctionnelles du présent paquet, l'on pourra, avec profit et à loisir, consulter la partie *Guide de l'utilisateur/référence des commandes* pour toutes précisions utiles concernant les différentes commandes.

Première partie

Guide de l'utilisateur/référence des commandes

1 Généralités

1.1 Principe général

La philosophie du paquet conditext repose sur les notions respectives de domaine de condition (« condition field » en anglais), de propriété de condition (« condition property ») et d'espace de condition (« condition space »).

Avec conditext, dans un document source, toute forme substituable est identifiée par un domaine de condition et une propriété de condition.

Le *domaine de condition* est une thématique fonctionnelle qui permet de regrouper des formes substituables pour une même gestion conditionnelle.

La *propriété de condition* est une caractérisation fonctionnelle propre à chaque forme substituable d'une même domaine de condition.

L'*espace de condition* permet de désigner la ou les formes substituables qui doivent figurer dans le document généré. L'on définit un espace de condition en spécifiant un domaine de condition et une propriété de condition propres à correspondre à une ou plusieurs formes substituables.

1.2 Déclaration du paquet/option

L'on déclare le paquet conditext par l'instruction `\usepackage` (ou `\requirespackage` s'il s'agit d'intégrer conditext dans un paquet personnalisé).

Pour information, conditext fait usage des paquets xifthen et xparse.

Option. Dans sa déclaration, conditext accepte une option, sous la forme :

```
\usepackage[defaultdisplayall]{conditext}
```

Par défaut (déclaration du paquet sans option), tant qu'aucun espace de condition n'est défini, aucune forme substituable quelle qu'elle soit, ne figure dans le document généré. L'option `defaultdisplayall` permet de modifier ce comportement : lorsqu'elle est spécifiée, si aucun espace de condition n'est défini, toute forme substituable figure dans le

document généré. Pour retrouver le comportement d'origine, il suffit de retirer l'option dans la déclaration du paquet.

1.3 Modalités d'utilisation des commandes

Le paquet conditext fournit une petite dizaine de commandes, décrites en détail dans les pages qui suivent, et qui se répartissent en deux grandes catégories selon leur nature et leurs fonctionnalités :

L'environnement `conditext` (cf. page ci-contre), ainsi que les commandes `\setcondispace` (cf. page 8), `\newcondifield` (cf. page 11), `\newcondiprop` (cf. page 13) et `\setimplicitcondifield` (cf. page 15) sont destinés à une gestion conditionnelle à l'échelle multi-paragraphique (de plusieurs dizaines à plusieurs milliers de signes), appelée désormais en tant que telle *gestion de contenu conditionnel*.

Les commandes `\miniconditext` (cf. page 16), `\setminicondispace` (cf. page 16) et `\setminicondiprop-` (cf. page 18) sont destinées à une gestion conditionnelle à l'échelle paragraphique ou intra-paragraphique (mots, groupes de mots), appelée désormais *mini-gestion conditionnelle*.

Complémentarité des commandes. Dans un document-source, une mini-gestion conditionnelle peut être mise en oeuvre de manière indépendante, ou conjointement à une gestion conditionnelle ; d'une certaine manière, l'une et l'autre sont *complémentaires*.

Dans le cas d'une gestion conditionnelle multilingue, l'environnement `conditext` et la commande `\setcondispace` sont directement exploitables, sans nécessiter de préalable d'aucune sorte. Pour toute autre gestion de contenu conditionnel, avant d'utiliser ces commandes, il est nécessaire de procéder aux créations des domaines de condition et propriétés de condition nécessaires, au moyen des commandes `\newcondifield` et `\newcondiprop`.

Emplacement des instructions. De manière générale, les intructions `\setcondispace`, `\newcondifield`, `\newcondiprop`, `\setimplicitcondifield`, `\setminicondispace` et `\setminicondiprop-` devront être placées dans le préambule.

Toute instruction `conditext` sera placée dans le corps même du document source.

Une instruction `\miniconditext` pourra, selon les cas, être placée soit dans le préambule, soit dans le corps du document source.

2 Commandes relatives à la gestion de contenu conditionnel

2.1 Saisie des formes substituables

Tout contenu que l'on souhaite gérer comme forme substituable doit être saisi dans un environnement `conditext`. Cet environnement – éponyme du présent paquet – s'utilise sous la forme suivante :

`\begin{conditext}`
`\end{conditext}`

`\begin{conditext}[`⟨*domaine de condition*⟩`]{`⟨*propriété de condition*⟩`}[*]`
⟨*text*⟩
`\end{conditext}`

2.1.1 Arguments et syntaxe

Les arguments ⟨*domaine de condition*⟩ et ⟨*propriété de condition*⟩ sont à renseigner l'un et l'autre avec une valeur de type alphabétique, sans caractères accentués ou à diacritiques, et sans espaces.

Concernant l'argument `*`, cf. page suivante.

Il est indispensable que ⟨*domaine de condition*⟩ et ⟨*propriété de condition*⟩ aient été au préalable déclarés – nous dirons désormais *créés* – (commandes `\newcondifield` et `\newcondiprop`, cf. respectivement page 11 et page 13) ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation.

En outre, la propriété de condition spécifiée doit avoir été impérativement créée avec un domaine de rattachement (cf. page 13) identique à celui précisé en argument ⟨*domaine de condition*⟩ ; si tel n'est pas le cas, un message d'erreur sera sera déclenché lors de la compilation.

Si les arguments ⟨*domaine de condition*⟩ ou ⟨*propriété de condition*⟩ sont conjointement laissés vides (`[]` et `{}`), un message d'erreur sera déclenché lors de la compilation.

Par défaut, dans un contexte de gestion de contenu conditionnel *multilingue*, l'argument ⟨*propriété de condition*⟩ n'a pas besoin d'être renseigné (cf. page 7).

2.1.2 Utilisation

Un environnement `conditext` est, à l'instar de l'environnement `document`, d'amplitude large et peut contenir de plusieurs dizaines à plusieurs milliers de signes. Tous les aspects propres à l'organisation et à la composition d'un texte sous LaTeX (divisions, paragraphes, listes etc.) y sont laissés à l'appréciation de l'utilisateur.

Dans un document source, il y aura autant d'environnments `conditext` que souhaité. Les propriétés de condition et/ou les domaines de condition spécifiés pourront ou non être identiques, selon les besoins (plu-

sieurs formes substituables appartenant à une même thématique, contenu conditionnel fragmenté et réparti à différents endroits du document source etc.).

Environnements imbriqués. Les environnements `conditext` peuvent être imbriqués, à volonté, et autoriser ainsi diverses « profondeurs » de condition, sur la base de domaines de condition différenciés (à savoir qu'imbriquer des environnements `conditext` de domaines de condition identiques sera inévitablement source d'effets inattendus ou indésirables).

La sémantique d'imbrication et son niveau de *profondeur* sont laissés à l'appréciation de l'utilisateur.

Dans le cas de contenus conditionnels imbriqués, les espaces de condition présentent certaines particularités de comportement (cf. page 9).

2.1.3 Argument étoile

L'environnement `conditext` accepte une étoile en troisième argument sous la forme :

`\begin{conditext}[`⟨*domaine de condition*⟩`]{`⟨*propriété de condition*⟩`}*`
⟨*text*⟩
`\end{conditext}`

Atypique de par sa position en regard de l'usage habituel de l'étoile dans les commandes et environnements LaTeX, l'argument `*` affecte ponctuellement la prise en compte du domaine de condition et de la propriété de condition dans un espace de condition et modifie ainsi la logique d'affichage de la forme substituable correspondante dans le document généré.

Plus précisément, l'étoile permet : soit de forcer l'inclusion de la forme substituable dans le document généré lorsqu'elle n'est pas désignée par un espace de condition ; soit à l'inverse, de forcer l'exclusion de la forme sustituable dans le document généré lorsqu'elle est désignée par un espace de condition.

Cet argument est prévu pour une gestion individualisée « au plus près » des formes substituables et son usage doit être en principe limité à des situations appropriées.

Il est possible – même si cette pratique est déconseillée car peu régulière (et encore moins efficace dans le cas d'un document source comportant de nombreuses instructions `conditext`) –, de ne pas définir d'espace de condition et d'utiliser uniquement l'argument `*` pour forcer directement l'affichage ou le non affichage des formes substituables dans le document généré (le résultat étant inversé selon que le paquet conditext est déclaré sans ou avec l'option `defaultdisplayall`).

2.1.4 Contenu conditionnel multilingue

Par défaut, lorsqu'un environnement `conditext` est destiné à gérer un contenu conditionnel multilingue, l'argument ⟨*domaine de condition*⟩ n'a pas à être spécifié. Il s'agit en fait d'un domaine de condition prédéfini, intitulé `language`, pris en compte implicitement. L'on peut ainsi formuler une instruction `conditext` à sémantique linguistique sous la forme :

`\begin{conditext}{`⟨*propriété de condition*⟩`}[*]`
⟨*text*⟩
`\end{conditext}`

Ce comportement par défaut est cependant modifié dès lors qu'un domaine de condition implicite est défini par l'utilisateur (commande `\setimplicitcondifield`, cf. page 15). il est alors nécessaire de spécifier *explicitement* `language` comme domaine de condition dans toute instruction `conditext` devant gérer un contenu conditionnel multilingue. Ceci s'applique également à la commande `\setcondispace`, cf. page suivante.

Prise en charge par babel. L'argument ⟨*propriété de condition*⟩ doit être impérativement renseigné avec l'une des appellations linguistiques en vigueur dans le paquet babel. Le paquet conditext dispose des propriétés de condition prédéfinies suivantes :

`albanian`, `american`, `arabic`, `armenian`, `bosnian`, `brazilian`, `breton`, `canadian`, `cantonese`, `catalan`, `chinese`, `croatian`, `czech`, `danish`, `dutch`, `english`, `esperanto`, `estonian`, `finnish`, `french`, `georgian`, `german`, `greek`, `hebrew`, `hungarian`, `icelandic`, `irish`, `italian`, `japanese`, `lithuanian`, `luxemburgish`, `macedonian`, `mexican`, `norsk`, `occitan`, `portuguese`, `romanian`, `romansh`, `russian`, `sanskrit`, `serbian`, `slovenian`, `spanish`, `swedish`, `tibetan`, `turkish`, `ukrainian`, `vietnamese`.

Les propriétés de condition de sémantique linguistique sont plus que de simples « étiquettes » ; elles permettent en fait de prendre en charge, pour chaque forme substituable saisie, le support linguistique spécifique – notamment la gestion des césures– disponible dans paquet babel.

Il est de ce fait indispensable que babel soit déclaré dans le préambule, accompagné des langues susceptibles d'être spécifiées en propriétés de condition (se reporter à la documentation babel pour plus de détails sur la déclaration du paquet et des langues). Les commandes du paquet conditext ne prennent pas en compte la précédence de langue propre à babel avec `main`.

Dans sa version actuelle, le paquet conditext ne permet pas de créer de nouvelles propriétés de condition à sémantique linguistique.

2.1.5 Domaines de condition et propriétés de condition prédéfinis

Outres le domaine de condition `language` et les propriétés de condition à sémantique linguistique, le paquet conditext fournit plusieurs domaines de condition et propriétés de condition prédéfinis, à appellation générique, avant tout destinés à des gestions de contenu conditionnel peu exigeantes ou à des tests.

Le principal (unique ?) avantage de ces domaines de condition et propriétés de condition prédéfinis est d'éviter à l'utilisateur des créations qui pourraient s'avérer fastidieuses, dans le cas de tests notamment.

Les domaines de condition génériques prédéfinis sont au nombre de 3, disposant chacun d'un jeu de 9 propriétés de condition, soit :

— le domaine de condition `fieldi`, avec les propriétés de condition

`propa`, `propb`, `propc`, `propd`, `prope`, `propf`, `propg`, `proph`, `propi` ;

— le domaine de condition `fieldii`, avec les propriétés de condition

`propaa`, `propbb`, `propcc`, `propdd`, `propee`, `propff`, `propgg`, `prophh`, `propii` ;

— le domaine de condition `fieldiii`, avec les propriétés de condition

`propaaa`, `propbbb`, `propccc`, `propddd`, `propeee`, `propfff`, `propggg`, `prophhh`, `propiii`.

2.2 Définition d'un espace de condition

`\setcondispace`

L'on définit un espace de condition au moyen de la commande `\setcondispace`, sur la base conjointe d'un domaine de condition et d'une propriété de condition, sous la forme suivante :

`\setcondispace[*][`⟨*domaine de condition*⟩`]{`⟨*propriété de condition*⟩`}`

2.2.1 Arguments et syntaxe

Les arguments ⟨*domaine de condition*⟩ et ⟨*propriété de condition*⟩ sont à renseigner l'un et l'autre avec une valeur de type alphabétique, sans caractères accentués ou à diacritiques, et sans espaces.

Concernant l'argument `*`, cf. page 10.

Il est indispensable que ⟨*domaine de condition*⟩ et ⟨*propriété de condition*⟩ aient été créés au préalable (commandes `\newcondifield` et `\newcondiprop`, cf. respectivement page 11 et page 13) ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation.

En outre, la propriété de condition spécifiée doit avoir été impérativement créée avec un domaine de rattachement (cf. page 13) identique à

celui précisé en argument ⟨*domaine de condition*⟩ ; si tel n'est pas le cas, un message d'erreur sera sera déclenché lors de la compilation.

Si les arguments ⟨*domaine de condition*⟩ ou ⟨*propriété de condition*⟩ sont conjointement laissés vides (`[]` et `{}`), un message d'erreur sera déclenché lors de la compilation.

Dans un contexte de gestion de contenu conditionnel *multilingue*, l'argument ⟨*domaine de condition*⟩ n'a pas besoin d'être renseigné (cf. page suivante).

Pour un usage spécifique de l'argument ⟨*propriété de condition*⟩ vide, cf. page 11).

2.2.2 Utilisation

Un espace de condition permet de désigner quelles formes substituables doivent figurer dans le document généré, en fonction d'un domaine de condition et d'une propriété de condition propres à correspondre aux formes substituables concernées.

Plus précisément, un espace de condition repose sur un test de correspondance de type « égal à » strict, entre : d'une part, un domaine de condition *et* une propriété de condition spécifiés en arguments d'une instruction `setcondispace`, et d'autre part, un domaine de condition *et* une propriété de condition spécifiés en arguments d'une instruction `conditext`.

Si aucune correspondance n'existe, l'instruction `\setcondispace` est sans effet (tout se passe comme s'il n'y avait pas d'instruction formulée) ; sinon, toute forme substituable satisfaisant au test de correspondance figure dans le document généré.

L'actuelle version du paquet `conditext` supporte uniquement les tests de correspondance de type « égal à » strict.

Instructions multiples. Il y aura autant d'instructions `\setcondispace` que souhaité, avec domaines de condition identiques ou non, en fonction des besoins (notamment dès lors la « profondeur » de la gestion conditionnelle mise en oeuvre sera concernée). Si un espace de condition est défini deux fois, un message d'erreur sera déclenché lors de la compilation.

Lorsque plusieurs instructions `\setcondispace` sont formulées, l'ordre et la disposition dans lequel elles figurent, sont en principe laissés à l'appréciation de l'utilisateur ; quel que soit l'ordre et la disposition adoptés pour les instructions `\setcondispace`, les formes substituables figureront toujours dans le document généré en fonction de l'ordre dans lequel elles figurent dans le document source.

Cas des contenus conditionnels imbriqués. Lorsque l'on gère des contenus conditionnels imbriqués (cf. page 5), une instruction `\setcondispace`

présente la particularité de comportement suivante : elle est sans effet si elle désigne un contenu conditionnel de niveau inférieur à un contenu conditionnel pour lequel un espace de condition n'est pas défini.

Autrement dit, tout niveau de forme substituable supérieur à un niveau de forme substituable désigné par un espace de condition, doit être lui-même désigné par un espace de condition pour que l'espace de condition relatif au niveau inférieur soit effectif.

2.2.3 Espace de condition à sémantique linguistique

Par défaut, lorsque l'on définit un espace de condition à sémantique linguistique, l'argument ⟨*domaine de condition*⟩, de valeur `language`, est pris en compte implicitement et n'a pas à être spécifié. L'on peut ainsi formuler une instruction `setcondispace` à sémantique linguistique sous la forme :

`\setcondispace{`⟨*propriété de condition*⟩`}`

Ce comportement par défaut est cependant modifié dès lors qu'un domaine de condition implicite est défini par l'utilisateur (commande `\setimplicitcondifield`, cf. page 15). Il est alors nécessaire de spécifier *explicitement* `language` comme domaine de condition pour tout espace de condition à sémantique linguistique que l'on est amené à définir.

L'argument ⟨*propriété de condition*⟩ quant à lui, doit être impérativement renseigné avec l'une des appellations linguistiques en vigueur dans le paquet babel (cf. page 7 pour la liste des propriétés de condition à sémantique linguistique disponibles).

2.2.4 Version étoilée

La commande `\setcondispace` accepte l'argument `*`, mais uniquement dans le cadre d'une gestion de contenu conditionnel *multilingue* ; l'argument `*` est sans effet sinon.

L'étoile permet de définir, relativement à une *même langue* spécifiée en argument ⟨*propriété de condition*⟩, conjointement un espace de condition pour les formes substituables gérées par l'environnement `conditext` *et* un espace de condition pour les formes substituables gérées par la `miniconditext` (cf. page 22).

L'argument `*` peut s'avérer efficace dans les cas où les deux types de contenus substituables coexistent de manière interdépendante dans un même document source. L'usage de l'étoile a toutefois son revers : l'instruction `\setminicondispace` (cf. page 21) n'apparaissant pas explicitement, cela peut être source de confusion.

De plus, dans la mesure où il ne peut exister qu'une seule instruction `\setminicondispace` dans le préambule (cf. page 21), il ne peut exister qu'une seule instruction `\setcondispace` avec étoile. Si plusieurs instructions de ce type sont spécifiées, le compilateur ne prendra en compte

que la dernière de ces instructions, avec le risque d'effets indésirables ou inattendus.

En toute logique, l'étoile est sans effet si l'instruction `\setcondispace` est utilisée pour définir un espace de condition global (cf. de la présente page).

2.2.5 Espace de condition global

En plus de l'usage préalablement décrit, la commande `\setcondispace` peut être utilisée sous la forme spécifique suivante :

`\setcondispace[`⟨*domaine de condition*⟩`]{}`

L'espace de condition alors défini, est un *espace de condition global* : toutes les formes substituables rattachées à un même domaine de condition s'affichent sans exception dans le document généré.

À noter que l'utilisation de la commande `\setcondispace` sous la forme `\setcondispace[]{}` déclenche un message d'erreur au moment de la compilation.

Si l'on souhaite définir un espace de condition global dans un contexte de contenu conditionnel multilingue, l'argument ⟨*domaine de condition*⟩ doit être *explicitement* renseigné avec la valeur `language`.

2.3 Création d'un domaine de condition

L'on crée un domaine de condition au moyen de la commande `\newcondifield`, que l'on utilise sous la forme suivante : `\newcondifield`

`\newcondifield[*]{`⟨*nouveau domaine de condition*⟩`}`

2.3.1 argument et syntaxe

L'argument ⟨*nouveau domaine de condition*⟩ est à renseigner sur la base d'une valeur de type alphabétique, sans caractères accentués ou à diacritiques, et sans espaces.

Si l'argument ⟨*nouveau domaine de condition*⟩ est laissé vide, un message d'erreur sera déclenché lors de la compilation.

Concernant l'argument `*`, cf. page suivante.

2.3.2 Utilisation

Un domaine de condition est une thématique fonctionnelle permettant de rattacher différents contenus substituables à une même sémantique de gestion conditionnelle.

Ainsi, dans le cas de contenus substituables imbriqués, c'est le domaine de condition qui permet d'en gérer la profondeur avec ses différents niveaux.

La création d'un domaine de condition se fait toujours sur le nom ⟨*nouveau domaine de condition*⟩, à l'exclusion de tout autre identifiant. Une fois créé, ce nom peut être directement utilisé en premier argument de l'environnement `conditext` et des commandes `\setcondispace` et `\newcondiprop`, ainsi qu'en argument unique de la commande `\setimplicitcondifield`.

Il y aura autant d'instructions `\newcondifield` que de domaines de condition à créer. Si un domaine de condition est créé deux fois, un message d'erreur sera déclenché lors de la compilation.

Il est possible de donner un nom identique à un domaine de condition et à une propriété de condition, mais, de par les risques de confusion que cette pratique est susceptible de produire, il convient de la réserver à des besoins spécifiques (cf. de la présente page).

En toute logique, toute instruction `\newcondifield` doit impérativement précéder l'utilisation de l'environnement `conditext` et des commandes `\setcondispace`, `\newcondiprop` et `\setimplicitcondifield` ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation. Plus généralement, un message d'erreur sera déclenché lors de la compilation si un domaine de condition est spécifié dans une instruction `conditext`, `\setcondispace` ou `\setimplicitcondifield` alors qu'il n'a pas été créé.

2.3.3 Version étoilée

La commande `\newcondifield` accepte l'argument `*`, qui permet de créer tout à la fois un domaine de condition *et* une propriété de condition portant le même nom.

Cet usage particulier de `\setcondispace` est essentiellement destiné à une gestion conditionnelle binaire, où une seule propriété de condition suffit par domaine de condition (l'instruction `\setcondispace` faisant office de « commande interrupteur »).

Il doit en principe rester modéré : aucune instruction explicite relative à la propriété de condition ainsi créée ne figurant dans le code du document, cela peut induire une certaine confusion lorsque plusieurs instructions `\newcondifield` avec étoile sont formulées, et favoriser le risque de créations de propriétés de condition en doublon si l'on n'est pas suffisamment vigilant.

2.3.4 Modification d'un domaine de condition

Pour modifier un domaine de condition, il conviendra d'utiliser les fonctionnalités de recherche/remplacement propres à l'éditeur LaTeX utilisé, en veillant à ce que la valeur ⟨*domaine de condition modifié*⟩ en argument de `\newcondifield` soit répercutée dans *toutes* les instruc-

tions concernées, `conditext`, `\setcondispace`, `\newcondiprop` ou encore `\setimplicitcondispace`.

2.3.5 Suppression d'un domaine de condition

La suppression d'un domaine de condition se pratique en retirant simplement l'instruction `\newcondifield` correspondant à ce domaine de condition. L'utilisateur veillera toutefois à ce qu'une fois cette instruction retirée, le domaine de condition concerné ne demeure spécifié dans aucune instruction `\conditext`, `\setcondispace`, `\newcondiprop` ou `\setimplicitcondispace` ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation.

2.4 Création d'une propriété de condition

L'on crée une propriété de condition au moyen de la commande `\newcondiprop`, que l'on utilise sous la forme suivante : `\newcondiprop`

`\newcondiprop{`⟨*domaine de condition de rattachement*⟩`}{`⟨*nouvelle propriété de condition*⟩`}`

2.4.1 Argument et syntaxe

Les arguments ⟨*domaine de condition de rattachement*⟩ et ⟨*propriété de condition*⟩ sont à renseigner l'un et l'autre sur la base d'une valeur de type alphabétique, sans caractères accentués ou à diacritiques, et sans espaces.

Il est indispensable que ⟨*domaine de condition de rattachement*⟩ ait été au préalable créé (commande `\newcondifield`, cf. page 11) ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation.

Même si le domaine de condition de rattachement est par ailleurs défini comme domaine de condition implicite (commande `\setimplicitcondifield`, cf. page 15), il doit être cependant spécifié ici *explicitement*. De manière générale, l'argument ⟨*domaine de condition de rattachement*⟩ ne peut être laissé vide ; sinon, un message d'erreur sera déclenché lors de la compilation.

De même, si l'argument ⟨*nouvelle propriété de condition*⟩ est laissé vide, un message d'erreur sera déclenché lors de la compilation.

2.4.2 Utilisation

Une propriété de condition est une caractérisation fonctionnelle propre à une forme substituable.

En tant que telle, une propriété de condition est toujours rattachée à un domaine de condition ; plusieurs propriétés de condition rattachées à un même domaine de condition constitue ainsi un « jeu » pouvant être

utilisé à chaque fois que le domaine de condition concerné est spécifié dans l'environnement `conditext` et la commande `\setcondispace`.

La création d'une propriété de condition se fait toujours sur le nom ⟨*nouvelle propriété de condition*⟩, à l'exclusion de tout autre identifiant. Une fois créé, ce nom peut être directement utilisé en deuxième argument de l'environnement `conditext` et de la commande `\setcondispace`.

Il y aura autant d'instructions `\newcondiprop` que de propriétés de condition à créer. Si une propriété de condition est créée deux fois, un message d'erreur sera déclenché lors de la compilation. Il convient d'être particulièrement vigilant sur ce genre d'erreurs lorsque l'on utilise la commande `\newcondifield` avec `*` (cf. page 12), aucune instruction `\newcondiprop` n'apparaissant alors pour la propriété créée.

Il est possible de donner à une propriété de condition le nom d'un domaine de condition, mais, de par les risques de confusion que cette pratique est susceptible de produire, il convient de la réserver à des besoins spécifiques (cf. page 12).

Similairement, dans l'absolu, une propriété de condition rattachée à un domaine de condition donné peut porter le même nom qu'une propriété de condition rattachée à un autre domaine de condition, mais cette pratique ne devrait être utilisée que pour répondre à des besoins très particuliers, afin d'éviter tout risque inutile de confusion.

En toute logique, toute instruction `\newcondiprop` doit impérativement précéder l'utilisation de l'`conditext` et de la commande `\setcondispace` ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation. Plus généralement, un message d'erreur est déclenché lors de la compilation si une propriété de condition est spécifiée dans une instruction environnement `conditext` ou `\setcondispace` alors qu'elle n'a pas été créée.

2.4.3 Modification d'une propriété de condition

Pour modifier une propriété de condition, il conviendra d'utiliser les fonctionnalités de recherche/remplacement propres à l'éditeur LaTeX utilisé, en veillant à ce que la valeur ⟨*propriété de condition modifiée*⟩ en argument de `\newcondiprop` soit répercutée dans *toutes* les instructions concernées, `conditext` ou `\setcondispace`.

2.4.4 Suppression d'une propriété de condition

La suppression d'une propriété de condition se pratique en retirant simplement l'instruction `\newcondiprop` correspondant à cette propriété de condition (ou l'instruction `\newcondifield*`, cf. page 12) . L'utilisateur veillera toutefois à ce qu'une fois cette instruction retirée, la propriété de condition concernée ne demeure spécifiée dans aucune instruction `\conditext` ou `\setcondispace` ; dans le cas contraire, un message d'erreur sera déclenché lors de la compilation.

2.5 Définition d'un domaine de condition implicite

Dans certains cas, notamment lorsque de nombreux environnements `conditext` doivent être utilisés avec un domaine de condition identique, il peut s'avérer pratique – à la fois pour faciliter la saisie et rendre le code plus lisible – que ce domaine de condition n'ait pas à être spécifié à chaque fois.

L'on peut alors définir ce domaine de condition comme *domaine de condition implicite* (en anglais, « implicit condition field »). Par défaut, c'est le domaine de condition `language` (cf. page 7) qui est défini comme domaine de condition implicite.

L'on définit un domaine de condition implicite au moyen de la commande `\implicitcondifield`, sous la forme :

`\implicitcondifield`

`\setimplicitcondifield{`⟨*domaine de condition*⟩`}`

2.5.1 Argument

L'argument ⟨*domaine de condition*⟩ est à renseigner avec une valeur de type alphabétique, sans caractères accentués ou à diacritiques, et sans espaces.

Il est indispensable que ⟨*domaine de condition*⟩ ait été au préalable créé (commande `\newcondifield` (cf. page 11) ; dans le cas contraire, un message d'erreur est déclenché au moment de la compilation.

De même, si l'argument ⟨*domaine de condition*⟩ est laissé vide, un message d'erreur sera déclenché lors de la compilation.

2.5.2 Utilisation

Il ne peut exister qu'une seule instruction `\setimplicitcondifield` – donc, un seul domaine de condition implicite – par document source. Si plusieurs instructions de ce type sont spécifiées, le compilateur ne prendra en compte que la dernière de ces instructions, avec le risque d'effets indésirables ou inattendus.

L'instruction `\setimplicitcondifield` devra être précisée avant toute instruction `\setcondispace`.

Il est possible, bien qu'inutile, de spécifier explicitement un domaine de condition défini comme domaine de condition implicite.

Si l'utilisateur définit un domaine de condition implicite tout en ayant à gérer par ailleurs certains contenus conditionnels multilingues dans le même document source, il est alors nécessaire de spécifier *explicitement* `language` dans les instructions `conditext` et `\setcondispace` liées à cette gestion conditionnelle multilingue – là où, par défaut, cela n'avait pas lieu d'être (cf. page 7).

L'on peut rétablir le comportement par défaut (`language` comme domaine de condition implicite) : soit en retirant simplement l'instruction

`\setimplicitcondispace`, soit en formulant l'instruction `\setimplicitcondifield{language}`.

3 Commandes relatives à la mini-gestion conditionnelle

3.1 Saisie des formes substituables pour une mini-gestion conditionnelle

Toute forme substituable destinée à une mini-gestion conditionnelle doit être saisie au moyen de la `\miniconditext`.

À la différence de l'environnement `conditext` qui permet de gérer des formes substituables multi-paragraphiques, la commande `\miniconditext` est destinée à gérer des contenus conditionnels à l'échelle paragraphique ou intra-paragraphique (mots, groupes de mots).

L'un des usages auxquels elle est destinée, est la gestion conditionnelle multilingue de termes ou d'intitulés, mais il peut aussi s'agir d'expressions à formatage conditionnel en fonction des destinataires d'un document donné etc.

Autre différence fondamentale avec l'environnement `conditext` : alors que, dans le cadre d'une gestion de contenu conditionnel à l'échelle multi-paragraphique, l'on peut créer et gérer autant de domaines de condition et de propriétés de condition, et utiliser autant d'environnement `conditext` et d'espaces de condition que souhaité, la commande `\miniconditext` est limitée à 9 propriétés de condition et 9 possibilités de formes substituables et il ne peut y avoir qu'un seul espace de condition pour mini-gestion conditionnelle dans le préambule.

La commande `\miniconditext` s'utilise sous la forme suivante :

`\miniconditext`

`\miniconditext{`⟨*forme substituable n° 1*⟩`}[`⟨*forme substituable n° 2*⟩`][`⟨*forme substituable n° 3*⟩`][`⟨*forme substituable n° 4*⟩`][`⟨*forme substituable n° 5*⟩`][`⟨*forme substituable n° 6*⟩`][`⟨*forme substituable n° 7*⟩`][`⟨*forme substituable n° 8*⟩`][`⟨*forme substituable n° 9*⟩`]`

3.1.1 Arguments et syntaxe

Les arguments ⟨*forme substituable n° 1*⟩, ⟨*forme substituable n° 2*⟩ etc., sont à renseigner sur la base d'une valeur de type alphanumérique représentant un ou plusieurs mots, éventuellement un paragraphe.

Les deuxième, troisième, ...neuvième arguments peuvent ne pas être renseignés, sur la base de la règle suivante : tout argument non renseigné entre deux arguments renseignés doit figurer sous la forme `[]` (tout argument non renseigné positionné après le dernier argument renseigné pouvant ne pas être mentionné).

Le fait de renseigner un argument auquel est associée une propriété de condition « neutre » (cf. page 19) est sans effet lors de la compilation.

3.1.2 Utilisation

L'on place une instruction `\miniconditext` à chaque endroit du document source où l'on souhaite que l'une des formes substituables spécifiées dans les arguments ⟨*forme substituable n° 1*⟩, ⟨*forme substituable n° 2*⟩ etc., figure dans le document généré en fonction de l'espace de condition qui sera défini.

Dans un document source, il y aura autant d'instructions `\miniconditext` que de formes substituables devant être gérées conditionnellement. Les formes substituables spécifiées dans les arguments ⟨*forme substituable n° 1*⟩, ⟨*forme substituable n° 2*⟩ etc. pourront être, selon les cas et les besoins, soit identiques, soit différentes d'une instruction `\miniconditext` à une autre.

Forme minimale d'utilisation. La forme minimale d'utilisation de la commande `\miniconditext` est :

`\miniconditext{`⟨*forme substituable*⟩`}`

Cela revient, dans ce cas, à pouvoir gérer l'affichage ou l'absence d'affichage d'une forme substituable donnée.

Formes substituables et propriétés de condition. La commande `\miniconditext` est, de toutes les commandes fournies par le paquet conditext, celle qui est la moins intuitive.

D'une part, en effet, dans un contexte de mini-gestion conditionnelle, la notion de domaine de condition n'existe pas à proprement parler ; plus exactement, `miniconditext` est un domaine de condition en soi.

D'autre part, à chacun des 9 arguments de la commande `\miniconditext` est associée une propriété de condition, dont la particularité essentielle réside dans le fait d'être *globale* à la commande elle-même (et donc *à l'ensemble* des instructions relatives à cette commande).

Ces 9 propriétés au total n'apparaissent pas dans la signature de la commande, mais il convient de toujours les garder à l'esprit puisque ce sont elles qui caractérisent chaque forme substituable.

C'est également sur la base de ces propriétés de condition que l'on définit l'espace de condition permettant de désigner laquelle des 9 formes substituables doit figurer dans le document généré, pour chaque instruction `\miniconditext` placée dans le document source. Dans la version actuelle du paquet conditext, il ne peut exister *qu'un seul espace de condition par préambule* (cf. page 21).

Par défaut, la sémantique des 9 propriétés de condition est linguistique (cf. page suivante) mais l'on peut également redéfinir ces propriétés

de condition pour les adapter à un usage spécifique (cf. page 20).

Mini-gestion conditionnelle multilingue. Par défaut, la commande `\miniconditext` est prévue pour gérer un contenu conditionnel multilingue.

En d'autres termes, les propriétés de condition associées aux arguments de la commande ont par défaut une sémantique linguistique, et chaque argument de la commande correspond ainsi à une langue donnée : le français pour ⟨*forme substituable n° 1*⟩, l'anglais pour ⟨*forme substituable n° 2*⟩, l'allemand pour ⟨*forme substituable n° 3*⟩, l'espagnol pour ⟨*forme substituable n° 4*⟩, l'italien pour ⟨*forme substituable n° 5*⟩, le portuguais pour ⟨*forme substituable n° 6*⟩, le néerlandais pour ⟨*forme substituable n° 7*⟩, le danois pour ⟨*forme substituable n° 8*⟩ et le norvégien pour ⟨*forme substituable n° 9*⟩.

Il est possible de modifier l'ordre des langues ou les langues elles-mêmes en redéfinissant les propriétés de condition (cf. page 20).

3.2 Redéfinition des propriétés de condition pour une mini-gestion conditionnelle

À chacun des 9 arguments de la `miniconditext` est associée une propriété de condition, qui sert à caractériser la forme substituable correspondant à cet argument. Chacune de ces 9 propriétés de condition est, dans son association avec l'un des arguments, *globale* à la commande elle-même.

Par défaut, chacun des arguments de la commande `\miniconditext` est associé à une propriété de condition à sémantique linguistique. Ces propriétés de condition ont les noms respectifs suivants, dans l'ordre des arguments de la commande `\miniconditext` : `french`, `english`, `german`, `spanish`, `italian`, `portuguese`, `dutch`,`danish`, `norsk`.

Dans un contexte de mini-gestion conditionnelle multilingue, ce sont ces noms que l'on utilise pour définir l'espace de condition concerné (cf. page 21).

Il est possible de modifier l'ordre des langues et/ou les langues elles-mêmes, cf. page 20.

Dès lors que l'on souhaite mettre en oeuvre une mini-gestion conditionnelle dans un contexte autre que multilingue, il est nécessaire de modifier les propriétés de condition, au moyen des commandes suivantes :
`\setminicondipropi`, `\setminicondipropii`, `\setminicondipropiii`, `\setminicondipropiv`, `\setminicondipropv`, `\setminicondipropvi`, `\setminicondipropvii`, `\setminicondipropviii` et `\setminicondipropix`.

Chacune de ces commandes sert à modifier – l'on dira désormais *redéfinir* – une propriété de condition sur la base d'un indice de position en chiffres romains, correspondant à celui de l'argument auquel est associée la propriété de condition dans la commande `\miniconditext`. Ainsi,

la commande `\setminicondipropi` redéfinit la propriété de condition associée au premier argument, la commande `\setminicondipropii` redéfinit la propriété de condition associée au deuxième argument et ainsi de suite.

Une commande `\setminicondiprop-` (où `-` représente `i`, `ii`, `iii`, `iv`, `v`, `vi`, `vii`, `viii` ou `ix`) s'utilise sous la forme :

`\setminicondiprop-{`⟨*propriété de condition redéfinie*⟩`}`

3.2.1 Argument et syntaxe

L'argument ⟨*propriété de condition redéfinie*⟩ est à renseigner sur la base d'une valeur de type alphabétique, sans espace ni caractères accentués ou à diacritiques.

Si l'argument ⟨*propriété de condition redéfinie*⟩ est laissé vide, un message d'erreur sera déclenché lors de la compilation.

3.2.2 Utilisation

Lorsque l'on utilise une commande `\setminicondiprop-`, il n'y a pas création d'une nouvelle propriété de condition, mais *redéfinition* d'une des propriétés de condition existant par défaut, par modification du nom, à l'exclusion de tout autre identifiant.

Pour redéfinir les propriétés de condition, il faut toujours respecter le schéma d'instructions suivant :

— d'abord, une instruction `\resettingminicondiprops` ;
— ensuite, les instructions `\setminicondiprop-`.

La commande `\resettingminicondiprops` est une commande particulière qui effectue certaines initialisations indispensables avant que les propriétés de condition puissent être redéfinies. Elle accepte un argument `*`, réservé à la redéfinition des propriétés de condition de sémantique linguistique (cf. page suivante). `\resettingminicondiprops`

De par sa fonction particulière, l'instruction `\resettingminicondiprops` est indissociable des instructions `\setminicondiprop-`. Elle doit être obligatoirement placée *avant* les instructions `\setminicondiprop-`, `\miniconditext` et `\setminicondispace`. En cas de non formulation, un message d'erreur sera déclenché lors de la compilation.

Principes de la redéfinition/propriétés de condition neutres. En fonction des besoins, la redéfinition des propriétés de condition peut être *totale ou partielle*. L'on ne formule que les instructions `\setminicondiprop-` relatives aux propriétés de condition qui doivent être redéfinies.

Cela signifie qu'il y a aura autant d'instructions `\setminicondiprop-` que de propriétés de condition à redéfinir. Pour rappel, toute instruction

de type `\setminicondiprop-{}` déclenchera un message d'erreur lors de la compilation.

Lorsque la commande `\resettingminicondiprops` est utilisée sans étoile, toute propriété de condition non redéfinie (aucune instruction `\setminicondiprop-` correspondante) est considérée comme « neutre ». Si une forme substituable est renseignée dans un argument de `\miniconditext` auquel est associée une propriété de condition neutre, cette forme substituable ne sera pas prise en compte lors de la génération du document.

Si une propriété de condition est redéfinie avec un nom déjà utilisé pour une autre propriété de condition, un message d'erreur sera déclenché lors de la compilation.

Si ⟨*propriété de condition redéfinie*⟩ fait elle-même l'objet d'une modification dans une ou plusieurs instructions, l'utilisateur veillera à ce que cette modification soit répercutée dans `\setminicondispace` si besoin est.

Similairement, le fait de retirer une instruction `\setminicondiprop-` nécessite de retirer également toute forme substituable préalablement saisie dans l'argument correspondant de `\miniconditext`.

Il convient d'être particulièrement attentif au fait que la redéfinition des propriétés de condition a des implications sémantiques (les formes substituables spécifiées dans les arguments ⟨*forme substituable n° 1*⟩, ⟨*forme substituable n° 2*⟩ etc. de la commande `\miniconditext`) et fonctionnelles (la propriété de condition spécifiée avec la commande `\setminicondispace`).

La modification des propriétés de condition doit se faire de manière raisonnée, selon une sémantique d'ensemble. Ainsi, même si les commandes `\setminicondiprop-` l'autorisent, a priori, faire coexister des propriétés de condition d'une certaine sémantique avec d'autres de sémantique différente est source potentielle de confusion, avec les éventuels effets indésirables ou inattendus que cela peut entraîner. De même – sauf pour un usage exotique – il conviendra d'éviter les redéfinitions discontinues et de privilégier à tout prix de redéfinir les propriétés de condition dans l'ordre des indices.

3.2.3 Redéfinition de propriétés de condition de sémantique linguistique

Il est possible d'adapter la mini-gestion conditionnelle multilingue à des besoins précis, en redéfinissant tout ou partie des langues par défaut. Dans l'actuelle version du paquet conditext, modifier l'ordre des langues revient à redéfinir les langues dans le nouvel ordre souhaité.

De manière générale, toute langue redéfinie doit l'être impérativement sous la forme d'une des appellations linguistiques en vigueur dans le paquet babel ; autrement, un message d'erreur sera déclenché lors de la compilation.

Deux manières de faire existent, selon que l'on fait précéder les ins-

tructions `\setminicondiprop-` de l'instruction `\resettingminicondiprops` avec ou sans étoile :

Lorsque les modifications à apporter s'avèrent importantes (toutes ou une grande partie des langues à redéfinir), l'usage de la commande `\resettingminicondiprops` *sans étoile* est recommandée, dans la mesure où le type de réinitialisation effectuée permet une lisibilité optimale : à une instruction `\setminicondiprop-` formulée correspond une langue effective et utilisable, et toute propriété de condition non redéfinie est considérée comme une absence de langue à la compilation. Si une forme substituable est renseignée dans un argument de `\miniconditext` auquel n'est associée aucune langue, cette forme substituable n'est pas prise en compte lors de la génération du document.

Lorsque les modifications à apporter s'avèrent peu importantes (quelques langues à redéfinir, ou la permutation d'une langue avec une autre), l'on peut utiliser la commande `\resettingminicondiprops` *avec étoile* : dans ce cas, toute propriété de condition non redéfinie *conserve l'intitulé de langue par défaut*. Cela étant, si elle permet une certaine efficacité, l'instruction `\resettingminicondiprops*` n'est cependant pas exempte de défauts, notamment une lisibilité moindre pouvant favoriser certaines confusions lors des redéfinitions.

L'usage de la commande `\resettingminicondiprops` avec l'argument `*` doit être réservé aux propriétés de condition de sémantique linguistique ; autrement, cela ne peut être que source d'effets indésirables ou inattendus.

3.3 Définition d'une espace de condition pour une mini-gestion conditionnelle

L'on définit un espace de condition pour mini-gestion conditionnelle au moyen de la commande `\setminicondispace`, sous la forme suivante :

`\setminicondispace`

`\setminicondispace{`⟨*propriété de condition*⟩`}`

3.3.1 Argument et syntaxe

L'argument ⟨*propriété de condition*⟩ est à renseigner sur la base d'une valeur de type alphabétique, sans espace ni caractères accentués ou à diacritiques.

Si l'argument ⟨*propriété de condition redéfinie*⟩ est laissé vide, un message d'erreur sera déclenché lors de la compilation.

Toute propriété de condition spécifiée doit correspondre à l'une des 9 propriétés de condition associées aux arguments de la commande `\conditext` (soit définie par défaut, soit redéfinie avec une commande `\setminicondiprop-`, cf. page 18) ; dans le cas contraire, un message d'erreur sera systématiquement déclenché lors de la compilation.

3.3.2 Utilisation

Un espace de condition pour mini-gestion conditionnelle se définit sur la base d'une des 9 propriétés de condition associées aux arguments de la `\miniconditext`. C'est cette propriété de condition qui, rapportée à un test de correspondance de type « égal à » strict, permet de désigner, pour chaque instruction `\miniconditext` formulée dans le document source, la forme substituable devant figurer dans le document généré.

Les 9 propriétés de condition spécifiques à la mini-gestion conditionnelles sont *globales* à la commande `\miniconditext` et, par conséquent, *à toutes* les instructions `\miniconditext` formulées dans le document source. Il ne peut donc exister qu'une seule instruction `\setminicondispace` effective dans le préambule. Si plusieurs instructions sont spécifiées, le compilateur ne prendra en compte que la dernière de ces instructions, avec le risque d'effets indésirables ou inattendus.

Il est toutefois possible – bien qu'il ne s'agisse pas d'une pratique régulière – de passer outre cette limitation de principe, en plaçant des instructions `\setminicondispace` à même le document source : chacune de ces instructions devient alors effective tant qu'une autre n'est pas rencontrée. Ainsi, d'une certaine façon, plusieurs espaces de condition pour mini-gestion conditionnelle peuvent coexister linéairement parlant (au risque d'effets indésirables ou inattendus).

3.3.3 Espaces de condition couplés

Dans un contexte de gestion de contenu conditionnel *multilingue*, l'usage de la commande `\setcondispace` avec l'argument `*` (cf. page 10) permet de définir, relativement à une *même langue* spécifiée en argument ⟨*propriété de condition*⟩ de `\setcondispace`, à la fois un espace de condition pour les formes substituables gérées par l'environnement `conditext` *et* un espace de condition pour les formes substituables gérées par la `miniconditext`.

En d'autres termes, une instruction `\setminicondispace` est formulée implicitement en même temps que l'instruction `\setcondispace` elle-même.

L'argument `*` peut s'avérer efficace dans les cas où les deux types de contenus substituables coexistent de manière interdépendante dans un même document source. L'usage de l'étoile a toutefois son revers : l'instruction `\setminicondispace` n'apparaissant pas explicitement, cela peut être source de confusion.

De plus, dans la mesure où il ne peut exister qu'une seule instruction `\setminicondispace` dans le préambule, il ne peut exister qu'une seule instruction `\setcondispace*`. Si plusieurs instructions de ce type sont spécifiées, le compilateur ne prendra en compte que la dernière de ces instructions, avec le risque d'effets indésirables ou inattendus.

Deuxième partie

Tutoriel : exemples commentés

L'une des principales applications du texte conditionnel résidant dans la production de documents multilingues, c'est ce genre de document que le paquet conditext permet de gérer par défaut, sans nécessiter de préalable d'aucune sorte.

Dans ce tutoriel, nous commencerons précisément avec un exemple de document multilingue, qui nous familiarisera avec les notions de formes substituables, de domaines de condition, de propriétés de condition et d'espaces de condition. Ce premier exemple sera l'occasion de présenter quelques bonnes pratiques d'utilisation quant aux principales commandes du paquet.

Nous passerons ensuite à un exemple où nous créerons cette fois des domaines de condition et des propriétés de condition. Nous verrons également avec ce deuxième exemple comment l'on peut imbriquer des formes substituables dans d'autres formes substituables et gérer ainsi une certaine « profondeur » de contenu conditionnel.

Enfin, dans un troisième exemple, en deux parties, nous apprendrons à utiliser un ensemble de commandes permettant une mini-gestion conditionnelle, portant sur des formes substituables minimales.

Remarque. Pour les besoins de certains de ces exemples, nous serons amenés à créer quelques commandes personnalisées. À cette occasion, nous adopterons la syntaxe propre au paquet xparse, assez intuitive et dont la lisibilité est un atout dans le cadre d'un tutoriel. Dans nos exemples, l'instruction pour créer les commandes concernées aura toujours la forme suivante :

`\NewDocumentCommand\`⟨*nom de la commande*⟩`{`⟨*arguments*⟩`}{`⟨*code*⟩`}`

où ⟨*arguments*⟩ représente les arguments qui doivent figurer dans le code : `m` (pour « mandatory ») et `o` (pour « optional »). Un argument optionnel peut avoir une valeur par défaut ; dans ce cas, il est représenté par `O{`⟨*valeur par défaut*⟩`}`.

Tous les exemples de document source présentés dans le tutoriel sont directement utilisables et compilables ; la manière dont ils sont présentés n'a, en tant que telle, rien d'impératif.

1 Exemple commenté n° 1 : un document en version multilingue

Imaginons – de manière quelque peu artificielle il est vrai – que nous ayons à rédiger dans un seul et même document source, un texte en

version française, anglaise, allemande, espagnole et italienne, avec pour objectif de générer ensuite un document par version linguistique de ce texte.

Pour des raisons de commodité, nous prendrons ici pour texte l'article premier de la *Déclaration des droits de l'homme*, décliné respectivement en français, anglais, allemand, espagnol et italien. À noter toutefois que le paquet conditext permet de gérer des textes conditionnels de plusieurs dizaines à plusieurs milliers de signes.

Voici le document source correspondant à notre exemple :

```
\documentclass{article}
\usepackage[english,german,spanish,italian,french]{babel}
\usepackage{conditext}

\begin{document}

\begin{conditext}{french}
 Tous les êtres humains naissent libres et égaux en dignité et en
 droits. Ils sont doués de raison et de conscience et doivent agir
 les uns envers les autres dans un esprit de fraternité.
\end{conditext}

\begin{conditext}{english}
 All human beings are born free and equal in dignity and rights. They
 are endowed with reason and conscience and should act towards one
 another in a spirit of brotherhood.
\end{conditext}

\begin{conditext}{german}
 Alle Menschen sind frei und gleich an Würde und Rechten geboren.
 Sie sind mit Vernunft und Gewissen begabt und sollen einander im
 Geist der Brüderlichkeit begegnen.
\end{conditext}

\begin{conditext}{spanish}
 Todos los seres humanos nacen libres e iguales en dignidad y derechos
 y, dotados como están de razón y conciencia, deben comportarse fraternalmente
 los unos con los otros.
\end{conditext}

\begin{conditext}{italian}
 Tutti gli esseri umani nascono liberi ed eguali in dignità e diritti.
 Essi sono dotati di ragione e di coscienza e devono agire gli uni
 verso gli altri in spirito di fratellanza.
\end{conditext}

\end{document}
```

L'on remarquera tout d'abord que chaque version linguistique du texte est placée dans un environnement nommé `conditext`, et se voit spécifier le nom de la langue dans laquelle elle est rédigée, sous la forme d'un argument de type obligatoire.

Cet argument correspond à une *propriété de condition*, c'est-à-dire une caractérisation fonctionnelle sur la base de laquelle l'on pourra ensuite désigner quelle version de texte doit être générée. Dans le cas de notre exemple, il s'agit de propriétés à sémantique linguistique, prédéfinies et fournies par le paquet conditext. Il est possible de créer et utiliser des propriétés de condition de toute nature, comme nous le verrons cf. page 31.

Dans le cas d'un contenu conditionnel multilingue, l'environnement `conditext` n'utilise qu'un seul argument. En principe, il existe un second argument, optionnel celui-là, servant à spécifier le domaine de condition, mais pour l'instant, il suffit de retenir que c'est l'argument obligatoire qui permet de spécifier la propriété de condition.

De manière générale, tout texte que l'on souhaite gérer comme *forme substituable* doit être placé dans un environnement `conditext`. Dans un même document source, il peut être spécifié autant d'environnements `conditext` que souhaité, y compris avec des propriétés de condition (et des domaines de condition) identiques.

Ainsi, dans notre exemple, l'on pourrait faire figurer toute la *Déclaration des droits de l'homme*, avec chacun des articles présenté en version linguistique distincte (nous verrons plus bas que, s'il peut sembler a priori plus pratique de regrouper tous les articles d'une même version linguistique dans un même environnement `conditext`, le partitionnement permet dans certains cas une gestion conditionnelle plus précise).

Les propriétés de condition de notre exemple ont respectivement la valeur `french`, `english`, `german`, `spanish` et `italian`. Lorsqu'il s'agit de texte conditionnel multilingue, les propriétés de condition doivent impérativement correspondre aux appellations linguistiques utilisées par le paquet babel. Il est indispensable que ce paquet soit déclaré dans le préambule, accompagné des langues susceptibles d'être spécifiées comme propriétés de condition. Pour les besoins de notre exemple, le préambule contient donc la déclaration suivante :

```
\usepackage[english,german,spanish,italian,french]{babel}
```

(faute de quoi, le compilateur produit un message d'erreur).

Si nous compilons le document source en l'état, le document généré... est vide (ou ne s'affiche pas, selon l'éditeur LaTeX utilisé). Ceci est logique, en ce sens où il manque en effet l'instruction permettant de désigner quelle forme substituable doit figurer sur le document. En l'absence de cette instruction, aucun contenu d'environnement `conditext` ne figure dans le document généré.

Partons de l'idée de générer un document avec la version italienne de l'article premier de la *Déclaration des droits de l'homme*. Pour ce faire, dans le préambule, à la suite de la déclaration des paquets, nous allons ajouter la ligne suivante :

```
\setcondispace{italian}
```

Si nous compilons à nouveau le document source, c'est désormais le texte suivant qui figure conformément à nos attentes dans le document généré :

> Tutti gli esseri umani nascono liberi ed eguali in dignità e diritti. Essi sono dotati di ragione e di coscienza e devono agire gli uni verso gli altri in spirito di fratellanza.

Si nous souhaitons maintenant générer un document avec la version française, il suffit de modifier l'instruction initiale en :

```
\setcondispace{french}
```

et de recompiler le document source. L'on obtient alors :

> Tous les êtres humains naissent libres et égaux en dignité et en droits. Ils sont doués de raison et de conscience et doivent agir les uns envers les autres dans un esprit de fraternité.

Et ainsi de suite...

L'instruction `\setcondispace` définit un *espace de condition*. Dans les deux cas précédents, ce sont les valeurs `italian` et `french` qui sont spécifiées en argument de `\setcondispace` afin de sélectionner : dans le premier cas, la forme substituable correspondant au contenu de l'environnement `conditext` ayant `italian` spécifié en argument, et dans le deuxième cas, la forme substituable correspondant au contenu de l'environnement `conditext` ayant `french` spécifié en argument.

Plus précisément, un espace de condition repose sur un test de correspondance de type « égal à » strict, entre une propriété de condition spécifiée en argument d'une instruction `setcondispace` et une propriété de condition spécifiée en argument d'une instruction `\conditext`. Si aucune correspondance n'existe, l'instruction `\setcondispace` est sans effet (tout se passe comme s'il n'y avait pas d'instruction formulée) ; sinon, la correspondance existante permet de désigner une ou plusieurs formes substituables pour figurer dans le document généré.

Dans le cas d'une gestion conditionnelle multilingue, la commande `\setcondispace` n'utilise qu'un seul argument (la propriété de condition). En principe, un second argument, optionnel, permet de spécifier le domaine de condition – le test de correspondance portant alors à la fois sur le domaine de condition et la propriété de condition.

Les instructions `conditext` et `\setcondispace` de notre exemple répondent à notre besoin initial. Imaginons maintenant – à nouveau, de manière assez artificielle – que nous ayons à générer les versions anglaise, allemande et espagnole du texte dans un seul et même document, sans les versions française et italienne.

La solution est simple : il suffit dans ce cas de spécifier autant d'instructions `\condispace` que de langues concernées. Dans le préambule, à la place de `\setcondispace{italian}`, nous aurons donc :

```
\setcondispace{english}
\setcondispace{german}
\setcondispace{spanish}
```

Désormais, le document généré contient le texte :

> All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.
>
> Alle Menschen sind frei und gleich an Würde und Rechten geboren. Sie sind mit Vernunft und Gewissen begabt und sollen einander im Geist der Brüderlichkeit begegnen.
>
> Todos los seres humanos nacen libres e iguales en dignidad y derechos y, dotados como están de razón y conciencia, deben comportarse fraternalmente los unos con los otros.

Ainsi qu'on peut le constater, il peut être formulé plusieurs instructions `\setcondispace` en fonction des besoins. En principe, ces instructions peuvent être indifféremment dans un ordre ou dans un autre ; ainsi, `\setcondispace{german}` aurait très bien pu précéder `\setcondispace{english}` sans provoquer une présentation différente du texte dans le document généré. En fait, le texte qui figure dans le document généré respecte toujours l'ordre et la disposition adoptés dans le document source ; d'un certain point de vue, c'est l'une des limites (actuelles ?) du paquet conditext.

Nous allons maintenant étoffer notre exemple de départ, en insérant, à la suite de chaque version française, anglaise, allemande, espagnole et italienne de l'article premier de la *Déclaration des droits de l'homme*, le texte de l'article 2 décliné dans les mêmes langues. Le document source est désormais le suivant, hors préambule :

```
\begin{document}

\begin{conditext}{french}
 Tous les êtres humains naissent libres et égaux en dignité et en
 droits. Ils sont doués de raison et de conscience et doivent agir
 les uns envers les autres dans un esprit de fraternité.
\end{conditext}
```

```
\begin{conditext}{french}
 Chacun peut se prévaloir de tous les droits et de toutes les libertés
 proclamés dans la présente Déclaration, sans distinction aucune,
 notamment de race, de couleur, de sexe, de langue, de religion, d'opinion
 politique ou de toute autre opinion, d'origine nationale ou sociale,
 de fortune, de naissance ou de toute autre situation.
 De plus, il ne sera fait aucune distinction fondée sur le statut
 politique, juridique ou international du pays ou du territoire dont
 une personne est ressortissante, que ce pays ou territoire soit indépendant,
 sous tutelle, non autonome ou soumis à une limitation quelconque
 de souveraineté.
\end{conditext}

\begin{conditext}{english}
 All human beings are born free and equal in dignity and rights. They
 are endowed with reason and conscience and should act towards one
 another in a spirit of brotherhood.
\end{conditext}

\begin{conditext}{english}
 Everyone is entitled to all the rights and freedoms set forth in
 this Declaration, without distinction of any kind, such as race,
 colour, sex, language, religion, political or other opinion, national
 or social origin, property, birth or other status.
 Furthermore, no distinction shall be made on the basis of the political,
 jurisdictional or international status of the country or territory
 to which a person belongs, whether it be independent, trust, non-self-governing
 or under any other limitation of sovereignty.
\end{conditext}

\begin{conditext}{german}
 Alle Menschen sind frei und gleich an Würde und Rechten geboren.
 Sie sind mit Vernunft und Gewissen begabt und sollen einander im
 Geist der Brüderlichkeit begegnen.
\end{conditext}

\begin{conditext}{german}
 Jeder hat Anspruch auf die in dieser Erklärung verkündeten Rechte
 und Freiheiten ohne irgendeinen Unterschied, etwa nach Rasse, Hautfarbe,
 Geschlecht, Sprache, Religion, politischer oder sonstiger Überzeugung,
 nationaler oder sozialer Herkunft, Vermögen, Geburt oder sonstigem
 Stand.
 Des weiteren darf kein Unterschied gemacht werden auf Grund der politischen,
 rechtlichen oder internationalen Stellung des Landes oder Gebiets,
 dem eine Person angehört, gleichgültig ob dieses unabhängig ist,
 unter Treuhandschaft steht, keine Selbstregierung besitzt oder sonst
 in seiner Souveränität eingeschränkt ist.
\end{conditext}

\begin{conditext}{spanish}
```

```
 Todos los seres humanos nacen libres e iguales en dignidad y derechos
 y, dotados como están de razón y conciencia, deben comportarse fraternalmente
 los unos con los otros.
\end{conditext}

\begin{conditext}{spanish}
 Toda persona tiene los derechos y libertades proclamados en esta
 Declaración, sin distinción alguna de raza, color, sexo, idioma,
 religión, opinión política o de cualquier otra índole, origen nacional
 o social, posición económica, nacimiento o cualquier otra condición.
 Además, no se hará distinción alguna fundada en la condición política,
 jurídica o internacional del país o territorio de cuya jurisdicción
 dependa una persona, tanto si se trata de un país independiente,
 como de un territorio bajo administración fiduciaria, no autónomo
 o sometido a cualquier otra limitación de soberanía.
\end{conditext}

\begin{conditext}{italian}
 Tutti gli esseri umani nascono liberi ed eguali in dignità e diritti.
 Essi sono dotati di ragione e di coscienza e devono agire gli uni
 verso gli altri in spirito di fratellanza.
\end{conditext}
\begin{conditext}{italian}
 Ad ogni individuo spettano tutti i diritti e tutte le libertà enunciate
 nella presente Dichiarazione, senza distinzione alcuna, per ragioni
 di razza, di colore, di sesso, di lingua, di religione, di opinione
 politica o di altro genere, di origine nazionale o sociale, di ricchezza,
 di nascita o di altra condizione.
 Nessuna distinzione sarà inoltre stabilita sulla base dello statuto
 politico, giuridico o internazionale del paese o del territorio cui
 una persona appartiene, sia indipendente, o sottoposto ad amministrazione
 fiduciaria o non autonomo, o soggetto a qualsiasi limitazione di
 sovranità.
\end{conditext}

\end{document}
```

Le document source contient maintenant deux environnements `conditext` par langue concernée, l'un pour l'article premier et l'autre pour l'article 2 de la *Déclaration des droits de l'homme*.

D'après ce que nous avons vu précédemment, si par exemple le document généré doit contenir les deux premiers articles respectivement en français et en espagnol, il suffit de formuler dans le préambule les instructions

```
\setcondispace{french}
\setcondispace{spanish}
```

et l'on obtient alors dans le document généré :

Tous les êtres humains naissent libres et égaux en dignité et en droits. Ils sont

doués de raison et de conscience et doivent agir les uns envers les autres dans un esprit de fraternité.

Chacun peut se prévaloir de tous les droits et de toutes les libertés proclamés dans la présente Déclaration, sans distinction aucune, notamment de race, de couleur, de sexe, de langue, de religion, d'opinion politique ou de toute autre opinion, d'origine nationale ou sociale, de fortune, de naissance ou de toute autre situation.
De plus, il ne sera fait aucune distinction fondée sur le statut politique, juridique ou international du pays ou du territoire dont une personne est ressortissante, que ce pays ou territoire soit indépendant, sous tutelle, non autonome ou soumis à une limitation quelconque de souveraineté.

Todos los seres humanos nacen libres e iguales en dignidad y derechos y, dotados como están de razón y conciencia, deben comportarse fraternalmente los unos con los otros.

Toda persona tiene los derechos y libertades proclamados en esta Declaración, sin distinción alguna de raza, color, sexo, idioma, religión, opinión política o de cualquier otra índole, origen nacional o social, posición económica, nacimiento o cualquier otra condición.
Además, no se hará distinción alguna fundada en la condición política, jurídica o internacional del país o territorio de cuya jurisdicción dependa una persona, tanto si se trata de un país independiente, como de un territorio bajo administración fiduciaria, no autónomo o sometido a cualquier otra limitación de soberanía.

Il peut sembler peu judicieux d'avoir dans notre document source autant d'environnements `conditext` qu'il est d'articles et de versions d'article, alors que l'on pourrait ne créer qu'un environnement pour les articles en langue française, un autre pour les articles en langue anglaise etc. Cette organisation textuelle s'avère cependant nécessaire chaque fois qu'il s'agit de gérer les formes substituables au plus près.

Supposons justement qu'à partir de notre document source, pour une raison donnée, nous devions présenter dans le document généré uniquement l'article 2 de la *Déclaration des droits de l'homme* en français et en espagnol. Telles quelles, les instructions `\setcondispace` spécifiées précédemment ne le permettent pas – et le permettraient encore moins si les articles étaient regroupés par langue dans un même environnement.

En fait, l'environnement `conditext` accepte une étoile en troisième argument, que l'on utilise sous la forme atypique suivante :

```
\begin{conditext}{french}*
Tous les êtres humains naissent libres et égaux en dignité et en droits.
Ils sont doués de raison et de conscience et doivent agir les uns envers
les autres dans un esprit de fraternité.\end{conditext}

\begin{conditext}{spanish}*
Todos los seres humanos nacen libres e iguales en dignidad y derechos
y, dotados como están de razón y conciencia, deben comportarse fraternalmente
los unos con los otros.\end{conditext}
```

Dans ce cas précis, l'étoile indique que l'article 2 de la *Déclaration des droits de l'homme* en français et en espagnol doit être ignoré par les instructions respectives `\setcondispace{french}` et `\setcondispace{spanish}`.

D'une façon plus générale, l'astérisque d'un environnement `conditext` – dont la position atypique signifie que c'est la *prise en compte de la propriété de condition* qui est affectée – permet : soit de ne pas faire figurer dans le document généré une forme substituable normalement désignée pour y figurer, soit à l'inverse de faire figurer dans le document généré une forme substituable qui n'est en principe pas désignée pour y figurer.

Nous voici désormais familiarisés avec l'environnement `conditext` et la commande `\setcondispace`, deux dispositifs fondamentaux du paquet conditext. Avant d'aller plus loin et d'aborder la création de domaines de condition et de propriétés de condition, indispensable dès que l'on a un besoin d'une gestion de contenu conditionnelle spécifique, ajoutons encore deux points.

Ainsi que cela a été précisé au début de cet exemple, tant qu'aucun espace de condition n'est défini, aucun contenu d'environnement `conditext` ne figure dans le document généré. Ceci correspond en fait à un comportement par défaut proposé par le paquet conditext. Il est possible de le modifier en déclarant le paquet conditext avec l'option `defaultdisplayall` sous la forme :

```
\usepackage[defaultdisplayall]{conditext}
```

En outre – même si cette pratique n'est pas véritablement régulière, et encore moins efficace dans un document source de taille importante comportant beaucoup d'instructions `conditext` –, il est possible d'utiliser l'argument `*` propre aux environnements `conditext` *sans définir d'espace de condition*, afin de forcer « directement » l'affichage ou le non affichage d'une ou plusieurs formes substituables dans le document généré.

2 Exemple commenté d'utilisation n° 2 : un document à gestion conditionnelle multiple

Pour les besoins de ce nouvel exemple, nous partons sur l'idée que nous avons à rédiger un document préparatoire relatif à un projet d'application informatique, comportant des illustrations, avec du contenu textuel fixe et du contenu textuel différencié par département (ici, le développement, la conception de la base de données et la conception graphique). Un document devra être généré pour chacun de ces départements avec en outre, pour chaque département concerné, une version avec illustrations et un version sans illustration. Nous sommes dans le cas typique d'une gestion conditionnelle multiple.

Pour des raisons de commodité, nous représenterons le texte sous la

forme de quelques paragraphes en latin simulé. Les illustrations correspondront à des fichiers image, simulés eux aussi, et supposés être placés dans le même répertoire que celui du document source.

À part pour la gestion conditionnelle multilingue (que nous avons abordée dans l'exemple précédent) et pour une gestion conditionnelle généraliste (qui ne sera pas abordée dans ce tutoriel), conditext ne fournit aucun autre élément prédéfini, mais offre la possibilité d'en créer à volonté pour répondre aux besoins de gestion de contenu conditionnel les plus spécifiques.

En un premier temps, nous allons créer les domaines de condition et les propriétés de condition nécessaires à la gestion conditionnelle de notre exemple.

Pour rappel, un domaine de condition est une thématique fonctionnelle sous laquelle l'on peut regrouper les formes substituables qui sont liées par une même sémantique. Il nous faut deux domaines de condition, l'un pour une thématique « départements » et l'autre pour une thématique « présentation ». Nous les créons au moyen des instructions suivantes, placées dans le préambule :

```
\newcondifield{departement}
\newcondifield{presentation}
```

La forme des noms `departement` et `presentation` (de même que celle des intitulés de propriétés de condition définis plus bas) n'est peut-être pas la plus judicieuse possible mais elle suffit pour notre exemple. De manière générale, le choix de l'intitulé d'un domaine de condition ou d'une propriété de condition est laissé à l'appréciation de l'utilisateur, avec cela que l'on ne peut utiliser que des caractères alphabétiques, avec ou sans majuscules, sans espace, sans caractères accentués ou à diacritiques.

Passons maintenant aux propriétés de condition. Pour rappel, une propriété de condition permet de caractériser fonctionnellement chaque forme substituable d'un domaine de condition donné. Nous créons celles relatives aux départements au moyen des instructions suivantes :

```
\newcondiprop{departement}{develop}
\newcondiprop{departement}{bdonnees}
\newcondiprop{departement}{graphisme}
```

Il y a une propriété de condition par département, puisque dans le cadre de notre exemple, la gestion conditionnelle par départements implique que toute forme substituable du domaine de condition `departement` soit susceptible de concerner soit le développement, soit la conception de la base de données, soit encore la conception graphique. Lorsque l'on crée une propriété de condition, outre le nom de la propriété en tant que telle, il faut *systématiquement spécifier le domaine de condition* auquel l'on souhaite rattacher la propriété de condition concernée.

La gestion conditionnelle avec et sans illustration nécessite, elle, une réflexion préliminaire. L'on pourrait être tenté en effet de définir d'emblée deux propriétés de condition, l'une pour caractériser les formes substituables avec illustration, et l'autre, les formes substituables sans illustration. Or, dans ce cas, cela reviendrait à devoir par la suite définir un espace de condition testant les formes substituables avec illustration et une espace de condition testant les formes substituables sans illustration, soit un test en doublon inversé, inutile et surtout source potentielle de confusion.

De manière générale, dans le cas d'une gestion conditionnelle binaire (avec/sans etc.), une seule propriété de condition s'avère la plupart du temps suffisante. Relativement au domaine de condition `presentation`, nous ne créons donc qu'une seule propriété de condition au moyen de l'instruction suivante :

```
\newcondiprop{presentation}{illustr}
```

En résumé, la création de domaines de condition s'effectue au moyen de la commande `\newcondifield` ; celle des propriétés de condition, au moyen de la commande `\newcondiprop`.

À noter que les instructions `\newcondifield` et `\newcondiprop` doivent obligatoirement figurer avant toute autre instruction : si, dans un environnement `conditext` ou une commande telle que `\setcondispace`, l'on spécifie un domaine de condition ou une propriété de condition non créée préalablement, un message d'erreur sera déclenché lors de la compilation.

Voici maintenant le document source de notre exemple, incluant les précédentes instructions :

```
\documentclass{article}
\usepackage[french]{babel}
\usepackage{graphicx}
\usepackage{conditext}

\newcondifield{departement}
\newcondifield{presentation}
\newcondiprop{departement}{develop}
\newcondiprop{departement}{bdonnees}
\newcondiprop{departement}{graphisme}
\newcondiprop{presentation}{illustr}

\begin{document}

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus
elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur
dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer
id, vulputate a, magna. Donec vehicula augue eu neque.

\begin{conditext}[presentation]{illustr}
```

```
 \includegraphics{image1}
\end{conditext}

\begin{conditext}[departement]{bdonnees}
 Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.
 Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis
 vitae, ultricies et, tellus.

 \begin{conditext}[presentation]{illustr}
  \includegraphics{image2}
 \end{conditext}

 Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet
 magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit
 mollis.
\end{conditext}

\begin{conditext}[departement]{develop}
 Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat
 at, tincidunt tristique, libero. Vivamus viverra fermentum felis.

 \begin{conditext}[presentation]{illustr}
  \includegraphics{image3}
 \end{conditext}

 Donec nonummy pellentesque ante. Phasellus adipiscing semper elit.
\end{conditext}

\begin{conditext}[departement]{develop}
 Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae
 lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer
 adipiscing elit. In hac habitasse platea dictumst.
\end{conditext}

\begin{conditext}[departement]{graphisme}
 Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla
 a faucibus semper, leo velit ultricies tellus, ac venenatis arcu
 wisi vel nisl.

 \begin{conditext}[presentation]{illustr}
  \includegraphics{image4}
 \end{conditext}

 Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere,
 turpis lacus congue quam, in hendrerit risus eros eget felis.
\end{conditext}

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet,
laoreet vitae, arcu. Aenean faucibus pede eu ante.
```

```
\end{document}
```

(Le paquet graphicx est nécessaire pour pouvoir utiliser la commande `\includegraphics` permettant l'insertion des images dans le document.)

Remarquons avant tout que les environnements `conditext` relatifs à la gestion conditionnelle des illustrations sont *imbriqués* dans les environnements `conditext` relatifs à la gestion conditionnelle des textes à destination des différents départements.

L'on peut ainsi à volonté et dans les limites de l'utilité, imbriquer différents environnements `conditext`. La sémantique d'imbrication et son niveau de *profondeur* sont laissés à l'appréciation de l'utilisateur – la seule contrainte étant que l'on ne peut imbriquer des environnments `conditext` avec un même domaine de condition, sous risque d'effets inattendus ou indésirables.

Pour l'instant, aucun espace de condition n'est défini, ce qui signifie que si le document source est compilé tel quel, seuls, les premier et dernier paragraphes figureront dans le document généré, puisqu'ils ne sont placés dans aucun environnement `conditext` (pour rappel, par défaut, en l'absence d'un espace de condition défini, aucune forme substituable ne figure dans le document généré).

Supposons que nous ayons à générer un document comportant le texte spécifique au développement, accompagné de son illustration. Deux espaces de condition nécessitent d'être créés, l'un pour gérer le texte spécifique au développement et l'autre, pour gérer l'illustration, soit :

```
\setcondispace[departement]{develop}
\setcondispace[presentation]{illustr}
```

De la sorte, après compilation, le document généré contient, conformément à nos attentes :

> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dic- tum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.
>
> (image1)
>
> Nulla malesuada porttitor diam. Donec felis erat, congue non, vo- lutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis.
>
> (image3)
>
> Donec nonummy pellentesque ante. Phasellus adipiscing semper elit.
>
> Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vi- tae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In

hac habitasse platea dictumst.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante.

Si nous ne souhaitons aucune illustration dans le document généré, il suffit alors de retirer l'instruction `\setcondispace[presentation]{illustr}`.

Maintenant, pour une raison donnée, nous pourrions vouloir exclure l'illustration accompagnant le contenu textuel spécifique au développement, mais conserver celle qui accompagne le contenu textuel fixe. Deux solution existent dans ce cas, la première étant plus « lisible » que la seconde :

Soit l'on maintient l'instruction `\setcondispace[presentation]{illustr}` et l'on utilise l'étoile en troisième argument de l'environnement `conditext` qui gère l'illustration de la forme substituable relative au développement, sous la forme :

```
\begin{conditext}[presentation]{illustr}*
 \includegraphics{image3}
\end{conditext}
```

Soit l'on retire l'instruction `\setcondispace[presentation]{illustr}` et l'on utilise l'étoile en troisième argument de l'environnement `conditext` qui gère l'illustration du contenu textuel *fixe*, sous la forme :

```
\begin{conditext}[presentation]{illustr}*
 \includegraphics{image1}
\end{conditext}
```

De manière générale, lorsque l'on est amené à modifier individuellement la prise en compte d'un domaine et d'une propriété de condition dans une instruction `conditext`, il est préférable pour des raisons de lisibilité de le faire dans un contexte d'espace de condition défini, plutôt qu'en l'absence d'espace de condition.

Ce présent exemple nous a familiarisés avec la création de domaines de condition et de propriétés de condition, ainsi qu'avec la pratique d'instructions `conditext` imbriquées.

Avant de poursuivre avec un troisième et dernier exemple, et d'y aborder l'utilisation de la commande la plus particulière du paquet conditext, voici quelques remarques complémentaires.

Dans les documents longs à contenu conditionnel, les formes substituables peuvent être nombreuses, et par la même occasion, un certain nombre d'instructions `conditext` est susceptible de figurer dans le document source, impliquant une saisie fastidieuse lorsqu'il s'agit notamment de spécifier un même domaine de condition de multiples fois.

Certes, l'on peut utiliser, dans ce cas comme dans bien d'autres d'ailleurs, les fonctions copier-coller de l'éditeur LaTeX utilisé. Toute-

fois, le paquet conditext propose la possibilité d'utiliser un *domaine de condition implicite*, à raison d'un par document source – ce qui, outre d'exonérer l'utilisateur d'une saisie fastidieuse, rend aussi le code plus lisible dans la plupart des cas.

Pour reprendre notre exemple, si nous souhaitons ne pas avoir à saisir (et ne pas voir figurer) le domaine de condition `departement` dans les instructions `conditext`, nous pouvons ajouter l'instruction suivante dans le préambule :

```
\setimplicitcondifield{departement}
```

Désormais, toutes les instructions `conditext` permettant de gérer les formes substituables propres aux différents départements pourront avoir la forme suivante :

```
\begin{conditext}{develop}
```

ou

```
\begin{conditext}{bdonnees}
```

ou

```
\begin{conditext}{graphisme}
```

À noter que, dans un tel cas de figure, il reste toujours possible, bien qu'inutile, de spécifier `departement` explicitement.

Cela étant, dans la mesure où il ne peut exister qu'un seul domaine de condition implicite par document source, à supposer que – toujours dans le cas de notre exemple –, nous soyons amenés à devoir gérer un contenu conditionnel multilingue en plus de ceux liés aux départements et aux illustrations, nous devrons alors *spécifier* `language` *explicitement* comme domaine de condition dans les instructions `conditext` et `\setcondispace` concernées par la gestion conditionnelle multilingue.

Le domaine de condition `language` est un domaine de condition prédéfini fourni avec le paquet conditext. Par défaut, c'est lui est est spécifié comme domaine de condition implicite ; toute définition d'un domaine de condition implicite autre que `language` (comme ici avec `departement`) revient donc à modifier ce comportement par défaut.

3 Exemple commenté d'utilisation n° 3 : une mini-gestion conditionnelle

3.1

Imaginons que nous ayons à rédiger un texte utilisant un certain nombre d'expressions techniques récurrentes, comme « texte conditionnel », « domaine de condition » etc. Dans ce genre de cas, la solution très

souvent adoptée pour faciliter la saisie consiste à créer une commande d'abréviation, telle que par exemple `\tc` qui, lors de la compilation, permettra d'afficher « texte conditionnel » dans le document généré.

Or, la limite de ces commandes d'abréviation se rencontre dès que l'on est amené à devoir gérer un contenu conditionnel multilingue. Ainsi, en l'état, notre commande `\tc` n'est exploitable que dans un texte en français. À la rigueur, l'on pourrait créer autant de commandes d'abréviation pour une expression donnée qu'il y aurait de langues concernées, mais, dans un long document multilingue et pour un nombre important d'expressions différentes, l'utilisation de ces commandes (sans parler de leur administration en cas de modification) deviendrait très vite aussi fastidieuse que la saisie répétée de l'expression elle-même, qui est pourtant ce que nous souhaitons éviter.

La commande `\miniconditext` permet de remédier à cette limite.

Partons sur l'idée que nous avons pour objectif de générer, à partir d'un seul document source à contenu conditionnel multilingue – français, anglais, espagnol et italien pour les besoins de notre exemple –, différents documents contenant chacun une des versions linguistiques du texte.

Dans le cas de notre exemple, plutôt qu'une commande définie sous la forme `\NewDocumentCommand\tc{ }{texte conditionnel}` et déclinée en français, anglais, espagnol et italien, nous allons pouvoir, grâce à la commande `\miniconditext`, inscrire différents « niveaux » linguistiques à l'interne même d'une seule commande d'abréviation. Choisissons `\tcml` (`ml` pour « multilingue ») comme nom de la commande et définissons-la de la manière suivante :

```
\NewDocumentCommand\tcml{ }{
 \miniconditext{texte conditionnel}
 [conditional text]
 []
 [texto condicional]
 [testo condizionale]}
```

Précisons avant tout que le code qui précède est supposé placé dans le préambule.

Cela étant, l'on remarquera en premier lieu que l'expression en français est spécifiée en tant qu'argument obligatoire de `\miniconditext` ; que les expressions en anglais, espagnol et italien sont spécifiées en tant qu'arguments optionnels de `\miniconditext` ; et qu'un argument vide figure entre l'argument correspondant à l'expression en anglais et l'argument correspondant à l'expression en espagnol.

En fait, nous utilisons ici les modalités par défaut de `\miniconditext`, destinées à une gestion conditionnelle multilingue (nous verrons plus bas comment l'on peut utiliser `\miniconditext` dans d'autres contextes).

Chaque argument permet de spécifier une forme substituable dans une langue donnée, sur la base de 9 arguments au total – tous option-

nels sauf le premier –, correspondant respectivement et dans l'ordre, aux langues française, anglaise, allemande, espagnole, italienne, portuguaise, néerlandaise, danoise et norvégienne. Bien que non explicites dans la syntaxe de `\miniconditext`, l'on doit toujours avoir à l'esprit les correspondances entre les arguments et les langues pour pouvoir dûment spécifier les formes substituables.

L'argument vide qui figure dans le code correspond donc à la langue allemande ; il est laissé vide puisque nous n'avons pas besoin de gérer cette langue (ni les langues portuguaise, néerlandaise etc.) dans le document source. À noter que, de façon générale, tout argument optionnel non renseigné placé entre un ou plusieurs arguments renseignés doit figurer sous la forme `[]`.

Notre commande `\tcml` ainsi définie peut désormais être utilisée à volonté, comme dans le document source ci-après.

Pour constituer ce document source, nous avons utilisé un petit texte sous traductions anglaise, espagnole et italienne fournies par le site Reverso, mais il pourrait tout aussi bien s'agir de textes différents d'une langue à l'autre puisque la commande `\tcml` y serait utilisée à l'identique.

```
\documentclass{article}
\usepackage[english,spanish,italian,french]{babel}
\usepackage{conditext}

\NewDocumentCommand\tcml{}{
 \miniconditext{texte conditionnel}
 [conditional text]
 []
 [texto condicional]
 [testo condizionale]}

\setcondispace{spanish}

\begin{document}

\begin{conditext}{french}
 Vous pouvez créer un champ \tcml{} qui affiche le mot "pages" au
 lieu du mot "page" dès lors que le document comporte plusieurs pages.
 Pour afficher un \tcml{} basé sur le nombre de pages...
\end{conditext}

\begin{conditext}{english}
 You can create a \tcml{} field that displays the word "pages" instead
 of "page" if your document contains more than one page. To display
 \tcml{} based on the number of pages...
\end{conditext}
```

```
\begin{conditext}{spanish}
 Puede crear un campo de \tcml{} que muestre la palabra "páginas"
 en lugar de "página" si el documento contiene más de una página.
 Para mostrar un \tcml{} basado en el número de páginas...
\end{conditext}

\begin{conditext}{italian}
 È possibile creare un campo \tcml{} che visualizza la parola "pagine"
 invece della parola "pagina" se il documento ha più pagine. Per visualizzare
 un \tcml{} basato sul numero di pagine...
\end{conditext}

\end{document}
```

Si nous compilons en l'état, nous obtenons bien un document comportant le texte espagnol, conformément à l'instruction `\setcondispace{spanish}` figurant dans le préambule. Toutefois, « texto condicional » ne figure pas pour autant. En fait, il manque l'instruction spécifique permettant d'intégrer l'expression dans le texte.

L'on désigne quelle forme substituable spécifiée dans `\miniconditext` doit figurer dans le document généré, au moyen de la commande `\setminicondispace`. Ainsi que son nom l'indique, `\setminicondispace` sert à définir un espace de condition permettant de désigner une forme substituable parmi celles saisies dans `\miniconditext`.

Dans le cadre de notre exemple, nous devons ajouter, avant ou après `\setcondispace{spanish}` mais *après* la définition de la commande `\tcml`, l'instruction suivante :

```
\setminicondispace{spanish}
```

De la sorte, le document généré correspond désormais à :

> Puede crear un campo de texto condicional que muestre la palabra "páginas" en lugar de "página" si el documento contiene más de una página. Para mostrar un texto condicional basado en el número de páginas...

Les instructions `\setcondispace` et `\setminicondispace` modifiées de la façon suivante :

```
\setcondispace{english}
\setminicondispace{english}
```

permettront à leur tour de générer un document contenant :

> You can create a conditional text field that displays the word "pages" instead of "page" if your document contains more than one page. To display conditional text based on the number of pages...

Et ainsi de suite.

À noter que, dans une instruction `\setminicondispace` utilisée pour une gestion conditionnelle multilingue, il faut toujours désigner la forme

substituable qui doit figurer dans le document au moyen de l'appellation linguistique propre au paquet babel (ici, respectivement `spanish` et `english`).

Il n'est pas possible de définir plusieurs « mini-espaces de condition ». Si plusieurs instructions `\setminicondispace` coexistent dans le préambule, c'est la dernière qui sera prise en compte, avec le risque éventuel d'effets inattendus ou indésirables.

3.2

L'on peut également utiliser `\miniconditext` dans des contextes de gestion conditionnelle autres qu'une gestion conditionnelle multilingue. Nous allons aborder dans cette dernière partie du tutoriel, la mise en oeuvre d'une mini-gestion conditionnelle spécifique.

Cette mise en oeuvre sera aussi un peu plus complexe que toutes celles que nous avons pu aborder dans les précédents exemples, dans la mesure où elle illustre la manière dont on peut exploiter certains formatages typographiques plus ou moins sophistiqués dans le cadre d'une mini-gestion conditionnelle.

Imaginons que nous ayons à rédiger un rapport d'activité, dont le texte doit pouvoir contenir à divers endroits des commentaires à l'attention de différents services (comptabilité, communication), de manière distincte. En d'autres termes, notre objectif est, à partir d'un même document source, de pouvoir générer trois documents : l'un, qui sera à usage général ; l'autre accompagné de notes à l'attention du service comptabilité (avec usage de la couleur pourpre), le dernier accompagné de notes à l'attention service communication (avec usage de la couleur bleue).

La mini-gestion conditionnelle, avec les commandes qui lui sont propres, est à même de répondre à ce besoin, moyennant certains aménagements.

Ceci implique d'abord de bien comprendre le fonctionnement de la commande `\miniconditext`. Ainsi que nous l'avons vu précédemment, la commande `\miniconditext` possède un argument obligatoire et 8 autres arguments optionnels, chacun de ces arguments servant à spécifier une forme substituable. Il n'y a pas à proprement parler de domaine de condition lorsque l'on utilise la commande `\miniconditext` ; plus exactement, `miniconditext` est un domaine de condition en soi.

D'autre part, les propriétés de condition sont invariablement au nombre de 9, à raison d'une par argument, et ont par défaut une sémantique linguistique (respectivement et par ordre d'argument : `french`, `english`, `german`, `spanish`, `italian`, `portuguese`, `dutch`, `danish` et `norsk`).

C'est essentiellement sur les propriétés de condition que doivent porter les modifications, ainsi que nous allons le voir maintenant en détail.

Voici le document source de notre exemple ; pour des raisons de commodité, nous représentons le texte sous la forme de quelques paragraphes

en latin simulé :

```
\documentclass{article}
\usepackage[french]{babel}
\usepackage{xcolor}
\usepackage{soul}
\usepackage{conditext}

\resettingminicondiprops
\setminicondipropi{general}
\setminicondipropii{compta}
\setminicondipropiii{com}

\NewDocumentCommand\notecompta{m O{merci de bien vouloir fournir le
graphique correspondant}}{
 \miniconditext{#1}
 [\ul{#1}\textcolor{purple}{$\rightarrow$ \emph{#2}}]}

\NewDocumentCommand\notecom{m O{merci de bien vouloir fournir un image}}{
 \miniconditext{#1}
 []
 [\ul{#1}\textcolor{blue}{$\rightarrow$ \emph{#2}}]}

\setminicondispace{general}

\begin{document}
 \notecom{Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
 Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis.
 Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer
 id, vulputate a, magna. Donec vehicula augue eu neque.}

 Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.
 Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis
 vitae, ultricies et, \notecompta{tellus}. Donec aliquet, tortor sed
 accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus
 a mi. Morbi ac orci et nisl hendrerit mollis.

 Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat
 at, tincidunt tristique, libero. Vivamus viverra \notecompta{vel
 fermentum felis}[disposez-vous d'un graphique eps concernant cet
 élément ?].

\end{document}
```

(Le paquet xcolor est nécessaire pour attribuer les couleurs mauve et bleue, et le paquet soul, pour souligner certains mots.)

Nous avons d'abord préparé les propriétés de condition correspondant à notre besoin. Dans le cadre d'une mini-gestion conditionnelle, dès

lors que l'on souhaite utiliser la commande `\miniconditext` dans un contexte autre que multilingue, il faut redéfinir les propriétés de condition par défaut (de sémantique linguistique). Cette redéfinition se fait au moyen des instructions suivantes :

```
\resettingminicondiprops
\setminicondipropi{general}
\setminicondipropii{compta}
\setminicondipropiii{com}
```

Il existe une commande de redéfinition par propriété de condition, soit 9 au total, chacune identifiée par chiffres romains (de `i` à `ix`, donc) : la commande `\setminicondipropi` permet de redéfinir la propriété de condition attachée au premier argument, La commande `\setminicondipropii` permet de redéfinir la propriété de condition attachée au deuxième argument, etc.

Dans le cadre de notre exemple, seules, les trois premières propriétés de condition doivent être redéfinies – une pour chaque service concerné – en fonction desquelles seront générées les différentes versions du document

La forme des noms `general`, `compta` et `com` n'est peut-être pas la plus judicieuse possible mais elle suffit pour notre exemple. De manière générale, le choix d'un intitulé de propriété de condition est laissé à l'appréciation de l'utilisateur, avec cela que l'on ne peut utiliser que des caractères alphabétiques, avec ou sans majuscules, sans espace, sans caractères accentués ou à diacritiques.

Quel que soit le nombre de propriétés de condition à redéfinir, Il faut toujours impérativement *commencer par l'instruction* `\resettingminicondiprops`, qui réinitialise les 9 propriétés de condition avant leur redéfinition. À noter que, dans la présente version du `conditext`, la commande `\miniconditext` ne peut être utilisée qu'avec un seul jeu de propriétés de condition par document source.

Le premier argument de la commande `\miniconditext` auquel est associée la propriété de condition `general` sera une valeur par défaut, non directement exploitée ; nous aurions pu ici ne redéfinir que les première et deuxième propriétés de condition (l'une `compta` et l'autre `com`) mais l'existence d'une propriété de condition ayant une valeur par défaut, rend le code plus lisible, et permet en outre l'usage – certes non essentiel – d'un espace de condition par défaut.

Nous avons défini ensuite deux commandes similaires, nommées respectivement `\notecompta` et `\notecom`, chacune intégrant une instruction `\miniconditext`.

Le principe de fonctionnalité est le même dans les deux cas : il s'agit de pouvoir mettre en relief des mots, des groupes de mots, voire des paragraphes dans le document généré en les accompagnant de commentaires à l'attention du service comptabilité ou du service communication

selon les cas. À noter que ces commandes n'ont d'autre prétention que celle d'illustrer un usage de `\miniconditext`) et ne présentent aucune subtilité de conception particulière.

Nous déclarons chacune des commandes avec 2 arguments, l'un obligatoire (représenté dans le code par `m` et `#1`) correspondant à la forme substituable à mettre en relief, et l'autre facultatif avec valeur par défaut (représenté dans le code par `O` et `#2`) correspondant au commentaire accompagnant la forme substituable mise en relief. Pour les besoins de notre exemple, si l'argument facultatif de la commande `\notecompta` n'est pas renseigné, le commentaire « merci de bien vouloir fournir le graphique correspondant » figurera par défaut ; de même, en ce qui concerne l'argument facultatif de la commande `\notecom`, le commentaire par défaut sera « merci de bien vouloir fournir un image ».

Dans l'une et l'autre commande, le premier argument de `\miniconditext`, auquel est associée la propriété de condition `general`, sert de valeur par défaut.

La commande `\notecompta` utilise le deuxième argument de `\miniconditext`, auquel est associée la propriété de condition `compta`. Le code placé dans cet argument est intégralement consacré au formatage de la forme substituable qui doit être mise en relief (`#1`) et au formatage du commentaire (`#2`). La forme substituable sera soulignée (instruction `\ul` propre au paquet soul), et sera suivie, précédée d'une flèche, d'un commentaire en italiques (instruction `\emph`) et en mauve (instruction `\xcolor`).

La commande `\notecom` utilise le troisième argument de de `\miniconditext`, auquel est associée la propriété de condition `com`. Le code placé dans cet argument est le même que celui de `\notecompta`, avec le bleu à la place du mauve pour le texte du commentaire.

Dans le texte de notre exemple, nous avons placé une instruction `\notecom` intégrant le premier paragraphe dans son ensemble, sans commentaire explicite (c'est la formule par défaut qui figurera dans le document généré à l'attention du service communication). Le deuxième paragraphe contient une instruction `\notecompta`, intégrant un seul mot, sans commentaire explicite (c'est donc, comme dans le cas précédent, la formule par défaut qui figurera dans le document généré, à l'attention du service comptabilité). Dans le troisième et dernier paragraphe, est placée une instruction `\notecompta`, intégrant plusieurs mots et cette fois, avec un commentaire personnalisé.

Si, maintenant, nous compilons le document source sans instruction `\setminiconditext`, ou comme ici avec l'instruction `\setminicondispace{general}`, nous obtenons le document suivant, dans sa version à usage général comme attendu :

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor

lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra vel fermentum felis.

Avec l'instruction `\setminicondispace{compta}`, le document généré sera :

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus → *merci de bien vouloir fournir le graphique correspondant*. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra vel fermentum felis → *disposez-vous d'un graphique eps concernant cet élément ?*.

et avec l'instruction `\setminicondispace{com}` :

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque → *merci de bien vouloir fournir un image*.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra vel fermentum felis.

Vous voici arrivés à la fin de ce tutoriel. Son auteur espère qu'il vous aura permis de vous familiariser agréablement avec le paquet conditext, ses différentes commandes et ses possibilités. Dans ce domaine comme dans d'autres, rien ne vaut la pratique. Vous pourrez à loisir consulter la partie *Guide de l'utilisateur/référence des commandes* de cette documentation pour obtenir la description détaillée d'une commande, certaines précisions d'utilisation etc.

Que ce paquet puisse vous faciliter la tâche relativement à la gestion de contenus conditionnels sous LaTeX, voilà le simple mais sincère souhait de son auteur.