15. Gérer les répertoires en utilisant tocbasic

Le but principal du pack tocbasic est de fournir des fonctionnalités pour les auteurs de packs et classes afin de créer leurs propres tables ou listes de contenu tels que la liste des figures et la liste des tables et de leur permettre ainsi certains types de contrôles de répertoires sur d'autres classes ou packs. Ce pack fournit tocbasic ainsi que Babel (voir [BB13]) qui gère la langue à l'intérieur de toutes les tables et listes de contenu.
Comme effet secondaire, le pack peut aussi être utilisé pour définir de nouveaux flottants ou des environnements similaires, faire des ajustements à d'autres packs ou aux classes elles-mêmes.
Pour plus de détails, voir les instructions dans le chapitre 15.2, illustrées par un exemple dans l'article 15.4, qui est repris sous une forme compacte chapitre 15.5

15.1. Commandes de base

Les instructions de base sont principalement utilisées pour gérer une liste d'extensions de fichiers connus, disponibles pour les répertoires. Les entrées dans les fichiers avec ces extensions sont généralement \addtocontents ou \addxcontentsline.

En outre, des déclarations permettent d'effectuer des actions sur toutes les extensions de fichiers. Il existe également des instructions pour effectuer les réglages d'un fichier qui appartient à une extension donnée.
Typiquement une extension de fichier a aussi un propriétaire qui peut être une classe, un pack ou un identifiant qui a lui-même choisi l'auteur de la classe ou du pack que tocbasic utilise librement. Même KOMA-Script est désigné, par exemple, comme le propriétaire des flottants utilisés pour les extensions de fichier LOF et pour l'imagerie et les tables.

Pour la table des matières KOMA-Script utilise le propriétaire du nom de fichier de la classe

\ifattoclist{extension de fichier}{partie exécutée}{else}

L'instruction suivante vérifie si l'extension de fichier est déjà présente dans la liste des extensions de fichiers. Si l'extension de fichier est déjà connue de cette liste, la commande est alors mise à exécution, sinon (else), c'est la branche else qui est exécutée.

Exemple :
Vous voulez peut-être savoir si l'extension "foo" est déjà utilisé pour afficher un message d'erreur, et si vous pouvez l'utiliser ou non :

```
\Ifattoclist{foo}{%
\packError{bar}{%
l'extension d'erreur 'foo' est déjà utilisée%
}{%
chaque extension ne peut être utilisée qu'une seule fois.
\MessageBreak
la classe ou un autre pack utilise déjà
l'extension 'foo'.\MessageBreak
cette erreur est fatale!\MessageBreak,
```

```
vous ne devriez pas continuer!}%
}{%
\packInfo{bar}{en utilisant l'extension 'foo'}%
}
```

\addtotoclist [propriétaire]{extension de fichier}

Cette instruction ajoute l'extension de fichier à la liste des extensions connues. Si l'extension de fichier est déjà enregistrée, une erreur est signalée pour éviter une double utilisation de la même extension de fichier. Si l'argument optionnel [fichier propriétaire] a été spécifié, le propriétaire fixé est stocké avec cette extension de fichier.

Si l'argument optionnel est omis, alors tocbasic est essayé et sauvé en tant que fi-chier propriétaire du nom de fichier de la classe ou du pack en cours de traite-ment. Cela ne fonctionne que si \addtotoclist est appelé lors du chargement de la classe ou de le pack, mais ne fonctionne pas, quand \addtoclist est appelé plus tard en rai-son de l'utilisation d'une déclaration par l'utilisateur qui est, dans ce cas, enregis-trée en tant que propriétaire.

Notez que le fichier propriétaire d'argument vide n'est pas le même que celui de l'omission complète de l'argument optionnel y compris les crochets. Un argument vide se traduirait également par un propriétaire vide.

Exemple :
Supposons que vous vouliez l'extension de fichier «foo» pour ajouter à la liste des extensions de fichiers connus pendant que votre pack avec le nom de fichier "bar.sty" est chargé:

```
\addtotoclist{foo}
```

Cela va ajouter l'extension de fichier «foo» avec le propriétaire »bar.sty" à la liste de l'extension de fichier connu si cette extension n'est pas déjà dans la liste. Si vous avez déjà un compte extension de fichier avec la classe utilisée ou un autre pack, vous obtenez l'erreur:

```
pack tocbasic Error : file extension « foo » cannot be used twice
See the tocbasic pack documentation for explanation.
```

(pack tocbasic erreur : l'extension de fichier «foo» ne peut pas être utilisée deux fois. Voir la documentation du pack tocbasic pour l'explication).

Tapez H <retour> pour une aide immédiate. Vous obtenez :

-- File extension ‘foo’ is already used by a toc-file, while bar.sty tried to use it again for a toc-file. This may be either an incompatibility of packs, an error at a pack, or a mistake by the user --

(L'extension de fichier 'foo' est déjà utilisée par un fichier toc, alors que bar.sty essaye de l'utiliser à nouveau. L'erreur est due à l'incompatibilité des packs ou à une erreur de l'utilisateur.)

Peut-être que votre pack fournit une déclaration préparée qui génère une liste

dynamique. Dans ce cas, utiliser l'argument optionnel de **add**totoclist pour préciser le fichier propriétaire:

```
\newcommand*{\createnewlistofsomething}[1]{%
\addtotoclist[bar.sty]{#1}%
% D'autres actions pour rendre
% ce répertoire disponible
}
```

Maintenant, lorsque l'utilisateur appelle cette commande, par exemple avec

```
\createnewlistofsomething{foo},
```

l'extension de fichier «foo» est ajoutée dans le fichier propriétaire "bar.sty" à la liste des extensions connues ou alors une erreur est signalée si l'extension est déjà utilisée. Vous pouvez spécifier, comme propriétaire, qui vous voulez mais il doit être unique! Si vous êtes, par exemple, l'auteur d'un ensemble de flottants, vous pouvez spécifier que vous en êtes aussi le propriétaire au lieu de float.sty.
Dans ce cas, les options de KOMA-Script pour la liste des figures et la liste des tables devraient aussi affecter les répertoires ajoutés lors de l'utilisation de l'option de la liste des extensions connues de fichiers. En effet, KOMA-Script enregistre les extensions de fichier "LOF" pour la table des figures et "LOT" pour la liste des tables avec le fichier propriétaire "float" et les options pour ce propriétaire.

\AtAddToTocList [le fichier propriétaire]{instructions}

peut de cette manière ajouter à une liste interne des instructions qui seront toujours exécutées lors d'une extension de fichier avec le fichier propriétaire spécifié et les instructions ajoutées à la liste des extensions de fichiers connus. En ce qui concerne la procédure décrite, l'argument optionnel est dans l'état de \addtotoclist. Si l'argument optionnel est vide, les actions sont exécutées indépendamment du propriétaire, et l'extension du fichier est ajouté à la liste des extensions de fichiers connus dans ce cas. Pendant l'exécution des commandes \@currext est l'extension de fichier qui est ajoutée.

Exemple :
tocbasic utilise lui-même

```
\AtAddToTocList[]{%
  \expandafter\tocbasic@extend@babel
    \expandafter{\@currext}}
```

chaque extension de fichier est à ajouter au répertoire existant des extensions de tocbasic pour le pack de babel.

L'utilisation deux fois de \expandafter dans l'exemple est nécessaire parce que l'argument de \tocbasic@extend@babel doit absolument être étendu. Voir aussi l'explication à propos de \tocbasic@extend@babel, section 15.3.

\removefromtoclist [fichier propriétaire]{*extension*}

Cette commande supprime l'extension de la liste des extensions connues.
Si l'argument optionnel [fichier propriétaire] est spécifié, l'extension de fichier n'est

supprimée que si elle a été enregistrée pour le propriétaire spécifié.

Si le propriétaire est déterminé en omettant l'argument optionnel, la déclaration se réfère à \addtotoclist.
Si un des propriétaires spécifiés est vide, aucun test de propriétaire n'aura lieu mais l'extension de fichier est supprimée indépendamment du propriétaire.

\doforeachtocfile[le fichier propriétaire]{instructions}

Jusqu'à présent, vous avez appris à connaître les commandes, qui se traduisent par plus de sécurité dans le traitement des extensions de fichiers, mais nécessitent aussi le besoin de certains efforts supplémentaires.

Avec \doforeachtocfile vous allez enfin récolter le fruit de votre travail.

La commande fournit aux processus des instructions pour chaque extension de fichier connu du propriétaire donné. Pendant l'exécution des commandes \@currext est l'extension du fichier en cours de traitement.

Si l'argument optionnel [le propriétaire] est omis, toutes les extensions de fichiers connus et indépendants du propriétaire seront utilisés.
Si l'argument optionnel est vide, seule les extensions de fichier avec un propriétaire vide seront traitée

Exemple :
Si vous voulez afficher une liste de toutes les extensions connues dans le fichier journal, vous pouvez simplement écrire

```
\doforeachtocfile{\typeout{\@currext}}
```

Voulez-vous en revanche, que seules les extensions de fichier du propriétaire "foo", soient classifiées:

```
\doforeachtocfile[foo]{\typeout{\@currext}}
```

Les classes de KOMA-Script scrbook et scrreprt utilisent cette déclaration dans chaque répertoire pour lequel la propriété \chapteratlist est spécifiée, par la façon dont vous entrez éventuellement une distance verticale ou le titre du chapitre dans ce répertoire. Comment définir cette propriété, est décrit au paragraphe 14.2.

\tocbasicautomode

Cette commande définit la macro \starttoc de LaTeX qui permet d'ajouter l'extension du dossier spécifié à la liste des extensions de fichiers connus, à condition qu'elle ne soit pas déjà là ,

En outre, \tocbasic@starttoc est utilisé au lieu de \starttoc. Consulter, pour plus de détails au sujet de \tocbasic@starttoc et \starttoc, l'article 15.3.

Avec \tocbasicautomode, chaque table des matières ou répertoire qui sera généré en utilisant \starttoc sera au moins partiellement sous le contrôle de tocbasic. Obtenir le résultat souhaité dépend de l'extension de contrôle du pack babel pour toutes les tables des matières et des répertoires.

Néanmoins, il est préférable que l'auteur de la classe ou du pack correspondant se serve explicitement de tocbasic. Il peut alors utiliser les avantages supplémentaires fournis par le pack et décrits dans les sections suivantes.

15.2. Génération d'une table des matières ou d'un répertoire

À la section précédente, vous avez vu des commandes qui permettent de gérer une liste d'extensions de fichiers connus et d'ajouter de nouvelles extensions de cette liste. Vous avez également vu une déclaration avec laquelle vous pouvez exécuter des instructions de pour toutes les extensions connues d'un fichier propriétaire.

Dans cette section, vous verrez les commandes pour gérer le fichier correspondant à une extension ou à la liste des extensions connues.

\addtoeachtocfile [le propriétaire]{contenu}

La commande \addtoeachtocfile écrit le contenu en utilisant \addtocontents du noyau LaTeX dans chaque fichier, qui se trouve dans la liste connue des extensions de fichiers pour le propriétaire spécifié.

Si vous omettez le contenu de l'argument optionnel, il écrit dans les fichiers de chaque extension de fichier connu. Le nom du fichier réel est dans ce cas construit à partir de \jobname et de l'extension de fichier.

Pendant la rédaction du contenu, \@currext est l'extension du fichier actuellement traité.

Exemple :
Vous voulez ajouter une espace verticale d'une ligne de texte dans tous les fichiers de liste des extensions de fichiers connus.

```
\addtoeachtocfile{%
  \protect\addvspace{\protect\baselineskip}%
}%
```

Et si vous voulez le faire, seulement, pour les fichiers toc de "foo":

```
\addtoeachtocfile[foo]{%
  \protect\addvspace{\protect\baselineskip}%
}
```

Instructions de service, qui ne devrait pas être déjà développé lors de l'écriture, sont de nature à être protégées dans \addtocontents avec \protect.

\addxcontentsline{extension}{niveau}
[numéro de section]{contenu}

Cette déclaration est similaire à la commande \addcontentsline à partir du noyau de LaTeX. Cependant, elle a un argument optionnel supplémentaire pour le numéro d'entrée de la section, alors que ce dernier est spécifié, avec \addcontentsline, dans le contenu. La déclaration est utilisée pour contenir des entrées numérotées ou non numérotées dans l'extension de fichier du répertoire spécifié. Ce niveau est le nom symbolique du niveau hiérarchique et le contenu de l'entrée correspondante.

Le numéro de page est automatiquement déterminé.
La déclaration elle-même teste d'abord si une commande ajouter un niveau d'entrée de l'extension de fichier \addEbene {} est définie. Si ce est le cas, elle est utilisée pour l'entrée, avec le numéro de section donné comme argument optionnel et passée dans le contenu comme argument obligatoire.

Un exemple d'une telle instruction fournie par les classes KOMA-Script, serait \addparttocentry (voir la section 21.3).

Si la déclaration correspondante n'est pas définie, on utilise à la place la commande interne \tocbasic@addxcontentsline qui reçoit les quatre arguments pour arguments obligatoires, puis \addcontentsline manuellement pour faire l'entrée souhaitée. Pour plus de détails au sujet de \tocbasic@addxcontentsline, consulter la section 15.3.

L'avantage d'utiliser \addxcontentsline à la place de \addcontentsline est que le numéro de ligne de la propriété est conservé.

En second lieu, la forme des entrées, quant à la définition de critères spécifiques pour les extensions de niveaux et de fichiers peut être configurée.

```
\addxcontentslinetoeachtocfile[propriétaire]{niveau}
                                {numéro de classification}{contenu}
\addcontentslinetoeachtocfile[propriétaire]{niveau}{contenu}
```

Ces deux déclarations sont en relation directe avec ce qui précède déclaré dans \addxcontentsline ou défini dans le noyau de LaTeX \addcontentsline.

La différence est que ces déclarations ne se limitent pas à un seul fichier, mais agissent dans tous les fichiers de propriétaire et que le renoncement du premier argument optionnel dans les fichiers de la liste spécifiée écrit dans tous les fichiers de la liste des extensions de fichiers connus.

La commande \addxcontentsline ajoute une entrée de niveau donné à tocfile avec extension. Si {numéro} est vide ou omis, l'entrée n'aura pas de numéro d'entrée avec le texte donné. Les entrées sans nombre peuvent être alignées à gauche au nombre des rubriques numérotées de même niveau ou en retrait comme le texte des rubriques numérotées de même niveau, selon la fonction de chaîne numérique.

Exemple :
Vous êtes auteur de classe et voulez écrire l'entrée de chapitre non seulement dans le fichier toc de la table des matières, mais aussi dans tous les fichiers toc, alors que #1 est le titre qui devrait être écrit dans les fichiers. Ce peut être fait en utilisant

```
\addxcontentslinetoeachtocfile{chapter}
                                [\thechapter]{#1}
```

Dans ce cas, bien sûr, le numéro de chapitre en cours est écrit directement dans le fichier de répertoire, s'il n'était pas protégé par \protect avant expansion. Alors que la rédaction de contenu est ici comme avec \addtoeachtocfile, \@currext gère l'extension de fichier dans lequel il est écrit présentement.

Comme vous pouvez le voir, il vous suffit de remplacer \addcontentsline par

\addxcontentsline pour soutenir la fonctionnalité chaîne numérique de tocbasic.

Notez que \addxcontentsline utilise \addLevel comme entrée de poste si une telle macro existe et sinon \tocbasic@addxcontentsline. Par conséquent, vous ne pouvez pas définir une macro d'entrée d'extension \addLevel à l'aide de \addxcontentsline mais avec \tocbasic@addxcontentsline.

La commande \addxcontentslinetoeachtocfile, dans la mesure du possible, prend la priorité sur \addcontentslinetoeachtocfile puisque les extensions \addxcontentsline sont simplement à appliquer.

Il est préférable d'utiliser \addxcontentsline au lieu de \addcontentsline chaque fois que possible.

Plus d'informations sur ces améliorations et avantages peuvent être trouvées dans l'explication précédente de la commande \addxcontentsline.

\listoftoc[Titre]{Extension de fichier}
\listoftoc*{Extension de fichier}
\listofeachtoc[propriétaire]
\listofnomdesextensionsdefichier

Ces instructions permettent de créer les répertoires. La version étoilée \listoftoc* nécessite un seul argument, l'extension de fichier du fichier contenant les données dans le répertoire.

La première déclaration définit les distances verticales et horizontales s'appliquant dans les répertoires, lance les instructions exécutées avant la lecture du fichier, puis lit le fichier et effectue la fin des déclarations qui sont à réaliser après cette lecture.

Ce \listoftoc* peut être considéré comme un remplacement direct de la commande \starttoc du noyau de LaTeX.

La version de \listoftoc sans étoiles définit l'ensemble du dossier avec le titre, table des matières optionnelle, entrée facultative dans cette table et titres courants des rubriques. Si le titre d'un seul argument est vide, une chaîne vide est utilisée.

Si, toutefois, l'argument entier est omis, y compris les crochets, la commande \listofDateierweiterung(nom) est utilisée si elle est définie. Si ce n'est pas le cas, un nom de remplacement par défaut est utilisé et un avertissement émis.

La commande \listofeachtoc renvoie tous les sous-répertoires du propriétaire ou ne importe quel répertoire de toutes les extensions de noms de fichiers connus. Enfin la commande \listofnomdesextensionsdefichier doit être définie de sorte que le titre correct puisse être délivré.

Puisque l'utilisateur lui-même pourrait éventuellement utiliser \listoftoc ou encore \listofeachtoc sans argument optionnel, il est recommandé de définir d'une manière plus appropriée \listofnomdufichierdextension.

Exemple :
Supposons que vous ayez un nouveau "répertoire des algorithmes» avec l'extension de fichier "loa" et que vous en voulez l'affichage:

```
\listoftoc[liste des algorithmes]{loa}
```

le fait pour vous. Cependant, vous pensez qu'un répertoire publié sans titre suffirait:

```
\listoftoc*{loa}
```

Dans le second cas, bien sûr, une entrée facultative activée ne serait pas placée dans la table des matières.
Pour plus de détails sur la propriété de l'entrée dans la table des matières voir la commande \setuptoc.

Si auparavant

```
\newcommand*{\listofloaname}{%
répertoire des algorithmes%}
```

ne sont définis d'une manière satisfaisante.

```
\listoftoc{loa}
```

sera utilisé pour créer un répertoire avec le titre désiré.
Pour l'utilisateur, il est peut-être plus facile de mémoriser cette définition.

```
\newcommand*{\listofalgorithms}{\listoftoc{loa}}
```

comme déclaration de répertoire simple.

Depuis ce même répertoire, LaTeX ouvre un nouveau fichier pour l'écriture de sortie et peut appeler n'importe laquelle de ces déclarations pour un message d'erreur de style

```
! Pas de place pour une nouvelle écriture.
\Ch@ck ... \else \errmessage {Pas de place pour un nouveau #3}
\fi
```

utilisé lorsque aucun fichier d'écriture n'est disponible.
Remède dans ce cas : le chargement du pack scrwfile décrit dans le chapitre 14.

\BeforeStartingTOC[Extension de fichier]{instructions}
\AfterStartingTOC[Extension de fichier]{instructions}

Parfois, il est utile de vérifier si les états peuvent être directement exécutés avant de lire le fichier avec les instructions de données. La macro \BeforeStartingTOC permet simplement de lire de telles instructions, soit dans une extension de fichier unique ou dans tous les fichiers en utilisant \listoftoc*, \listoftoc ou \listofeachtoc.

Vous pouvez également réaliser des instructions après avoir lu le fichier à exécuter lorsque vous le définissez avec \AfterStartingTOC.

Pendant l'exécution des commandes \currext est l'extension de fichier du répertoire qui est lu ou qui vient d'être lu.

Un exemple d'utilisation de \AfterStartingTOC peut être trouvé dans la section 14.3.

\BeforeTOCHead[Extension de fichier]{instructions}
\AfterTOCHead[Extension de fichier]{instructions}

Ces instructions peuvent également être prédéfinies pour exécution immédiatement avant ou après le réglage de la légende si vous utilisez \listoftoc ou \listofeachtoc. En ce qui concerne l'argument optionnel et la signification de \currext, ce qui a déjà été expliqué ci-dessus dans \BeforeStartingTOC et \AfterStartingTOC s'applique.

\MakeMarkCase

Chaque fois que tocbasic met une marque sur un titre de colonne, il le fait comme argument de la commande \MakeMarkCase. Cette déclaration vise à modifier si nécessaire majuscules /minuscules du titre de la colonne. Par défaut, elle utilise une classe KOMA-Script \@firstofone, de sorte que l'argument lui-même reste inchangé. Si vous utilisez une autre classe \makeMarkCase, sera, en revanche, en opposition avec \MakeUppercase.
Si elle n'est pas déjà définie, l'instruction n'est fixée que par tocbasic. Ainsi, vous pouvez prédéfinir une classe de la manière souhaitée : elle ne sera pas redéfinie, contrairement aux habitudes, par tocbasic.

Exemple :
Vous voulez, pour une raison étrange, que les titres courants soient affichés dans votre classe en minuscules, ce qui sera également vrai pour les titres en cours d'exécution, fixés par tocbasic:

```
\let\MakeMarkcase\MakeLowercase
```

Permettez-moi une remarque à propos de \MakeUppercase : tout d'abord, cette déclaration n'est pas entièrement extensible, ce qui signifie qu'elle peut conduire à des problèmes avec d'autres déclaration en interaction.

En outre, tous les typographes conviennent que chaque phrase, chaque mise en forme de mots ou chaque passage en lettres majuscules nécessite une révision de l'espacement des lettres.

Pourtant, aucune distance fixe ne doit être utilisée. Au lieu de cela, une distance différente doit être comprises entre les différentes lettres qui ont besoin d'espaces différentes entre-elles, pour composer différentes combinaisons se comportant de différentes manières.

Dans le même temps, certaines formes de lettres, qui exigent une prise en compte, provoquent des trous dans le texte. Des packs tels que ulem ou soul peuvent en faire plus que la commande \MakeUppercase elle-même.

L'utilisation du pack de micro-caractères pour un espacement automatique est une étape, non satisfaisante, vers une solution qui n'a rien d'idéale car elle ne peut ni reconnaître, ni prendre en compte les glyphes des lettres.

Cette mise en forme implique presque toujours un travail manuel qui reste plutôt réservé aux experts et nous recommandons aux amateurs, pour le moins d'agir avec prudence et même, dans l'absolu, de s'abstenir et de ne pas l'utiliser en position de

premier plan comme dans les rubriques

\deftocheading{extension}{définition}

Le pack tocbasic comprend une définition standard des caractétistiques des en-têtes de répertoires, configurables comme décrit au \setuptoc suivante.

Mais si ces fonctionnalités ne sont pas suffisantes, une commande alternative peut être définie en utilisant \deftocheading, déclaration de titre de remplacement pour un répertoire avec une extension de fichier particulier. Elle peut contenir # 1 comme seul paramètre.
Lors de l'appel de la déclaration dans \listoftoc ou \listofeachtoc le titre est alors passé comme argument pour le répertoire.

\setuptoc {extension}{liste des Propriétés}
\unsettoc {extension}{liste des propriétés}

Avec ces deux déclarations de propriétés, une extension de fichier ou de répertoire qui lui appartient peut être réglée. La liste des attributs est une chaîne de propriétés séparées par des virgules. Le pack tocbasic exploite les propriétés suivantes:

leveldown n'utilise pas le niveau supérieur de la rubrique \part - le cas échéant \chapter ou \section si disponible - mais le premier niveau au-dessous. Cette fonctionnalité sera estimée par la commande de rubrique en interne. Toutefois, si un utilisateur spécifie la position commande avec \deftocheading, cet utilisateur est responsable du résultat conséquent de la fonctionnalité. Les classes KOMA-Script définissent cette fonctionnalité en utilisant l'option listof=leveldown pour toutes les extensions de fichiers du propriétaire float.

numberline Indique le numéro de ligne des articles écrits avec \addxcontentsline ou avec \addxcontentslinetoceachtocfile dans le répertoire, où l'argument optionnel du nombre manquant ou vide, doit comprendre une déclaration (vide) de \numberline (numéro de ligne). Cela entraîne en général le fait que ces entrées ne sont pas définies alignées à gauche avec le nombre, mais sont réglées au même niveau que le texte des entrées numérotées. Les classes de KOMA-Script définissent cette propriété avec listof=numberline pour les extensions de fichier propriétaire flottant et toc=numberline pour l'extension de fichier. En conséquence, lorsque vous utilisez l'option listof=nonumberline ou l'option toc=nonumberline, cette propriété est à chaque fois remise à zéro .

nobabel signifie que l'extension généralement utilisée automatiquement pour le changement de langue avec Babel n'est pas utilisée pour cette extension de fichier. Cette propriété ne doit être utilisée que pour les répertoires qui sont créés dans une langue dans laquelle le changement de langue ne devraient pas être inclus dans le document On la trouve également dans le pack scrwfile dont les extensions clonage sont déjà prises en charge elles-mêmes, à partir de la source de clonage.

noprotrusion
empêche la désactivation de la correction de bord optique dans les répertoires. L'alignement optique des marges est désactivé par défaut lorsque le pack type de micro ou un pack qui fournit la configuration de la commande \microtypesetup est chargé. Ainsi, lorsque l'alignement optique est souhaité dans les répertoires, cette

propriété doit être activée. Il convient toutefois de noter que la compensation optique périphérique peut être mal réglée et donner de mauvais résultats.

numbered
utilise un titre numéroté pour la table des matières ou toute autre liste et génère également une entrée dans cette même table des matières. Cette fonctionnalité donne, au titre, une valeur de numéro d'écriture par une instruction interne. D'autre part, si un utilisateur défini lui-même la rubrique commande avec \deftocheading, cet utilisateur est responsable du résultat. Les classes de KOMA-Script définissent cette fonctionnalité en utilisant l'option listof=numbered pour toutes les extensions de fichiers de l'intitulé du flottant.

onecolumn
signifie que le mode colonne LaTeX utilise automatiquement pour ce répertoire \onecolumn. Toutefois, cela se applique uniquement que si ce répertoire n'est pas déplacé avec la propriété décrite ci-dessus à un niveau hiérarchique bas.
Les classes KOMA-Script scrbook et scrreprt définissent, pour tous les flottants liés à un identifiant, cette propriété via \AtAddToTocList (ou liès à eux-mêmes en tant que propriétaire). La table des matières des deux classes, liste des figures et liste des tables est automatiquement mise en colonne. Le mode multi-colonne du pack multicol ne sera pas reconnu ou modifié par cette option.

totoc
ajoute le titre de l'annuaire à inclure dans la table des matières. Cette capacité est contrôlée par une instruction interne. Si l'utilisateur définit lui-même la fonction avec \deftocheading, il est responsable du résultat. Les classes KOMA-Script fixent cette fonctionnalité en utilisant l'option listof=totoc pour toutes les extensions de l'intitulé du flottant.

Les classes KOMA-Script utilisent aussi une autre fonctionnalités :

chapteratlist
active un code spécial pour être mis dans la liste au début d'un nouveau chapitre. Par défaut, ce code peut être soit un espace vertical ou le titre du chapitre. Voir listof en section 3.20 pour plus d'informations sur ces caractéristiques.

Exemple :
Depuis que les classes de KOMA-Script utilisent tocbasic pour la liste des figures et la liste des tables, il y a une autre façon d'empêcher la publication du sommaire de ces répertoires:

```
\unsettoc{lof}{chapteratlist}
\unsettoc{lot}{chapteratlist}
```

il suffit de définir le répertoire de fichier des chapitres dans les classes koma-script avec l'utilisation de l'extension loa, à utilisez si vous voulez que dans les classes qui utilisent \chapter comme plus haut niveau de détail dans le répertoire, une colonne soit automatiquement positionnée,
Et si vous voulez ajouter le chapitre structuration des classes de KOMA-Script à votre propre liste d'algorithmes avec l'extension de fichier "loa" des exemples précédents, vous pouvez utiliser

```
\setuptoc{loa}{chapteratlist}
```

Vous pouvez aussi, dans les classes qui utilisent \chapter comme plus haut niveau de détail dans le répertoire, forcer également le mode colonne pour cette liste avec

```
\ifundefinedorrelax{chapter}{}{%
\setuptoc{loa}{onecolumn}%
}
```

L'utilisation de \ifundefinedorrelax présume le pack scrbase (voir section 12.3). Il importe peu que le pack soit utilisé avec une autre classe. Vous ne devriez pas avoir besoin de définir cette fonctionnalité, car si l'autre classe reconnaît également la fonction de votre pack, elle l'utilisera automatiquement.

\iftocfeature{extension de fichier}{*propriétés*}{instruction}
{complément}

Cela peut être indiqué pour chaque propriété, et cette commande peut être utilisée pour tester si une fonction a été fixée pour une extension de fichier. Si c'est le cas, alors l'instruction est exécutée, sinon c'est la partie complément (else) par défaut. Cela peut être utile, par exemple, si vous définissez vos propres déclarations d'en-tête avec \deftocheading, mais que vous voulez soutenir les propriétés décrites ci-dessus totoc, numbered ou leveldown.

15.3. Commandes internes pour auteurs de classes et packs

Le pack tocbasic fournit des instructions internes, dont l'utilisation est partagée par les auteurs de classes et de packs. Ces instructions commencent toutes par \tocbasic@. Les auteurs de classes et de packs doivent exclusivement les utiliser en l'état sans chercher à les redéfinir!
Votre fonction interne peut toujours être modifiée ou prolongée, de sorte qu'une redefinition des instructions peut endommager sérieusement le fonctionnement du pack tocbasic!

\tocbasic@extend@babel{extension de fichier}

Le pack babel (voir [BB13]) ou un noyau de LaTeX qui a été étendu à la gestion des langues de babel, écrit des instructions pour changer la langue dans les fichiers avec les extensions toc, lof et lot à chaque changement du langage usuel à réaliser, soit au commencement ou à l'intérieur d'un document.

tocbasic étend ce mécanisme de sorte que l'utilisation de \tocbasic@extend@babel bénéficie également à d'autres extensions de fichier.

L'extension du fichier d'argument doit dans ce cas être totalement élargie, sinon, le sens de l'argument peut changer jusqu'à son utilisation réelle. Cette commande est normalement ajoutée, par défaut, à la liste des extensions connues de chaque fichier en utilisant \addtotoclist. Elle peut être supprimée en utilisant la fonction nobabel (voir \setuptoc, article 14.2).

Pour les extensions de fichiers toc, lof et lot, tocbasic le fait automatiquement pour éviter que la commutation ne soit entrée à plusieurs reprises dans les fichiers appropriés. Il n'y a donc aucune raison, normalement, d'appeler cette commande vous-même.

Cependant, il est concevable qu'il existe des répertoires que tocbasic ne contrôle pas, qui ne sont pas inclus dans la liste des extensions de fichiers connus, mais qui doivent, néanmoins, être manipulés par le mécanisme de changement de langue de babel.
La commande peut être utilisée explicitement, dans ce but, mais assurez-vous que cela n'est fait qu'une seule fois pour chaque extension de fichier.

\tocbasic@starttoc{prolongation}

Cette déclaration est en fait le remplacement de la commande \starttoc à partir du noyau de LaTeX. C'est la commande après \listoftoc* (voir article 15.2). Les auteurs de classes ou de packs qui veulent profiter de tocbasic devraient exploiter cette déclaration, mais il vaut mieux utiliser \listoftoc.

La commande \tocbasic@starttoc utilise \starttoc en interne, mais définit \parskip et \parindent, avant, à 0 et \parfillskip de 0 jusqu'à l'infini.

En outre, \@currext est réglé sur l'extension du fichier en cours disponible entre crochets pour être exécutée par la suite, avant et après la lecture des fichiers d'aide.

LaTeX ouvre immédiatement un nouveau fichier d'aide pour l'écriture après avoir lu ce fichier, l'utilisation de \tocbasic@starttoc peut entraîner un message d'erreur tel que :

```
Pas de place pour une nouvelle écriture
\ch@ck ... \else \errmessage {Pas de place pour un nouveau 3 #}
\fi
```

exprimé lorsque aucun fichier d'écriture n'est disponible.

La solution peut être trouvée dans ce cas dans les tiroirs du pack scrwfile décrit au chapitre 14.

\tocbasic@@before@hook
\tocbasic@@after@hook

Le crochet \tocbasic@@before@hook est exécuté immédiatement avant de lire un fichier d'aide d'une table des matières ou d'une liste de répertoire et avant même l'exécution des instructions d'une commande définies par \BeforeStartingTOC. Il est possible d'étendre ce crochet avec \g@addto@macro.

KOMA-Script utilise ces crochets pour permettre aux répertoires une adaptation dynamique et fournir le calcul automatique de la largeur de la place nécessaire par rubrique de numéros. Ils sont utilisés seulement pour les classes et, sous réserve, pour les packs.
Les utilisateurs doivent se limiter à \BeforeStartingTOC et \AfterStartingTOC. Les auteurs de pack doivent également utiliser de préférence ces deux instructions.

Les éditions à l'intérieur des deux crochets ne sont pas autorisées!

Si aucune des commandes \listofeachtoc, \listoftoc et \listoftoc* n'est utilisée pour la sortie d'une table des matières ou une autre liste, les crochets doivent être exécutés de manière explicite.

\tb@Extension de fichier@before@hook
\tb@Extension de fichier@after@hook

Ces instructions sont exécutées immédiatement après \tocbasic@@before@hook ou avant \tocbasic@@after@hook, respectivement pour chaque répertoire qui contient l'extension de fichier.

De même \tocbasic@@after@hook sera exécutée immédiatement après la lecture d'un tel fichier d'aide et avant l'exécution des instructions de \AfterStartingTOC.

Les auteurs de classes et packs ne doivent jamais les modifier!

Mais si ni \listofeachtoc, ni \listoftoc, ni \listoftoc* n'est utilisé pour l'édition d'une table des matières ou d'une liste quelconque, les crochets, s'ils sont définis, doivent être exécutés de manière explicite.

Noter que les instructions peuvent être indéfinies. Pour un test correspondant voir \scr@ifundefinedorrelax, chapitre 12.3.

\tocbasic@listhead{titre}

Cette commande est utilisée par \listoftoc et \listofeachtoc pour insérer l'instruction de définition du titre d'un répertoire ou d'une liste. Cela peut être la déclaration du pack tocbasic ou le titre d'une rubrique définie individuellement, ou encore un titre par défaut. Si vous définissez vos propres instructions, vous pouvez également utiliser \tocbasic@listhead.

Dans ce cas, avant d'utiliser \tocbasic@listhead, vous devez définir \currext comme extension de fichier du fichier d'aide correspondant.

\tocbasic@listhead@extension de fichier{titre}

Cette instruction personnalisée est utilisée dans \tocbasic@listhead pour définir l'en-tête de répertoire, les rubriques individuelles, ou les entrées optionnelles toc. Sinon il sera défini et utilisé automatiquement dans \tocbasic@listhead qui l'appelle.

\tocbasic@addxcontentsline{extension fichier}{niveau}
{numéro}{entrée}
\nonumberline

La commande \tocbasic@addxcontentsline crée une entrée de texte numérotée au toc-fichier avec l'extension donnée dans le niveau spécifié.
Une entrée est numérotée si le numéro hiérarchique de l'argument n'est pas vide, et vous pouvez utiliser un argument vide, si vous ne voulez de numéro Dans le cas d'un argument vide l'entrée d'un \nonumberline n'est préfixée par aucun argument. Sinon \numberline est utilisé avec le numéro de classification comme argument.

La déclaration nonumberline dans \listoftoc est redéfinie en fonction de la ligne de numéro de fonction (voir chapitre 15.2). Ainsi, l'attribut fixer ou effacer l'ensemble affecte déjà la prochaine exécution de LaTeX.

Notez que la fonction \tocbasic@addxcontentsline est obligatoire.

15.4. Un exemple complet

Ce chapitre présente l'exemple complet d'un environnement défini par l'utilisateur avec KOMA-Script et la liste de genre flottant en utilisant l'intégration de tocbasic.
Cet exemple utilise des commandes internes, qui ont un "@" dans leur nom.
Cela signifie que les instructions codées doivent être employées dans un pack séparé, placées entre \makeatletter et \makeatother, ou créer pour ce faire sa propre classe. Un nouvel environnement flottant, qui fournira figure ou table, est nécessaire. Cela pourrait tout simplement être fait en utilisant:

```
\newenvironment{remarkbox}{%
\@float{remarkbox}%
}{%
\end@float
}
```

Le nouvel environnement s'appelle remarkbox. Chaque figure, table, environnement de flottant a un placement par défaut. construit par certaines options de placement bien connues:

```
\newcommand*{\fps@remarkbox}{tbp}
```

Ainsi, le nouvel environnement flottant doit être placé par défaut soit en haut d'une page, au bas d'une page, ou sur une page séparée.
Les environnements flottants ont un type flottant numérique. Mais, avec le même bit actif de type flottant, ils ne peuvent pas changer leur ordre.
Figures et table utilisent normalement les types 1 et 2. Ainsi, un chiffre qui vient plus tard dans le code source d'une table, peut être édité plus tôt que la table et vice versa.

```
\newcommand*{\ftype @remarkbox}{4}
```

Le nouvel environnement flottant est de type 4, de sorte qu'il peut dépasser tables et figures et peut être dépassé par ceux-ci.

Les environnements flottants ont aussi une légende numérotée.

```
\newcounter{remarkbox}
\newcommand*{\remarkboxformat}{%
remarque~\theremarkbox\csname autodot\endcsname}
\newcommand*{\fnum@remarkbox}{\remarkboxformat}
```

Voici tout d'abord un nouveau compteur est défini, qui est indépendant des chapitres ou autres compteurs de la structure.
Il définit LaTeX égal aussi à \theremarkbox avec une édition standard et des chiffres arabes qui seront ensuite utilisés dans la définition de l'édition formatée.
La sortie formatée est à son tour définie comme environnement variable numéroté pour une utilisation dans \caption. Les environnements flottants sont identifiés par un nom de répertoire \jobname et une extension de fichier.

\newcommand*{\ext@remarkbox}{lor}
L'extension de fichier d'aide pour la liste remarkbox est "lor". C'est la définition de

l'environnement flottant. La liste des légendes de cet environnement manque mais tocbasic est utilisé pour faciliter sa mise en œuvre qui est documentée dans

\usepackage{tocbasic}
dont le préambules sera utilisé par un auteur de classe ou de pack.

\Requirepack{tocbasic} à la place.
Permet d'enregistrer l'extension de nom de fichier et un titre de répertoire pour le pack tocbasic:

\addtotoclist[float]{lor}
utilisé en tant que propriétaire du flottant, établit le lien qui appelle les options de KOMA-Script qui se réfèrent à des répertoires de flottants, même dans le nouveau répertoire. Permet de définir un titre pour ce répertoire ou un intitulé pour la liste des remarkbox

\newcommand*{\listoflorname}{liste des jeux d'instructions}
Le pack scrbase est à utiliser pour définir des titres (par défaut en anglais) dans toutes les autres langues que vous souhaitez soutenir. Une commande est nécessaire pour positionner les entrées dans la liste des jeux d'instructions. Voir chapitre 12.4.

\newcommand*{\l@remarkbox}{\l@figure}
Voici simplement les entrées dans la liste des remarkbox pour obtenir la même disposition que les entrées dans la table des figures.
C'est la solution la plus simple. Une plus explicite serait, par exemple :

\newcommand*{\l@remarkbox}{\@dottedtocline{1}{1em}{1.5em}}
Vous pouvez utiliser les entrées de chapitre qui ont un impact sur le répertoire. En outre, vous pouvez structurer la liste remarkbox selon les chapitres.

\setuptoc{lor}{chapteratlist}
La définition de cette propriété le permet lorsque vous utilisez une classe KOMA-Script et toute autre catégorie qui prend en charge cette fonctionnalité. Malheureusement, les classes standard ne sont pas incluses et ne le permettent pas. Mais cela reste suffisant. L'utilisateur peut disposer, à l'aide des options d'une classe de KOMA-Script ou de \setuptoc, de diverses formes de positions (avec ou sans table d'entrée de contenu de répertoire, avec une table des matières, avec une entrée TOC numérotée), les sélectionner et éditer le répertoire \lor{listoftoc}.

Mais une simple

\newcommand*{\listofremarkboxes}{\listoftoc{lor}}

peut rendre l'utilisation encore plus facile.

Comme vous l'avez vu, cinq commandes se réfèrent à la liste des remarques. Seules, trois d'entre-elles sont nécessaires.
La nouvelle liste de remarques prévoit une numérotation facultative de la rubrique et l'entrée non numérotée en option dans la table des matières. En option, même un niveau inférieur de la structure du document peut être utilisé pour la rubrique.
Les titres d'en-tête sont fournis avec les classes de KOMA-Script, standard, et toutes les classes qui soutiennent explicitement tocbasic. Le support de classes se réfère au nouveau répertoire, même lors du passage à un nouveau chapitre. Les

changements de la langue courante sont traités à l'intérieur de la liste des remarques comme ils le sont à l'intérieur de la liste des figures ou à l'intérieur de la liste des tables.
En outre. l'auteur d'un pack peut ajouter plus de fonctionnalités. Ce pourrait être, par exemple, des options pour cacher l'utilisation de \setuptoc à l'utilisateur.
Il est intéressant de noter que l'emploi de tocbasic peut être référencé pour décrire les fonctions correspondantes. L'avantage de cette situation est, pour l'utilisateur de pouvoir obtenir des informations sur les nouvelles fonctions fournies par tocbasic. Il doit, en outre, être en mesure de définir les caractéristiques des remarques, même sans connaissances sur une simple extension de fichier lor.

\newcommand*{\setupremarkboxes}{\setuptoc{lor}}
passe les arguments des propriétés directement à la liste \setupremarkboxes comme une liste des caractéristiques de l'extension de fichier lor.

15.5. Tout dans une seule instruction

L'exemple dans la section précédente a montré qu'il est assez facile avec tocbasic de définir vos propres flottants avec leurs propres répertoires.
L'exemple suivant montre que ce peut-être encore plus facile.

\DeclareNewTOC [liste des options]{extension de fichier}
Cette commande d'extension déclare en une seule étape une nouvelle liste, le titre de cette liste, le terme utilisé pour les entrées dans la liste, et gére l'extension de nom de fichier. En outre les options flottant et non-flottant peuvent être définies, et \caption peut être utilisé à l'intérieur de ces deux environnements .
Les fonctionnalités supplémentaires \captionabove, \captionbelow et captionbeside des classes de KOMA-Script (voir section 3.20) peuvent également être utilisées à l'intérieur de ces environments.

Extension de fichier est le nom de fichier du répertoire en cours de création, comme déjà expliqué dans la section 15.1. Cet argument est obligatoire et ne doit pas être vide! La liste d'options de l'argument est une liste séparée par des virgules, commedans toutes les listes de \KOMAoptions (voir section 2.4). Mais toutefois, ces options ne peuvent pas être réglées à l'aide \KOMAoptions!

Un aperçu de toutes les options disponibles peut être trouvé dans la table 15.2.

Exemple :
L'exemple de l'article 15.4 peut être considérablement réduit en utilisant la nouvelle déclaration \DeclareNewTOC:

```
\DeclareNewTOC[%
      type=remarkbox, %
      types=remarkboxes, %
      float, % sont à définir.
      floattype=4,%
      name=repère,%
      listname={liste des repères \"théorème}%
]{lor}
\setuptoc{lor}{chapteratlist}
```

sont définis, côté environnements remarkbox et remarkbox* le compteur remarkbox,

les commandes \theremarkbox, \remarkboxname et \remarkboxformat utilisées pour des sous-titres; les commandes \listofremarkboxes et \listremarkboxnames utilisées à la liste de remarques; et certaines commandes internes utilisées en référence à l'extension de fichier lor.

table 15.1 remarkbox :
Comparaison des environnement échantillon et figure

remarkbox	figure	Options de \DeclareNewTOC	Courte description
remarkbox	figure	type, float	Environnement particulier du type flottant
remarkbox*	figure*	type, float	Créer des colonnes sur chaque type d'environnement flottant
remarkbox	figure	type, float	Compteur utilisé par \caption
\theremarkbox	\thefigure	type, float	Déclaration à la sortie de chaque compteur
\remarkboxformat	\figureformat	type, float	Instructions pour le formatage de chaque compteur à la sortie de \caption
\remarkboxname	\figurename	type, float, name	Nom de l'étiquette utilisée dans \caption
\listofremarkboxes	\listoffigures	types, float	Déclaration à la sortie de chaque répertoire
\listremarboxname	\listfigurename	type, float, listname	En route pour l'annuaire
\fps@remarkbox	\fps@figure	type, float, floattype	numérotation séquencée de flottants
lor	lof		extension de fichier auxiliaire pour chaque répertoire

table 15.2 : Options pour la commande \DeclareNewTOC

atbegin=instructions
Les instructions seront exécutées au commencement d'un environnement flottant ou non

atend=instructions
Les instructions seront exécutées à la fin d'un environnement flottant ou non.

counterwithin=LaTeX-compteur
Si vous définissez une nouvelle figure ou un table flottant (ou non-flottant), les légendes seront numérotées et un type de compteur sera spécifié. Vous pouvez déclarer un autre compteur pour le compteur parent LaTeX. Ce compteur peut être utilisé de la même manière que, par exemple, le compteur des figures de la classe book dont dépend la numérotation des chapitres faite avec LaTeX. Dans ce cas, le compteur parent sera mis avant le compteur float qui sera réinitialisé à chaque incrémentation du compteur parent en utilisant \refstepcounter.float ou \stepcounter. Si float est défini, l'environnement flottant pour ce type sera aussi

défini. Les noms de l'environnement compteur seront respectivement la valeur de type et pour la double colonne la valeur du flottant avec addendum "*".

float
ne définit pas seulement un nouveau type de répertoire, mais également le type de flottants (voir type d'option) et le type d'entrée* (voir, figure et figure*).

floatpos= position du flottant
Chaque figure ou chaque table a un comportement de positionnement prédéfini qui peut être modifié par l'utilisation d'un argument optionnel de l'environnement du flottant. Avec cette option, le comportement de glissement est fixé pour les figures et les tables en création (voir option float). La syntaxe et la sémantique sont identiques à celles de l'argument facultatif pour les figures et les tables. Si l'option n'est pas utilisée, la valeur TBP est appliquée par défaut, comme pour les classes standard des figures et tables : Top, Bottom et Page. (la désignation de cette valeur pourrait être francisée en HBP [Haut – Bas - Page])

floattype=nombre
Chaque figure ou table a un type numérique. Les flottants dans lesquels seuls les bits différent sont liés à ce type de glissement et peuvent dépasser les uns des autres. Les types de flottants avec bits communs ne peuvent pas être réorganisés. Les flottants figures sont généralement de type1 et les flottants tables de type2 Des types autorisés existent de 1 à 31 (tous les bits sont utilisés, de sorte qu'ils puissent dépasser et être dépassés par n'importe quel autre type). Si aucun type n'est spécifié, le type le plus élevé – 16 - d'un bit est utilisé

hang=longueur
Permet de paramétrer le montant du retrait négatif des entrées pour cette liste. S'il n'est pas spécifié, 1,5 em seront utilisés comme pour l'entrée des classes standard de listes de figures ou de tables ... .

forcenames
Voir options name et listname.

hang=retrait
Entrée d'annuaire multi-lignes dans des répertoires hiérarchisés à partir de la deuxième ligne pour mise en retrait du texte en partant de la gauche. La valeur de cette entrée est également utilisée pour la position du texte dans la première ligne d'un enregistrement numéroté. Sans cette option, le texte est indenté de 1,5 em par défaut.

indent= tiret
Sous une forme hiérarchique, chaque entrée du répertoire est en retrait à partir de la gauche, de la valeur d'un tiret. Ce retrait peut être déterminé en utilisant cette option. Il est, par défaut, de 1em.

level=niveau hiérarchique
Chaque entrée dans un répertoire a un niveau hiérarchique numérique, qui peut être réglé par cette option. Si cette option n'est pas spécifiée, la valeur utilisée par défaut est de 1.

listname=titre de répertoire
Chaque répertoire titre a un titre qui peut être déterminé par cette option.

Si l'option n'est pas spécifiée, nous utiliserons un répertoire intitulé «Liste de divers types d'entrée» (voir types d'options), où le premier caractère de la dénomination du type d'entrée est converti en lettres majuscules. Il existe aussi une macro \listedetypedentrée définie avec cette valeur qui peut être modifiée à tout moment. Cette macro n'est définie que si elle ne l'est pas déjà ou que l'option supplémentaire forcename est encodée. Si aucun nom n'est donné, la valeur de type avec le premier caractère en majuscule sera utilisée.

name=nom de l'entrée
nom utilisé à la fois comme préfixe facultatif pour les entrées dans le répertoire ainsi que pour l'étiquette dans un environnement flottant (voir l'option float) ou non (voir l'option nonfloat). Sans cette option, le nom d'entrée (voir l'option type d'entrée) est utilisé, avec le premier caractère converti en majuscules. Il existe aussi une macro \nomdutypedentrée définie avec cette valeur qui peut être modifiée à tout moment. Cette macro n'est définie que si elle ne l'est pas déjà ou que l'option supplémentaire forcename est encodée.

nonfloat
S'il est défini, il détermine, non seulement un nouveau type de répertoire, mais aussi un environnement non flottant de listing (voir Option type) similaire à un flottant qui peut être utilisé dans les limites de l'environnement en cours de validité.
Le nom de l'environnement est la valeur de type avec postfix "-" owner=string, comme décrit dans les sections précédentes.

owner=propriétaire
Chaque nouveau répertoire a un propriétaire avec tocbasic (voir la section 14.1). Il peut être indiqué ici.
Si aucun propriétaire n'est spécifié, c'est le propriétaire « float » qui est utilisé avec les classes de KOMA-Script pour les répertoires des figures et des tables.

type=type d'entrée
spécifie le type d'entrées dans le répertoire approprié. type est utilisé comme nom de base pour diverses macros et éventuellement pour les environnements et les compteurs ad hoc. Il ne devrait donc contenir que des lettres. Si aucun type n'est mis en place l'extension de l'argument obligatoire sera used.types=string avec type au pluriel. Si le pluriel a été omis, l'addendum "s" sera utilisé automatiquement.

type=string
définit le type d'une nouvelle liste déclarée, il peut être utilisé, par exemple avec \listofstring.

types=type d'entrée (au pluriel)
À divers moments, la forme plurielle du type d'entrée est utilisée pour définir, par exemple, le type d'entrée par une commande \listofpluriel (\listofMehrzahl).
Si cette option n'est pas utilisée, mais une option d'entrée simple il suffit de taper « type d'entrée utilisée s ».

Si le pack doit utiliser un type flottant par défaut, l'option Optionfloattype peut être omise. En outre, si l'option nonfloat est utilisée, un environnement non-flottant remarkbox- sera également défini, qui pourra aussi être exploité dans \caption.
Pour une meilleure compréhension, une comparaison, table 14.2, des instructions et

environnements nouvellement créés, par exemple, avec les commandes remarkbox, dans un environnement approprié pour figures, et ci-dessous, un usage possible.

Exemple :

```
\begin{remarkbox}
\centering

  Le même élément devrait toujours être typographié de la même manière
  avec une même mise en forme et la même apparence.

  \caption{Première règle de typographie}
  \label{rem:typo1}
\end{remarkbox}
```

Un extrait à partir de cet environnement pourrait ressembler à ceci:

Le même élément devrait toujours être typographié de la même manière avec une même mise en forme et la même apparence.

repère 1: Première règle de typographie