13 Dépendances des packs avec scrlfile

L'introduction de LaTeX 2ε en 1994 a apporté de nombreuses innovations dans l'utilisation d'extensions de LaTeX. L'auteur du pack dispose aujourd'hui d'une série de commandes pour déterminer si un autre pack, une classe ou certaines options sont utilisées.

Le pack auteur peut charger d'autres packs ou peut spécifier si des options en cours de route prévoient d'être chargées plus tard. Cela a conduit, entre autres, à tester le chargement des packs dans n'importe quel ordre en pensant que ce serait sans importance, supposition qui, malheureusement, s'est avérée inexacte.

13.1. À propos des dépendances des packs

De plus en plus souvent des packs définissent ou redéfinissent la même commande et l'ordre dans lequel ils sont chargés devient très important, ce qui dans certains cas rend difficile la compréhension de leur comportement et n'incite pas l'utilisateur à réagir. Il est même souvent déconseillé d'intervenir lors du chargement d'un pack. ce qui n'est pas toujours facile.

Prenons un exemple simple, le chargement du pack longtable lors de l'utilisation de KOMA-Script.

Le pack longtable définit ses propres rubriques qui s'adaptent parfaitement aux classes standard mais les légendes à l'aide de documents KOMA-Script sont totalement inadaptées et ne répondent pas aux options de configuration appropriées. Afin de résoudre ce problème, les commandes du pack longtable, responsables des légendes de table, doivent être redéfinies.

Cependant, au moment où le pack longtable est chargé, la classe KOMA-Script a déjà été traitée. Jusqu'à présent, la seule façon de résoudre ce problème pour KOMA-Script était de retarder la redéfinition jusqu'au début du document avec l'aide de la macro \AtBeginDocument.

Si l'utilisateur souhaite modifier les définitions correspondante, il est possible de le faire dans le préambule du document, mais cette opération est irréaliste, car plus tard, lors de \begin{document} KOMA-Script écrasera ces définitions qu'il remplacera par les siennes. Il suffirait de retarder l'exécution jusqu'à ce que le pack longtable ait été chargé. Malheureusement, le noyau de LaTeX ne définit pas de commandes appropriées.

Le pack scrlfile apporte ici une solution.

De même, il pourrait être concevable que, avant de charger un pack, vous souhaitiez enregistrer la définition d'une macro dans un aide-macro, afin de restaurer son sens après que le pack ait été chargé.

Le pack scrlfile le permet aussi.

L'utilisation de scrlfile est pas limitée à la dépendance des packs, et les dépendances à d'autres fichiers peuvent être prises en considération. Par exemple, l'utilisateur peut être averti si le fichier non critique french.ldf a bien été chargé

Bien que ce pack intéresse, en premier lieu, les auteurs de packs, il existe des cours d'applications pour les utilisateurs ordinaires de LaTeX. Les exemples donnés dans ce chapitre sont à destination des deux groupes.

13.2. Actions avant et après le chargement

scrlfile peut exécuter des actions avant et après le chargement des fichiers.
Dans les commandes utilisées pour ce faire, des distinctions sont faites entre les fichiers généraux, les classes et packs.

\BeforeFile{fichier}{instructions}
\AfterFile{fichier}{instructions}

La macro \BeforeFile assure que les instructions sont exécutées avant le pro-chain chargement d'un fichier de macro \BeforeFile particulier..

\AfterFile fonctionne d'une manière similaire, et les instructions ne seront exécutées qu'après le chargement du fichier. Si le fichier n'est pas chargé, les instructions dans les deux cas, bien sûr, ne seront jamais exécutées.

Pour mettre en œuvre ces fonctionnalités scrlfile redéfinit la commande LaTeX bien connue \InputIfFileExists. Si cette macro n'a pas prévu la définition, scrlfile émet un avertissement.
Ceci est fait pour le cas où la déclaration dans les versions ultérieures de LaTeX est modifiée ou redéfinie par un autre pack.

La commande \InputIfFileExists est utilisée par LaTeX lorsqu'un fichier doit être chargé. Cela se produit indépendamment du fait que le fichier \LoadClass, \documentclass, \usepackage \Requirepack, \include ou des instructions similaires sont chargés.

Seul \input{foo} charge le fichier sans utiliser \InputIfFileExists.

Vous devriez toujours utiliser \input{foo}. Notez les accolades autour du nom de fichier!

\BeforeClass{classe}{instructions}
\Beforepack{pack}{instructions}

Ces deux commandes fonctionnent d'une manière similaire à \BeforeFile. La seule différence est que la classe ou le pack sont spécifiés avec leurs noms et pas avec leurs noms de fichiers. Les terminaisons « cls » et « sty » peuvent également être omises.

\AfterClass{classe}{instructions}
\AfterClass*{classe}{instructions}
\AfterClass+{classe}{instructions}
\AfterClass!{classe}{instructions}
\AfterAtEndOfClass{classe}{instructions}
\Afterpack{pack}{instructions}
\Afterpack*{pack}{instructions}
\Afterpack+{pack}{instructions}
\Afterpack!{pack}{instructions}
\AfterAtEndOfpack{pack}{instructions}

Les commandes \AfterClass et \Afterpack opèrent en grande partie de la même manière que \AfterFile avec l'infime différence que la classe ou le pack sont répertoriés avec leur nom et pas avec le nom de fichier. Les terminaisons « .cls » et « .sty » sont aussi omises ici. Une fonctionnalité supplémentaire existe dans les versions étoilées, qui exécutent les instructions non seulement à la prochaine fois que la classe correspondante ou le pack approprié est chargé, mais aussi immédiatement si la classe ou de le pack a été déjà chargé.

Dans la version Plus, les instructions sont exécutées lorsque la classe ou le pack a été entièrement chargé.

La différence avec la version étoilée n'est valable que si le chargement de la classe ou du pack a déjà commencé, mais n'est pas encore été terminé.
Toutefois, les instructions seront exécutées avant les instructions \AtEndOfClass ou \AtEndOfpack alors que le chargement de la classe ou du pack n'est pas encore terminé.

Si une classe utilise \AtEndOfClass ou un pack utilise \AtEndOfpack pour exécuter des instructions après que les fichiers de la classe ou du pack aient été chargés complètement, et si vous voulez exécuter de nouvelles instructions après les instructions de ces commandes, vous pouvez utiliser les versions avec point d'exclamation, respectivement \AfterClass! et \Afterpack!.

Dans ce cas, les instructions ne sont pas exécutées dans le contexte de la classe ou du pack spécifié. Si vous voulez, juste au cas où la classe ou le pack n'a pas été chargé, vous assurer que les instructions sont bien exécutées en dehors du contexte de la classe ou du pack spécifié, utiliser pour les classes la commande \AfterAtEndOfClass et pour les packs \AfterAtEndOfpack.

Exemple :
Dans ce qui suit, un exemple de classes et de pack est donné pour les auteurs Il montre comment KOMA-Script lui-même emploie les nouvelles commandes. La classe scrbook contient:

```
\Afterpack{hyperref}{%
\@ifpacklater{hyperref}{2001/02/19}{}{%
\ClassWarningNoLine{scrbook}{%
Vous utilisez une ancienne version du pack hyperref !
\MessageBreak%
Cette version provoque le dysfonctionnement de nombreux pilotes
\MessageBreak%
produisant \string\addchap\space un comportement étrange.
\MessageBreak%
S'il vous plaît mettez à jour hyperref au moins avec la version 6.71b}%
}%
}
```

Les anciennes versions du pack hyperref définissent une macro de la classe scrbook de telle manière que son exécution ne fonctionne pas avec les nouvelles versions de KOMA-Script.

Les nouvelles versions s'abstiennent de faire des changements si une version plus récente de KOMA-Script est identifiée.

Dans le cas où hyperref est chargé ultérieurement, scrbook effectue immédiatement après le chargement un contrôle du pack pour vérifier si la version est compatible.

Si ce ne est pas le cas, un avertissement est émis.

Les trois classes KOMA-Script utilisent la macro suivante

```
\Afterpack{caption2}{%
\renewcommand*{\setcapindent}{%
```

Après que le pack caption2 a été chargé, et seulement s'il a été chargé, KOMA-Script redéfinit sa propre commande \setcapindent.

Le code exact de la redéfinition n'est pas important pour cet exemple.

On peut noter que caption2 prend le contrôle de la commande \caption légende et que, par conséquent, la définition normale de la macro \setcapindent n'a plus aucun effet. La redéfinition améliore la collaboration avec caption2.
Il existe des exemples de l'utilisation pertinente des nouvelles instructions pour l'utilisateur de LaTeX normal.

Supposons que vous créez un document qui devrait être disponible sous forme de fichier PS, en utilisant LaTeX et dvips, ainsi que sous forme d'un fichier PDF, en utilisant pdfLaTeX. Le document devrait également contenir des liens hyper-textes.

Dans la table des matières, vous avez des entrées qui s'étendent sur plusieurs lignes, ce qui n'est pas un problème pour la méthode de pdfLaTeX, car ici des

hyperliens peuvent aussi être répartis sur plusieurs lignes, ce qui est impossible si vous utilisez un pilote de hyperref tel que dvips ou hypertex.

Dans ce cas, on souhaite utiliser linktocpage pour la configuration hyperref et la décision d'utiliser le pilote hyperref se fait automatiquement via hyperref.cfg.

Le fichier aura, par exemple, le contenu suivant:

```
\ProvidesFile{hyperref.cfg}
\@ifundefined{pdfoutput}{\ExecuteOptions{dvips}}
                                {\ExecuteOptions{pdfTeX}}
\endinput
```

Tout le reste peut maintenant être ignoré et laissé à \AfterFile

```
\documentclass{article}
\usepackage{scrlfile,frenchb}
\AfterFile{hdvips.def}{\hypersetup{linktocpage}}
\AfterFile{hyperTeX.def}{\hypersetup{linktocpage}}
\usepackage{hyperref}
\begin{document}
\listoffigures
\clearpage
\begin{figure}
    \caption{Ceci est un exemple pour une légende assez longue qui
    n'utilise pas l'argument optionnel qui permettrait d'en écrire une
    plus courte dans la liste des numéros.}
\end{figure}
\end{document}
```

Si maintenant les pilotes de hyperref, hypertex ou dvips sont utilisés, la fonction linktocpage de hyperref sera appliquée. Dans le cas de pdfLaTeX, l'option ne sera pas réglée, puisqu'un autre pilote hyperref, hpdfTeX.def, sera utilisé. Cela signifie que ni le fichier hdvips.def, ni hyperTeX.def ne seront chargés.

Mais par ailleurs, vous pouvez insérer le chargement scrlfile et les commandes \AfterFile ci-dessus peuvent être écrites dans un hyperref.cfg privé. Si vous le faites, il vaut mieux utiliser la macro \Requirepack plutôt que \usepackage (voir [Tea06]).

Les nouvelles lignes doivent être directement insérée après la ligne \ProvidesFile, et immédiatement avant l'exécution des options dvips ou pdftex.

\BeforeClosingMainAux{instructions}
\AfterReadingMainAux{instructions}

Ces commandes diffèrent dans le détail des instructions énoncées précédemment car elles permettent des actions avant et après le chargement des fichiers, pour les auteurs de pack, notamment, qui ont souvent un problème d'écriture dans le fichier auxiliaire (file.aux) après que la dernière page du document ait été éditée. Dans l'ignorance de ce problème, le code suivant est fréquemment utilisé comme suit :

```
\AtEndDocument{%
 \if@filesw\write
```

```
  \@auxout{%
   \protect\writethistoaux%
}%
 \fi
}
```

Toutefois, cette solution n'est pas idéale pour répondre au problème. Si la dernière page du document a déjà été expédiée avant \end {document}, le code ci-dessus n'entraînera pas d'écriture dans le fichier.aux.
Si vous essayez de résoudre ce nouveau problème en utilisant \immédiate juste avant \write, le problème inverse se produit: si une dernière page n'a pas été éditée avant \end {document}, \writethistoaux est écrit trop tôt dans le fichier .aux.

Une autre suggestion proposée comme solution pour ce trouble est :

```
\AtEndDocument{%
\if@filesw
\clearpage
\immediate\write\@auxout{%
\protect\writethistoaux%
}%
\fi
}
```

Cette suggestion présente un nouvel Inconvénient : la sortie de la dernière page a été forcée par le \clearpage. Après cela, des instructions comme

```
\AtEndDocument{%
\par\vspace*{\fill}%
Note a la fin du document.
\par
}
```

qui ne provoque plus un avis émis à la fin de la dernière page du document, mais plutôt à la fin de la page suivante.
Dans le même esprit, \writethistoaux serait écrit une page trop tôt dans le fichier auxiliaire.

La meilleure solution au problème serait de pouvoir tout simplement écrire dans le fichier auxiliaire immédiatement après le \clearpage final dans \end{document}, mais juste avant la fermeture du fichier aux.

C'est l'objectif de \BeforeClosingMainAux:

```
\BeforeClosingMainAux{%
\if@filesw
\immediate\write\@auxout{%
\protect\writethistoaux%
}%
\fi
}
```

C'est aussi réussi lorsque le résultat \end{document} de \clearpage ne provoque pas

une sortie d'écran avec une utilisation correcte ou dans l'ignorance des problèmes évoqués ci-dessus, et si \clearpage est inséré en argument de \AtEndDocument

Néanmoins il existe une restriction dans l'utilisation de \BeforeClosingMainAux: Vous ne devez pas utiliser une instruction composée à l'intérieur des instructions de \BeforeClosingMainAux! Si cette restriction est ignorée, le résultat en serait aussi imprévisible que les résultats des suggestions problématiques utilisant l'ascendance \AtEndDocument.

La commande \AfterReadingMainAux exécute même les instructions après la fermeture et la lecture du fichier au sein \end {document}. Ceci n'est utile que dans quelques cas très rares, par exemple, pour afficher des informations statistiques pour le fichier journal qui ne sont valides qu'après avoir lu le fichier .aux, ou pour exécuter les demandes de rediffusions supplémentaires.

Les instructions Typeset sont encore plus critiques à l'intérieur de ces instructions qu'à l'intérieur de l'argument de \BeforeClosingMainAux.

13.3. Remplacer les fichiers lors de la lecture

Dans les sections précédentes de ce chapitre, les commandes sont décrites, avec lesquelles il est possible d'exécuter des instructions avant ou après la lecture d'un fichier particulier, un pack spécifique ou une classe. C'est possible avec scrlfile mais également possible de lire dans un fichier complètement différent comme l'a demandé

\ReplaceInput {filename}{fichier de remplacement}

Cette déclaration définit une substitution du fichier avec le premier argument, nom du fichier source, par le fichier du second argument, nom du fichier de remplacement.

Si LaTeX doit ensuite charger ce fichier, c'est le fichier de remplacement qui sera appelé. La définition du fichier de remplacement affecte tous les fichiers chargés en utilisant \InputIfFileExists, que ce soit par l'utilisateur ou par LaTeX en interne.
Il est nécessaire que scrlfile redéfinisse \InputIfFileExists.

Exemple :
Vous voulez remplacer le fichier \jobname.aux par le fichier \jobname.xua. Pour ce faire, utilisez:

```
\ReplaceInput{\jobname.aux}{\jobname.xua}
```

Vous voulez maintenant remplacer \jobname.xua par \jobname.uxa :

```
\ReplaceInput{\jobname.xua}{\jobname.uxa}
```

et finalement remplacer \jobname.aux par \jobname.uxa

```
\ReplaceInput{\jobname.aux}{\jobname.uxa}
```

Et ainsi la chaîne complète de remplacement sera traitée.
Un remplacement dans le circuit:

```
\ReplaceInput{\jobname.aux}
{\jobname.xua}\ReplaceInput{\jobname.xua}
{\jobname.aux}
```

se traduirait par une erreur de taille de la sélection. Il n'est donc pas possible de définir le remplacement d'un fichier par lui-même directement ou indirectement. En théorie il serait également possible de remplacer un pack par un autre ou une classe par une autre. Mais, dans ce cas, LaTeX reconnaîtrait que le nom de fichier demandé ne correspond pas au nom de l'ensemble ou de la classe et qu'il est erroné.

Une solution à ce problème peut être trouvée ci-dessous.

```
\ReplaceClass{classe source}{classe de remplacement}
\Replacepack{pack source}{pack de remplacement}
```

Classe ou pack ne doit jamais utiliser la déclaration ci-dessus \ReplaceInput pour remplacer leur nom. Dans ce cas, LaTeX signale par un avertissement que le nom de la classe ou le nom du pack ne correspond pas.

Exemple :
Vous remplacez inconsidérément le pack fancyhdr pour le remplacer à la légère par le scrpage2 non-conforme

```
\ReplaceInput{fancyhdr.sty}{scrpage2.sty}
```

Le chargement fancyhdr entraînerait un avertissement LaTeX:
« Vous avez demandé 'scrpage2', mais le pack fournit 'fancyhdr'. »

Pour l'utilisateur, cet avertissement serait plus que déroutant, il n'a pas demandé scrpage2, mais fancyhdr qui, en fait, a été changé par scrpage2.sty en raison de votre définition de substitution.
Une solution à ce problème consiste à remplacer \ReplaceInput par l'utilisation de l'une des commandes \ReplaceClass ou \Replacepack. Il est important de noter que, comme dans \documentclass et \usepackage le nom de la classe ou du pack doit être spécifié et non pas le nom de fichier complet. Le remplacement fonctionne pour les classes chargées avec \documentclass, \LoadClassWithOptions ou \LoadClass et pour les colis chargés avec \usepackage \RequirepackWithOptions et \Requirepack.
Il convient de noter que classe et pack de remplacement sont chargés avec les mêmes options que celles initialement requises. Un pack ou une classe remplacés, et qui ne supporteraient pas une option nécessaire, conduirait aux avertissements et messages habituels d'erreurs .
Cependant, il est possible de redéfinir les options manquantes par des options de remplacement équivalentes en utilisant \BeforeClass ou \Beforepack.

Exemple :
Supposons que le pack oldfoo doit être remplacé par newfoo. Ceci est réalisé par:

```
\Replacepack{oldfoo}{newfoo}
```

Supposons que l'ancien pack oldopt fournisse une option que le nouveau pack newOpt n'a pas intègrée.

L'utilisation de

```
\Beforepack{newfoo}{%
\DeclareOption{oldopt}{%
\packInfo{newfoo}%{l'option ‘oldopt’ n'est pas supportée}%
}}%
```

définit désormais cette option pour le pack newfoo.
Cela évite de charger le pack oldfoo avec une erreur sur l'option newfoo qui ne serait pas prise en charge.

Une option NewOpt utilisée à la place de l'option oldopt peut aussi être activée par:

```
\Beforepack{newfoo}{%
\DeclareOption{oldopt}{%
\ExecuteOptions{newopt}%
}}%
```

Un dernier point : iI est même possible de spécifier que des nouveaux préréglages sont à appliquer à d'autres forfait, lors du chargement :

```
\Beforepack{newfoo}{%
\DeclareOption{oldopt}{%
\ExecuteOptions{newopt}%
}%
\PassOptionsTopack{newdefoptA,newdefoptB}
%
{newfoo}%
}
```

ou plus directement:

```
\Beforepack{newfoo}{%
\DeclareOption{oldopt}{%
\ExecuteOptions{newopt}%
}%
}% \PassOptionsTopack{newdefoptA,newdefoptB}
%
{newfoo}%
```

Pour remplacer une classe, il est nécessaire de charger scrlfile avant la classe en utilisant \Requirepack au lieu de \usepackage.

\UnReplaceInput{*Nom de fichier*}
\UnReplacepack{*pack*}
\UnReplaceClass{*Classe*}

Une définition de remplacement peut également être annulée en utilisant une de ces commandes. La définition de remplacement d'un fichier doit toujours être enlevée en utilisant \UnReplaceInput, celle d'un pack avec \UnReplacepack et la définition d'une classe avec \UnReplaceClass.
L'abolition des instructions de chargement peut entraîner que le fichier, le pack ou la classe d'origine soient chargés au lieu du fichier, du pack ou de la classe de

remplacement.

13.4. Empêcher de chargement de fichier

Surtout dans les classes et les packs utilisés au sein des entreprises ou des institutions, il est souvent constaté que de nombreux packs ne sont pas nécessaires mais néanmoins chargés parce qu'ils sont fréquemment exploités par les utilisateurs.
Si un seul de ces packs chargés automatiquement pose un problème, vous devez en quelque sorte en empêcher le chargement.
Encore une fois scrlfile offre une solution simple.

\PreventpackFromLoading [ici le code]{liste des packs}
\PreventpackFromLoading*[ici le code]{liste des packs}

Si cette déclaration est appelée avant de charger un pack avec \usepackage, \Requirepack ou \RequirepackWithOptions, le chargement du pack est efficacement empêché s'il se trouve dans la liste des packs.

Exemple : Supposons que vous travaillez dans une société où tous les documents sont créés avec la police latin moderne. La classe de la société, firmenci, contient les lignes:

```
\Requirepack[T1]{fontenc}
\Requirepack{lmodern}
```

Mais maintenant, vous souhaitez utiliser XeLaTeX ou LuaLaTeX pour la première fois. Dans ce cas, le chargement de fontenc ne serait pas une bonne suggestion et latin moderne serait la spécification de police recommandée, police par défaut du pack, alors vous voulez éviter de charger les deux packs. Cela peut être fait, en chargeant la classe dans votre propre document comme ceci:

```
\Requirepack{scrlfile}
\PreventpackFromLoading{fontenc,lmodern}
\documentclass{firmenci}
```

L'exemple ci-dessus montre aussi que scrlfile pack peut être chargé avant la classe. Dans ce cas \Requirepack doit être utilisé, plutôt que \usepackage qui n'est pas autorisé avant \documentclass.
Une liste d'pack vide ou un pack déjà chargé lancent un avertissement par \PreventpackFromLoading, tandis que \PreventpackFromLoading* écrit seulement un message correspondant dans le fichier journal.
L'argument optionnel peut être utilisé pour exécuter du code au lieu de charger le pack. Cependant, aucun autre pack ni fichier ne peuvent être chargés à la place dans le même temps. Pour remplacer un pack par un autre, voir \Replacepack à la section 12.2 Noter également que le code sera exécuté à chaque fois que vous essayez de charger le pack!

\StorePreventpackFromLoading{\commande}
\ResetPreventpackFromLoading

\commande définit avec \StorePreventpackFromLoading la liste actuelle des packs dont le chargement doit être évité.

En revanche \ResetPreventpackFromLoading réinitialise la liste des packs qui devraient être impossible à charger. Après cela, tous les packs peuvent être chargés à nouveau.

Exemple :
Pour empêcher qu'un utilisateur bloque le chargement d'un pack dont vous avez vraiment besoin dans votre propre pack, avec \PreventpackFromLoading, il faut anticiper en ajoutant ce pack à la liste des exceptions que vous établissez préalablement au chargement de votre pack:

```
\ResetPreventpackFromLoading
\Requirepack{foo}
```

Malheureusement, cela présente l'inconvénient d'effacer la liste complète de toutes les exceptions de l'utilisateur. Pour éviter cela, stockez d'abord cette liste que vous restaurerez à la fin:

```
\newcommand*{\Users@PreventList}{}%
\StorePreventpackFromLoading\Users@PreventList
\ResetPreventpackFromLoading
\Requirepack{foo}
\PreventpackFromLoading{\Users@PreventList}
```

Il est important de noter que \Users@PreventList serait également défini par la commande \StorePreventpackFromLoading même déjà défini auparavant.
En d'autres termes: \StorePreventpackFromLoading écrase une définition existante. Pour cette raison, \newcommand* a été utilisé dans cet exemple pour obtenir un message d'erreur, si \Users@PreventList a déjà été défini.
À ce stade, il est important de noter, que tout le monde qui manipule la liste en utilisant \StorePreventpackFromLoading est responsable de sa bonne capacité de restauration.
Par exemple, les éléments de la liste doivent être séparés par une virgule, ne doivent pas contenir d'espaces blancs ou de groupe entre accolades, et doivent être entièrement extensibles.
Noter également que \ResetPreventpackFromLoading ne nettoie même pas le code d'un pack, mais suspend seulement temporairement son exécution qui ne sera pas faite tant que la prévention n'est pas réactivée.

```
\UnPreventpackFromLoading{liste des packs}
\UnPreventpackFromLoading*{liste des packs}
```

Au lieu de cela, la liste des packs dont le chargement doit être empêché complètement est remise à zéro, vous pouvez également supprimer certains packs de cette liste. La version étoilée de la commande nettoie également le code.

Le code d'une liste restaurée, par exemple, lors de la réactivation de la liste des packs, ne fonctionne pas toujours.

Exemple :
En supposant que vous voulez éviter de charger un pack foo et que vous ne voulez pas d'un code déjà enregistré mais exécuter, à la place, votre propre code. Vous pouvez le faire:

```
\UnPreventpackFromLoading*{foo}
\PreventpackFromLoading[ %
  \typeout{code de substitution} %]{foo}
```

Pour la commande \UnPreventPackageFromLoading*, il n'est pas important que les packs étaient auparavant exclus du chargement.

Vous pouvez , à la place, utiliser l'instruction indirectement pour supprimer le code de tous les packs :

```
\StorePreventpackFromLoading\TheWholePreventList
\UnPreventpackFromLoading*{\TheWholePreventList}
\PreventpackFromLoading{\TheWholePreventList}
```

Dans ce cas, les packs empêchés d'être chargés, sont toujours empêchés d'être chargés, mais leur code a été nettoyé et ils ne seront pas exécutés plus longtemps.