Comment empaqueter votre paquetage* LaTeX

Scott Pakin <scott+dtx@pakin.org>†

5 novembre 2004

*C'est bien la 1re fois que je traduis *package* par *paquetage*. Je préfère, et j'utiliserai ici même, *extension* pour ce faire mais pour approcher le jeu de mot du titre originel je me voyais mal écrire « étendez votre extension » — et « emballez votre emballage » ne m'emballait pas. [Le TdS]

†Traduction en français par le TeXnicien de surface, version 1, 2008-04-16

Résumé

Ce tutoriel est destiné aux utilisateurs avancés de LaTeX 2_ε qui désireraient apprendre à créer des fichiers `.ins` et `.dtx` pour distribuer leurs fichiers de classes ou de styles faits maison.

Table des matières

1 Introduction

Prérequis Nous supposons que vous savez déjà comment *programmer* en LaTeX. C'est-à-dire que vous devriez savoir utiliser `\newcommand`, `\newenvironment` et, de préférence, un iota de TeX. Vous devriez aussi être familier du « LaTeX 2_ε for Class and Package Writers » qui est disponible sur le CTAN (`http://www.ctan.org`) et qui fait partie de la majorité des distributions de LaTeX 2_ε dans un fichier appelé `clsguide.dvi`. Enfin, vous devriez savoir installer une des extensions distribuée sous forme d'un fichier `.dtx` accompagné d'un fichier `.ins`.

Vocabulaire Un *fichier de style* (`.sty`) est d'abord une collection de définitions de macros et d'environnements. Une *extension* est constitué d'un ou plusieurs fichiers de style — p. ex. un fichier principal incorpore par `\input`s ou `\RequirePackage` plusieurs fichiers auxiliaires.

On charge une extension dans un document avec `\usepackage{`⟨*fichier .sty principal*⟩`}`. Dans la suite de ce document, on notera « ⟨*extension*⟩ » le nom de votre extension.

Motivation Les parties principales d'une extension sont le code, la documentation du code et la documentation destinée à l'utilisateur. À l'aide des programmes Doc et DocStrip, on peut combiner ces trois parties dans un seul *fichier LaTeX* (`.dtx`) *documenté*. Le principal avantage d'un fichier `.dtx` est qu'il permet d'utiliser n'importe quelle construction LaTeX pour commenter son code. Aussi, les macros, les environnements, les lignes de code, les variables, et ainsi de suite, peuvent être expliquée à l'aide de tables, figures, mathématiques et changements de polices. Le code peut être organisé en sections à l'aide des commandes de sectionnement de LaTeX. Doc facilite même la création d'un index unifié qui renvoit tant aux définitions des macros — dans le code LaTeX — qu'aux descriptions des macros — dans la documentation pour l'utilisateur. Cette insistance sur l'écriture d'un code accompagné de commentaires loquaces et proprement mis en page — ce qui revient à traiter un programme comme un livre qui décrit un ensemble d'algorithmes — est connu sous le nom de *programmation littéraire*, (*literate programming*) [4], ce que l'on a utilisé depuis les premiers jours de TeX.

Ce tutoriel vous apprendra à écrire des fichiers `.dtx` et `.ins` de base et

à les manipuler. Bien qu'il recoupe souvent le chapitre 14 du *The LaTeX Companion* [3], ce document est structuré comme un tutoriel pas à pas alors que le *The LaTeX Companion* est plus un ouvrage de référence. De plus, ce tutoriel montre comment écrire un fichier unique qui serve à la fois de fichier de documentation et de fichier pilote, ce qui est une utilisation plus courante du système Doc que celle qui consiste à utiliser des fichiers séparés.

2 Le fichier `.ins`

La première étape pour préparer une extension à être distribuée est d'écrire un *fichier d'installation* (`.ins`). Un fichier d'installation extrait le code d'un fichier `.dtx`, utilise DocStrip pour enlever les commentaires et la documentation et produit un fichier `.sty`. La bonne nouvelle est qu'un fichier `.ins` est généralement plutôt court et ne change presque pas d'une extension à une autre.

Les fichier `.ins` commence d'habitude avec des commentaires donnant

des informations à propos du *copyright* et de la licence

```
%%
%% Copyright (C) <année> by <votre nom>
%%
%% This file may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either
%% version 1.2 of this license or (at your option) any later
%% version.  The latest version of this license is in:
%%
%%    http://www.latex-project.org/lppl.txt
%%
%% and version 1.2 or later is part of all distributions of
%% LaTeX version 1999/12/01 or later.
%%
```

La Licence Publique du Projet LaTeX (*LaTeX Project Public License*) (LPPL) est la licence sous laquelle la plupart des extensions — et LaTeX lui-même — sont distribuées. Bien sûr, on peut publier son extension sous n'importe quelle licence ; la LPPL est simplement la plus commune

pour les extensions LaTeX. La LPPL précise qu'un utilisateur peut faire ce qu'il veut de l'extension — y compris la vendre et ne rien donner à l'auteur. La seule restriction est que l'auteur doit être reconnu et cité comme tel et que l'utilisateur doit changer le nom de l'extension s'il modifie quoi que ce soit ; cela pour éviter la confusion dans la gestion des versions.

L'étape suivante est de charger DocStrip :

```
\input docstrip.tex
```

`\keepsilent`

Par défaut, DocStrip produit un rapport de son activité ligne à ligne. Ces messages ne sont guère utiles aussi la plupart des gens les désactivent :

```
\keepsilent
```

`\usedir {`⟨*répertoire*⟩`}`

Un administrateur système peut déclarer le répertoire de base sous lequel tous les fichiers associés à TEX doivent être installés, p. ex. `/usr/share/texmf` — voir « `\BaseDirectory` » dans le manuel de DocStrip. Le fichier `.ins` précise l'endroit où ses fichiers devraient être installés relativement au répertoire de base. Ce qui suit est classique :

```
\usedir{tex/latex/<extension>}
```

`\preamble`
⟨*texte*⟩
`\endpreamble`

L'étape suivante est de définir le *préambule* (*preamble*) qui est un bloc de commentaire qui sera écrit au début de chacun des fichiers créés :

```
\preamble
```

```
This is a generated file.

Copyright (C) <année> by <votre nom>

This file may be distributed and/or modified under the
conditions of the LaTeX Project Public License, either
version 1.2 of this license or (at your option) any later
version.  The latest version of this license is in:

   http://www.latex-project.org/lppl.txt

and version 1.2 or later is part of all distributions of
LaTeX version 1999/12/01 or later.

\endpreamble
```

Le préambule ci-dessus ferait que ⟨*extension*⟩`.sty` commencerait comme suit :

```
%%
%% This is file `<extension>.sty',
```

```
%% generated with the docstrip utility.
%%
%% The original source files were:
%%
%% <extension>.dtx  (with options: ‘package’)
%%
%% This is a generated file.
%%
%% Copyright (C) <année> by <votre nom>
%%
%% This file may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either
%% version 1.2 of this license or (at your option) any later
%% version.  The latest version of this license is in:
%%
%%    http://www.latex-project.org/lppl.txt
%%
%% and version 1.2 or later is part of all distributions of
%% LaTeX version 1999/12/01 or later.
%%
```

`\generate {\file {⟨fichier-de-style⟩} {\from {⟨fichier-dtx⟩} {⟨balise⟩}}}`

Nous arrivons maintenant à la partie la plus importante du fichier `.ins` : la spécification des fichiers qui doivent être créer depuis le fichier `.dtx`. Ce qui suit demande à DocStrip de créer ⟨*extension*⟩`.sty` à partir de ⟨*extension*⟩`.dtx` en n'extrayant que les parties marquées « `package` » dans le fichier `.dtx` — on décrit comment marquer des parties d'un fichier `.dtx` dans la section 3.

```
\generate{\file{<extension>.sty}{\from{<extension>.dtx}{package}}}
```

`\generate` peut extraire un nombre quelconque de fichier d'un fichier `.dtx` donné. Il peut même extraire un fichier unique à partir de plusieurs fichiers `.dtx`. Voir le manuel de DocStrip pour plus de détails.

`\Msg {⟨texte⟩}`

La partie suivante du fichier `.ins` contient des commandes pour écrire à l'écran des messages à destination de l'utilisateur, lui indiquant quels

fichiers doivent être installés et lui rappelant comment produire la documentation pour utilisateur. L'ensemble suivant de commades `\Msg` est classique :

```
\obeyspaces
\Msg{*************************************************}
\Msg{*                                               *}
\Msg{* To finish the installation you have to move the  *}
\Msg{* following file into a directory searched by TeX: *}
\Msg{* Pour finir l'installation vous devez placer le   *}
\Msg{* fichier suivant dans un répertoire connu de TeX: *}
\Msg{*                                               *}
\Msg{*     <extension>.sty                           *}
\Msg{*                                               *}
\Msg{* To produce the documentation run the file        *}
\Msg{* <extension>.dtx through LaTeX.                  *}
\Msg{* Pour produire la documentation compiler         *}
\Msg{* le fichier <extension>.dtx avec LaTeX.          *}
\Msg{*                                               *}
\Msg{* Happy TeXing!  Joyeux TeXage                   *}
\Msg{*                                               *}
\Msg{*************************************************}
```

Notez l'utilisation de `\obeyspaces` pour interdire à TEX de fusionner les espaces multiples en un seul.

`\endbatchfile`

Pour finir nous disons à DocStrip que nous avons atteint la fin du fichier `.ins` :

```
\endbatchfile
```

L'annexe A.1 présente le squelette complet d'un fichier `.ins`. L'annexe A.2 lui ressemble mais contient de légères modifications requises pour produire un fichier de classe `.cls` au lieu d'un fichier de style `.sty`.

3 Le fichier `.dtx`

Un fichier `.dtx` contient à la fois le code source commenté et la documentation de l'extension pour l'utilisateur. En compilant un fichier `.dtx` avec `latex`[1], on produit la documentation pour l'utilisateur qui contient aussi généralement une version joliment mise en page du code source commenté.

Du fait de quelques facéties de Doc, le fichier `.dtx` est évalué *deux fois*. La première fois, seule une petite partie du code du pilote LaTeX est évalué. La seconde fois, les *commentaires* du fichier `.dtx` sont évalués, comme s'il n'y avait pas de « % » devant. Cela peut conduire à pas mal de confusion lorsque l'on écrit des fichiers `.dtx` et parfois même à des constructions étranges. Heureusement, une fois la structure de base du fichier `.dtx` en place, le dépot du code est plutôt sans malice. [2]

1. On peut bien entendu le compiler aussi avec `pdflatex`. [Le TdS]
2. Cela dit, un éditeur sachant traiter correctement les fichiers `.dtx` ne peut qu'être *très* utile. Emacs les reconnait grâce à un des modes fournit par AucTeX. [Le TdS]

3.1 Prologue

Les fichiers `.dtx` commencent en général par des commentaires à propos du *copyright* et de la licence :

```
% \iffalse meta-comment
%
% Copyright (C) <année> by <votre nom>
%
% This file may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either
% version 1.2 of this license or (at your option) any later
% version.  The latest version of this license is in:
%
%    http://www.latex-project.org/lppl.txt
%
% and version 1.2 or later is part of all distributions of
% LaTeX version 1999/12/01 or later.
%
% \fi
```

Les `\iffalse` et `\fi` sont indispensables sinon lors de la 2e passe de LaTeX sur le `.dtx`, les caractères `%` étant ignorés, le *copyright* et la licence serait évalués comme du code LaTeX. L'ajout de « `meta-comment` » n'est rien de plus qu'une convention pour indiquer que le commentaire est prévu pour être lu par des humains et non par Doc, DocStrip ou LaTeX.

`\NeedsTeXFormat {`⟨*nom-du-format*⟩`} [`⟨*date-de-publication*⟩`]`
`\ProvidesPackage {`⟨*non-de-l'extension*⟩`} [`⟨*info-de-publication*⟩`]`

Les quelques lignes qui suivent sont également entourées par `\iffalse` ... `\fi` afin qu'elles ne soient pas évaluées par `latex` lors de la seconde passe. Toutefois, ces lignes ne sont pas destinées au lecteur humain mais à DocStrip — ce qui explique l'absence de « `meta-comment` » :

```
% \iffalse
%<<package>>\NeedsTeXFormat{LaTeX2e}[1999/12/01]
%<<package>>\ProvidesPackage{<extension>}
%<<package>>   [<AAAA>/<MM>/<JJ> v<version> <description>]
%
```

Nous rencontrerons le `\fi` *bientôt.*

Vous souvenez-vous de la ligne du `\generate` dans le fichier `.ins`, page 12 ? Elle se terminait par la balise « `package` ». Cela dit à DocStrip d'écrire les lignes qui commence par `%<package>`dans le fichier `.sty`, en retirant, au passage, le `%<package>`. Aussi, notre fichier `.sty` commencera comme suit :

```
\NeedsTeXFormat{LaTeX2e}[1999/12/01]
\ProvidesPackage{<extension>}
   [<AAAA>/<MM>/<JJ> v<version> <description>]
```

Par exemple :

```
\NeedsTeXFormat{LaTeX2e}[1999/12/01]
\ProvidesPackage{skeleton}
   [2002/03/25 v1.0 .dtx skeleton file]
```

La ligne de `\NeedsTeXFormat` assure que l'extension ne fonctionnera pas avec une version de LaTeX 2_ε plus ancienne que celle avec laquelle

l'extension a été testée. Les chaines de caractères de la date et de la version du 2nd argument `\ProvidesPackage` sont utilisées par Doc pour définir les macros `\filedate` et `\fileversion`. Notez le format de la date ; *AAA/MM/JJ* est utilisé dans tout le monde LaTeX 2_ε et devrait être utilisé également dans votre extension.

```
\EnableCrossrefs
\CodelineIndex
\RecordChanges
\DocInput {⟨nomdefichier⟩}
```

Vient ensuite la seule partie du fichier `.dtx` dont chaque ligne n'est pas précédé par un `%` :

```
%<<*driver>>
\documentclass{ltxdoc}
\usepackage{<extension>}
\EnableCrossrefs
\CodelineIndex
\RecordChanges
```

```
\begin{document}
  \DocInput{<extension>.dtx}
\end{document}
%<</driver>>
% \fi
```

Cette strophe de code est ce que `latex` évalue lors de sa première passe à travers le fichier `.dtx`. Examinons cette strophe ligne à ligne :

1. Placer le code entre « `%<*driver>` » et « `%</driver>` » est un raccourci offert par DocStrip qui revient à placer un « `%<driver>` » au début de chaque ligne. Cela marque le code du pilote de Doc.
2. Le `\documentclass` devrait presque toujours utiliser `ltxdoc` car cela charge Doc et fournit plusieurs macros utiles pour présenter la documentation d'un programme.
3. On devrait toujours charger, avec `\usepackage`, sa propre extension. Sinon, Doc ne verra pas la commande `\ProvidesPackage` de l'extension et ne saura pas définir `\filedate` ni `\fileversion` — voir page 28. C'est également ici que l'on devrait charger avec

`\usepackage` toute autre extension nécessaire à la mise en forme de la documentation destinée à l'utilisateur.

4. `\EnableCrossrefs` dit à Doc que l'on veut qu'il construise un index pour le code — ce qui est normalement une bonne idée. Le contraire est obtenu avec `\DisableCrossrefs` qui accèlère le processus d'une quantité négligeable.
5. `\CodelineIndex` dit à Doc que l'index devrait renvoyer à la ligne de programme plutôt qu'à la page — l'effet contraire est obtenu avec `\PageIndex`. Avec `\CodelineIndex` on trouve les entrées de l'index plus facilement mais l'index est moins cohérent puisque les descriptions de macros et d'environnements sont toujours indexées par numéro de page. L'index, toutefois, commence avec une note explicative.
6. En page 27 nous verrons comment enregistrer les changements faits à chaque révision de l'extension. `\RecordChanges` demande à Doc de noter et réunir les entrées du journal des changements.
7. Il ne devrait y avoir qu'une seule commande entre `\begin{document}` et `\end{document}` : un appel à `\DocInput`

avec lequel le fichier `.dtx` s'importe lui-même. Cela permet au fichier maitre d'incorporer, par `\DocInput` plusieurs fichiers afin de produire un seul document couvrant plus d'une extension mais contenant un seul index unifié. Les fichiers maitres de documentation sont décrit page 53.

`\OnlyDescription`

Une autre commande qui apparait quelquefois dans le préambule, c.-à-d. avant le `\begin{document}`, est `\OnlyDescription` qui dit à Doc de ne composer que la documentation pour l'utilisateur et pas le code de l'extension ni les commentaires du code. Il vaut mieux en général omettre `\OnlyDescription` — ou le placer en le précédant de `%` . Un utilisateur peut toujours l'ajouter manuellement ou même mettre `\OnlyDescription` en vigueur pour *tous* les fichiers `.dtx` en ajoutant ce qui suit dans le fichier `ltxdoc.cfg` :

```
\AtBeginDocument{\OnlyDescription}
```

Le reste de cette section traite de la seconde passe de `latex` à travers le fichier `.dtx`. De ce fait, tous les exemples suivants sont préfixés par des signes de pourcentage.

`\CheckSum {`⟨*nombre*⟩`}`

Doc fournit une forme simpliste de somme de contrôle d'un document pour permettre de vérifier qu'une extension n'a pas été corrompue pendant le transport. Doc compte simplement le nombre de controbliques (*backslash*) présentes dans le code. Si ce nombre correspond à la somme de contrôle, Doc délivre un message de réussite[3] :

```
**********
* Checksum passed *
**********
```

3. Message qui signifie : « Somme de contrôle valide » [Le TdS]

Sinon, il dit ce que devrait être la somme correcte [4] :

```
! Package doc Error: Checksum not passed (<incorrect><<>><correct>).
```

Pour définir la somme de contrôle d'un fichier `.dtx`, il suffit d'ajouter un simple `\CheckSum` :

```
% \CheckSum{<nombre>}
```

Si ⟨*nombre*⟩ vaut `0` ou si le fichier `.dtx` n'utilise pas du tout de `\CheckSum`, Doc produit le message d'avertissement le suivant [5] :

```
*********************************
* This macro file has no checksum!
* The checksum should be <nombre>!
*********************************
```

4. Message qui signifie : « Erreur de l'extension doc : Somme de contrôle non valide (`<incorrect><<>><correct>`) » [Le TdS]

5. Message qui signifie : « Ce fichier de macros n'a pas de somme de contrôle ! La somme devait valoir ⟨*nombre*⟩ ! » [Le TdS]

Il est utile, pendant le développement du code, de fixer `\CheckSum{0}` afin de ne pas recevoir de message d'erreur à chaque fois que l'on compile avec `latex`. Mais il ne faudra pas oublier de remplacer « `0` » par le nombre correct avant de publier l'extension !

`\CharacterTable {`⟨*texte*⟩`}`

Le second mécanisme qu'utilise Doc pour s'assurer que le fichier `.dtx` n'a pas été corrompu est la table de caractères. Si l'on place la commande suivante *verbatim* dans son fichier `.dtx`, alors Doc s'assurera qu'aucune altération des caractères n'a eu lieu pendant le transport [6] :

```
% \CharacterTable
%  {Upper-case    \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
%   Lower-case    \a\b\c\d\e\f\g\h\i\j\k\l\m\n\o\p\q\r\s\t\u\v\w\x\y\z
%   Digits        \0\1\2\3\4\5\6\7\8\9
```

6. La table de caractères est souvent préfixée avec un double signe de pourcentage afin qu'elle soit écrite dans le fichier de style `.sty`. Cela semble inutile et c'est pourquoi nous la présentons ici avec un seul signe pourcent.

```
%   Exclamation   \!     Double quote  \"     Hash (number) \#
%   Dollar        \$     Percent       \%     Ampersand     \&
%   Acute accent  \'     Left paren    \(     Right paren   \)
%   Asterisk      \*     Plus          \+     Comma         \,
%   Minus         \-     Point         \.     Solidus       \/
%   Colon         \:     Semicolon     \;     Less than     \<<
%   Equals        \=     Greater than  \>>     Question mark \?
%   Commercial at \@     Left bracket  \[     Backslash     \\
%   Right bracket \]     Circumflex    \^     Underscore    \_
%   Grave accent  \‘     Left brace    \{     Vertical bar  \|
%   Right brace   \}     Tilde         \~}
```

Le message de réussite est le suivant [7] :

```
**************************
* Character table correct *
**************************
```

et le message d'erreur est [8] :

7. Le message signifie : « Table de caractères correcte ». [Le TdS]

8. Le message signifie : « Erreur de l'extension doc : table de caractères corrompue ». [Le TdS]

```
! Package doc Error: Character table corrupted.
```

`\changes {⟨version⟩} {⟨date⟩} {⟨description⟩}`

En page 21 nous avons appris que Doc possède un mécanisme d'enregistrement des changements réalisés dans l'extension. Cette commande est « `\changes{`⟨*version*⟩`}{`⟨*date*⟩`}{`⟨*description*⟩`}` » et on utilise fréquemment `\changes` dans la version initiale de l'extension pour en enregistrer la date de création :

```
% \changes{v1.0}{2002/03/25}{Première version}
```

Une des caractéristiques sympa de la commande `\changes` est qu'elle sait si elle a été utilisée à l'intérieur d'une définition de macro/environnement. Comme le montre la figure 1 les changements qui ont lieu au niveau le plus haut sont préfixés par « General : » et les changements internes sont préfixés par le nom de la macro ou de l'environnement dans lequel ils ont eu lieu.

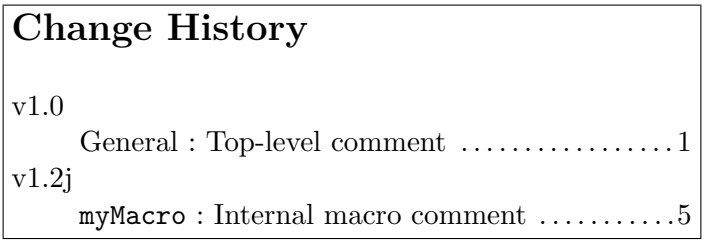

FIGURE 1 – Exemple d'historique des changements

```
\GetFileInfo {⟨fichier-de-style⟩}

\filedate
\fileversion
\fileinfo
```

Ensuite, nous demandons à Doc d'analyser la commande `\ProvidesPackage` — page 17 — en appelant les trois composantes de l'argument de `\ProvidesPackage` avec « `\filedate` », « `\fileversion` » et « `\fileinfo` » :

```
% \GetFileInfo{<extension>.sty}
```

L'exemple de `\ProvidesPackage` montré page 18 serait analysé comme suit :

`\filedate`	≡	2002/03/25
`\fileversion`	≡	v1.0
`\fileinfo`	≡	.dtx skeleton file

`\DoNotIndex {`⟨*nom-de-macro* , ... ⟩`}`

Lorsqu'il produit un index, Doc indexe normalement *toutes* les séquences de contrôle c.-à-d. les symboles et les mots précédés d'une contreoblique dans le code. Le problème à ce niveau d'automatisme est que de nombreuses séquences de contrôles sont sans intérêt pour la compréhension du code. Par exemple, un lecteur n'a certainement pas envie de voir tous les endroits où on utilise `\if` — ni `\the`, `\let` ni `\begin` ni aucune d'un nombre important d'autres séquences de contrôle.

Comme son nom l'indique, la commande `\DoNotIndex` fournit à Doc une liste de séquences de contrôle qui ne doivent pas être indexées. On peut utiliser `\DoNotIndex` autant de fois qu'on le souhaite et elle accepte autant de noms de séquences de contrôle par invocation que l'on souhaite :

```
% \DoNotIndex{\#,\$,\%,\&,\@,\\,\{,\},\^,\_,\~,\ }
% \DoNotIndex{\@ne}
% \DoNotIndex{\advance,\begingroup,\catcode,\closein}
% \DoNotIndex{\closeout,\day,\def,\edef,\else,\empty,\endgroup}
```

⋮

3.2 Documentation pour l'utilisateur

Nous pouvons enfin commencer à écrire la documentation destinée à l'utilisateur. On trouvera fréquemment le début suivant :

```
% \title{L'extension \textsf{<extension>}\thanks{Ce document
%   correspond à la version~\fileversion de \textsf{<extension>},
%   datée du~\filedate.}}
% \author{<votre nom> \\ \texttt{<votre addresse électronique>}}
%
% \maketitle
```

On peut certainement faire preuve de plus de créativité dans le titre mais il est d'usage de présenter le nom d'une extension avec `\textsf` et d'utiliser `\thanks` pour préciser la date et la version de l'extension. Cela apporte un des avantages de la programmation littéraire : à chaque fois que l'on change la version de l'extension — le 2e argument optionnel de `\ProvidesPackage` — la documentation destinée à l'utilisateur est mise à jour de façon cohérente. Bien entendu, on doit s'assurer *à la main* que la documentation décrit toujours précisément l'extension mise à jour.

Il faut écrire la documentation pour l'utilisateur comme si l'on écrivait un document LaTeX quelconque mais en préfixant chaque ligne avec un

« `%` ».[9] Notez que la classe de document `ltxdoc` dérive de la classe `article` et donc que le premier niveau de sectionnement est obtenu avec `\section` et pas avec `\chapter`.

`\DescribeMacro {`⟨*macro*⟩`}`
`\DescribeEnv {`⟨*environment*⟩`}`

Doc fournit une paire de macros pour aider à mettre en forme la documentation pour l'utilisateur. Si on place « `\DescribeMacro{`⟨*macro*⟩`}`[10] » dans un paragraphe, Doc fera apparaitre un ⟨*macro*⟩ dans la marge pour la rendre plus visible. Doc ajoutera également ⟨*macro*⟩ dans l'index et formatera le numéro de page pour indiquer que c'est là que la macro est décrite — ce qui s'oppose à l'endroit du code source où elle est définie.

9. Là encore, sans vouloir insister outre mesure, un bon éditeur bien adapté à la tache semble indispensable. Je pense à Emacs avec AucTeX mais on l'aura deviné. [Le TdS]

10. ⟨*macro*⟩ doit commencer par la contreoblique.

La commande `\DescribeEnv` est l'analogue pour décrire un environnement. On peut utiliser ces deux macros `\DescribeMacro` et `\DescribeEnv` plusieurs fois dans un paragraphe.

`\marg {`⟨*argument*⟩`}`
`\oarg {`⟨*argument*⟩`}`
`\parg {`⟨*argument*⟩`}`
`\meta {`⟨*texte*⟩`}`

La classe de documents `ltxdoc` fournit trois commandes pour faciliter la saisie de la syntaxe d'une macro ou d'un environnement — voir table 1. `\marg` met en forme les arguments obligatoires, `\oarg` les arguments optionnels et `\parg` les arguments d'un environnement de type `picture`. Ces trois commandes utilisent `\meta` pour présenter l'argument lui-même. `\meta` est aussi utile par lui-même. Par exemple, « `Ceci requiert une \meta{dimension}` » est présenté sous la forme « Ceci requiert une ⟨*dimension*⟩ ».

En plus de fournir ces commandes, Doc facilite la mise en forme de la descriptions de macros en chargeant automatique l'extension `shortvrb`.

TABLE 1 – Commandes de mise en forme d'argument

Commande	Résultat
`\marg{texte}`	{⟨*texte*⟩}
`\oarg{texte}`	[⟨*texte*⟩]
`\parg{texte}`	(⟨*texte*⟩)

Cette extension `shortvrb` permet d'utiliser `|... |` comme raccourci commode de `\verb| ... |`. Par exemple, « `|\mamacro| \oarg{pos} \marg{largeur} \marg{texte}` » est typographié comme suit :

```
\mamacro [<pos>] {<largeur>} {<texte>}
```

Tout comme `\verb`, le raccourci `| ... |` ne marche pas dans `\footnote` ou autres macros fragiles.

3.3 Code et commentaires

`\StopEventually {`⟨*texte*⟩`}`
`\Finale`

Le code source de l'extension est délimité par `\StopEventually` et `\Finale`. Notez que la macro `\CheckSum` — page 23 — ne tient compte que du code source de l'extension. `\StopEventually` prend un argument, un bloc de texte qui sera typographié après le code. Si on spécifie `\OnlyDescription` — page 22 — alors rien ne sera typographié de ce qui suit `\StopEventually` y compris ce qui suivrait éventuellement la macro `\Finale`. C'est grace au paramètre ⟨*texte*⟩ de `\StopEventually` que l'on peut fournir un morceau de texte qui sera typographié, que le code le soit ou non. Ce paramètre contient souvent une bibliographie ou quelques unes des commandes qui suivent.

`\PrintChanges`
`\PrintIndex`

`\PrintChanges` produit une section non numérotée intitulée « Historique des changements » — voir la figure 1 page 28. Cette section regroupe le contenu de toutes les macros `\changes` utilisées dans le fichier `.dtx` en une seule liste de modifications classées par version. Cela permet de trouver facilement ce qui a changé d'une version à l'autre.

`\PrintChanges` utilise le mécanisme de glossaire de LaTeX. La compilation par `latex` de ⟨*extension*⟩`.dtx` produit un fichier ⟨*extension*⟩`.glo` contenant les données de l'historique des changements. Pour faire apparaitre effectivement l'historique dans le fichier image (pdf ou dvi), l'utilisateur doit faire appel à `makeindex` comme suit :

```
makeindex -s gglo.ist -o <extension>.gls <extension>.glo
```

`\PrintIndex` produit une section non numérotée intitulée « Index ». L'index contient automatiquement une entrée pour chaque macro et chaque environnement utilisé, défini ou décrit dans le document. Tous les environnements sont, de plus, énumérés sous l'entrée « *environments*[11] ».

11. À moins que l'on ait pris les précautions nécessaires pour franciser les parties

La table 2 illustre les différentes manières dont les entrées sont typographiées. Dans cette table, « 27 » renvoie à une page et « 123 » à une ligne du code.[12] Notez que les définitions et les utilisations de macros et d'environnements ne sont incorporées dans l'index que si le document contient une partie de code c.-à-d. si l'on n'a *pas* déclaré `\OnlyDescription`.

TABLE 2 – Présentation des entrées dans l'index

Entrée	Fonction	Présentation dans l'index
Macro	Utilisée	`\maMacro` 123
Macro	Définie	`\maMacro` ..... <u>123</u>
Macro	Décrite	`\maMacro` *27*
Environnement	Défini	`monEnv` (environment) ..... <u>123</u>
Environnement	Décrit	`monEnv` (environment) *27*
Autre, c.-à-d. un	`\index` explicite	`monEntrée` 27

automatiques du document. [Le TdS]

12. Si `\CodelineIndex` — page 19 — n'était pas en vigueur « 123 » renverrait également à une page.

La présentation par défaut d'une entrée placée explicitement par une commande `\index` utilise un numéro de page en romain [13]. Cela prête à confusion puisque c'est également la présentation des numéros de ligne du code source de l'extension. La solution est de préciser une mise en forme « `usage` » dans la macro `\index`

```
\index{explicit indexing|usage}
```

La compilation par `latex` de ⟨*extension*⟩`.dtx` produit le fichier d'index brut ⟨*extension*⟩`.idx`. Pour créer le fichier d'index effectif ⟨*extension*⟩`.ind`, l'utilisateur appellera `makeindex` comme suit :

```
makeindex -s gind.ist -o <extension>.ind <extension>.idx
```

L'indexation du code est une plus-value rendue possible par la programmation littéraire. Elle ne nécessite presque aucun effort supplémentaire

13. La police pas le nombre. [Le TdS]

et aide grandement les mainteneurs du code à trouver les définitions des macros et à voir à quelles autres macros une extension fait appel.

```
\begin{macrocode}
⟨code⟩
\end{macrocode}
```

Les fragments de code placé entre `\begin{macrocode}` et `\end{macrocode}` sont extraits et placés *verbatim* dans le fichier `.sty`. Lorsqu'ils sont typographiés, les lignes des fragments de code sont précédées d'un numéro, obtenu par un compteur, pour permettre de faire aisèment référence à une ligne précise. Voici quelques points dont il faut se souvenir en ce qui concerne l'environnement `macrocode` :

1. Il doit y avoir *exactement* quatre espaces entre le « `%` » et le « `\begin{macrocode}` » ou « `\end{macrocode}` ». Autrement, Doc ne détectera pas la fin du fragment de code. [14]

14. Petite note : seul `\end{macrocode}` a besoin de cet espacement précis et, de plus, uniquement lors de la composition de la documentation. Toutefois il est d'une

2. Les lignes de code à l'intérieur de `\begin{macrocode}`... `\end{macrocode}` ne devrait pas commencer par `%`. Le code serait, dans le cas contraire, écrit exactement comme il se présente au fichier `.ins`, sans suppression des `%`.

Ce qui suit est un exemple de fragment de code. Il se trouve être une définition complète d'une macro mais ce n'est pas obligatoire, tout fragment de code LaTeX peut figurer à l'intérieur d'un environnement `macrocode`.

```
%    \begin{macrocode}
\newcommand{\MaMacro}{Ceci est
  une macro \LaTeX{}.}
%    \end{macrocode}
```

Doc typographie le fragment de code précédent comme suit

```
1 \newcommand{\MaMacro}{Ceci est
2   une macro \LaTeX.}
```

bonne pratique d'utiliser également « `%␣␣␣␣` » pour le `\begin{macrocode}`.

Notez que chaque ligne est numérotée de manière unique pour tout le programme c.-à-d. que la numérotation n'est pas remise à zéro à chaque changement de page. Si on utilise `\PrintIndex` dans un fichier `.dtx` contenant la définition précédente de `\MaMacro`, l'index contiendra automatiquement une entrée pour `\newcommand`, `\MaMacro` et `\LaTeX` à moins que l'une de ces commandes n'ait été rendue inindexable par `\DoNotIndex`.

```
\begin{macro}{⟨macro⟩}
    ⋮
\end{macro}

\begin{environment}{⟨environnement⟩}
    ⋮
\end{environment}
```

On utilise les environnement `macro` et `environment` pour délimiter une définition complète d'une macro et d'un environnement respectivement.

Ces environnements contiennent en général un ou plusieurs environnements `macrocode` entrecoupés de documentation du code. Ce qui suit est une version plus complète de l'exemple de `macrocode` présenté page 40.

```
% \begin{macro}{\MaMacro}
% Nous définissons une macro banale, |\MaMacro|, pour
% illustrer l'utilisation de l'environment |macro|.
%    \begin{macrocode}
\newcommand{\MaMacro}{Ceci est
  une macro \LaTeX.}
%    \end{macrocode}
% \end{macro}
```

La version typographiée en est :

`\MaMacro` Nous définissons une macro banale, `\MaMacro`, pour illustrer l'utilisation de l'environment `macro`.

```
1 \newcommand{\MaMacro}{Ceci est
2   une macro \LaTeX.}
```

Pour plus de visibilité, Doc compose le nom de la macro ou de l'environnement dans la marge. Il ajoute également les entrées idoines dans l'index — voir la table 2 page 37 pour des exemples de présentation de ces entrées. Notez que `\begin{macro}`… `\end{macro}` n'est pas indispensable pour indiquer la définition de la macro. Il peut par contre être utilisé aussi pour indiquer des définitions de « variables LaTeX » comme des compteurs, des dimensions ou des boites :

```
% \begin{macro}{myCounter}
% Ceci est un exemple d'utilisation de l'environment |macro|
% pour présenter quelque chose d'autre qu'une macro.
%    \begin{macrocode}
\newcounter{myCounter}
%    \end{macrocode}
% \end{macro}
```

On peut imbriquer les environnements `macro` et `environment`. Cette facilité est utile non seulement pour les macros qui définissent d'autres macros mais aussi pour définir un groupe de variables reliées entre elles et partageant une description commune :

```
% \begin{macro}{\hauteurdetruc}
% \begin{macro}{\largeurdetruc}
% \begin{macro}{\profondeurdetruc}
% Ces longueurs conservent les dimensions de notre boite |\truc|.
% (En fait nous montrons juste comment imbriquer des
% environnements |macro|.)
%    \begin{macrocode}
\newlength{\hauteurdetruc}
\newlength{\largeurdetruc}
\newlength{\profondeurdetruc}
%    \end{macrocode}
% \end{macro}
% \end{macro}
% \end{macro}
```

On devrait habituellement éviter les environnements `macro` sans description car la présentation en est légèrement laide ; le nom de la macro apparait seul sur une ligne, à gauche d'une description « vide » mais le code ne commence qu'à la ligne d'après.

On peut placer de nombreux environnements `macrocode` à l'intérieur

d'un bloc `\begin{macro}`... `\end{macro}` ou `\begin{environment}`... `\end{environment}`. Voici le mécanisme par lequel le code peut être commenté à l'intérieur d'une macro ou d'un environnement. On voit d'un mauvais œil l'utilisation de `%` dans un bloc `macrocode` pour faire un commentaire. Voici un exemple de la façon dont on peut commenter une macro non triviale :

```
% \begin{macro}{\complexMacro}
% Faisons comme s'il s'agissait d'une macro très complexe
% dont il faut commenter chaque partie.
%    \begin{macrocode}
\newcommand{\complexMacro}{%
%    \end{macrocode}
% Mettons à zéro tous nos compteurs.
%    \begin{macrocode}
  \setcounter{count@i}{0}%
  \setcounter{count@ii}{0}%
  \setcounter{count@iii}{0}%
  \setcounter{count@iv}{0}%
%    \end{macrocode}
```

```
% Ici nous faisons une action vraiment compliquée.
%    \begin{macrocode}

                ⋮

%    \end{macrocode}
% Et nous en avons enfin fini ici.
%    \begin{macrocode}
}
%    \end{macrocode}
% \end{macro}
```

L'annexe A.3 présente un squelette complet de fichier `.dtx` regroupant un fichier `.sty` et sa documentation.

Fichiers de classe Le processus de production d'un fichier de classe à partir d'un fichier `.dtx` est loin d'être aussi direct que celui d'un fichier de style. Le problème est que `\DocInput` compte sur

la ligne de `\usepackage{`⟨*extension*⟩`}` — plus précisément la ligne de `\ProvidesPackage` présente dans ⟨*extension*⟩ `.sty` — pour définir les macros `\fileversion` et `\filedate`. De plus un fichier de classe ne peut être chargé avec `\usepackage`. On ne peut pas non plus le charger avec `\documentclass{`⟨*extension*⟩`}` parce que un document ne peut charger qu'une seule classe et que l'on a besoin de la classe `ltxdoc`.

La solution passe par l'utilisation de `\ProvidesFile` pour rendre disponible la date et la version du fichier à l'intérieur du fichier `.dtx`. L'annexe A.4 présente un squelette complet de fichier `.dtx` regroupant un fichier `.cls` et sa documentation. Il ressemble au squelette de fichier présenté à l'annexe A.3 mais possède une section d'entête avec une structure différente.

4 Tours, trucs et recommandations

– Écrivez beaucoup de bonne documentation ! C'est vraiment une aide précieuse à la compréhension par les autres de votre code et de votre

extension comme un tout.

– Si vous pensez que la communauté LaTeX puisse être intéressée par votre extension alors vous devriez la déposer sur le CTAN à l'adresse http://www.ctan.org/upload. En tant que dépôt de référence de tout ce qui est lié à TeX, le CTAN permet aux autres de trouver votre extension bien plus facilement que si elle n'est située que sur votre page personnelle.
– Lorsque vous diffusez votre extension, n'oubliez pas d'inclure un fichier `README`[15] décrivant ce qu'accomplit votre extension ainsi que la documentation *précompilée* de préférence sous forme de fichier PDF. La documentation précompilée épargne à l'utilisateur d'avoir à télécharger votre extension avant même de savoir ce qu'elle est supposée faire ou si elle est à même de répondre à ses besoins.
– Utilisez les commandes de sectionnement de LaTeX pour organiser

15. Et un `LISEZMOI` si possible. Je profite de cette note pour signaler que le `README` peut être la seule chose rédigée en anglais dans votre distribution et un anglais très basique suffit. Si votre extension est utile, il y aura bien quelqu'un pour en fournir une documentation succinte en "*langue internationale*". Que cela ne vous dispense pas de documenter votre code dans la langue de votre choix ! [Le TdS]

le code et clarifier sa structure — p. ex. `\subsection{Macros d'initialisation}`, `\subsection{Fonctions d'aide}`, `\subsection{Macros et environnements exportés}`, ...).

– Bien que les commentaires ne devraient vraiment apparaitre que dans la documentation typographiée, il est aussi possible d'écrire des commentaires qui ne sont visibles que dans le fichier `.sty`, dans la documentation typographiée et le fichier `.sty` ou encore uniquement dans le source `.dtx`. La table 3 montre comment contrôler la visibilité des commentaires.
– Toute ligne se trouvant entre `<*package>` et `</package>`, à l'exception de celles placées dans un environnement `macrocode`, doit commencer par un `%`. N'utilisez pas de lignes vides, elles seraient reproduites dans le fichier `.sty`, ce qui ne devrait pas être.
– Il est de bonne pratique, dans les programmes LaTeX, d'utiliser le caractère `@` dans les noms de macros, compteurs, dimensions, etc. déclarés globalement mais prévus pour n'être utilisés que de manière interne à l'extension. Cela évite à l'utilisateur de corrompre l'état de l'extension en redéfinissant par inadvertance un élément interne de

TABLE 3 – Visibilité des commentaires

Apparait dans la doc	Apparait dans le `.sty`	Mécanisme
Non	Non	`% ^^A` ⟨*commentaire*⟩

Non	Oui	`% \iffalse`
		`%%` ⟨*commentaire*⟩
		`% \fi`

Oui	Non	`%` ⟨*commentaire*⟩
Oui	Oui	`%%` ⟨*commentaire*⟩

cette dernière.[16] Il est également conseillé de préfixer tous les noms globaux internes à l'extension par le nom de l'extension — p. ex. « \⟨*extension*⟩`@thing` » au lieu de « `\@thing` » ou, pire encore, simplement « `\thing` ». Cela permet d'éviter les conflits de noms entre extensions. Enfin, comme on ne peut pas normalement utiliser les chiffres arabes dans les noms de macros, on utilise communément les nombres romains à leur place, p. ex. `\arg@i`, `\arg@ii`, `\arg@iii`, `\arg@iv`, etc.

– On peut utiliser `\index` comme d'habitude pour indexer autre chose qu'une macro ou un environnement.
– Si on utilise Emacs comme éditeur de texte, on peut essayer le `doctex-mode` de `swiftex.el`, un mode Emacs créé spécialement pour l'écriture de fichier `.dtx`. `swiftex.el` est disponible sur CTAN.
  À défaut et de façon plus simpliste, on peut utiliser les commandes `string-rectangle` et `kill-rectangle` d'Emacs. Elles sont bien utiles

16. Dans un document LaTeX, `@` est de catégorie (*catcode*) 12 « autre » et non de catégorie 11 « lettre » aussi l'utilisateur ne peut facilement ni définir ni utiliser une macro dont le nom comporte un `@`.

pour ajouter et enlever un `%` au début de chaque ligne d'une région.[17]

– On lira, bien sûr, « (*The DocStrip Program*) » et « (*The Doc and `shortvrb` Packages*) », les documentations de DocStrip et de Doc respectivement — fournie au format `.dtx` bien entendu[18]. Elles expliquent comment réaliser des choses avec les fichiers `.ins` et `.dtx` plus avancées que ce que couvre ce tutoriel. On trouvera, dans les sujets avancés :
  – extraction de plusieurs fichiers `.sty` d'un unique fichier `.dtx` ;
  – placement de préambules différents dans différents fichiers `.sty` ;
  – extraction d'autres choses que des fichiers `.sty`, p. ex. des fichiers de configuration ou des scripts Perl, d'un fichier `.dtx` ;
  – changement de la présentation de la documentation typographiée.

17. Je ne répèterai pas ici ce que j'ai déjà dit d'AucTeX ! [Le TdS]

18. Et dont une traduction en français, de votre serviteur, devrait être bientôt (?) disponible. [Le TdS]

5 Caractéristiques obscures de l'emballage

Cette section contient différents trucs de sorcellerie que l'on peut réaliser avec Doc et DocStrip. Peu d'extension requerrons ces techniques mais on les place ici pour que l'on puisse s'y référer commodément.

5.1 Fichier de documentation maitre

Doc gère les fichiers de documentation maitres qui contrôlent plusieurs fichiers `.dtx`. L'avantage en est qu'un ensemble de fichiers `.dtx` liés entre eux peut être typographié avec une numérotation continue des sections et un unique index unifié. En fait, le code source de LaTeX 2_ε lui-même est typographié avec un tel document maitre — `source2e.tex` — qui contient la myriade[19] de fichiers `.dtx` qui comprend LaTeX 2_ε.

Pour aider à la création de documents maitres, la classe `ltxdoc` fournit une commande appelée « `\DocInclude` ». Cette commande de `ltxdoc`

19. Exemple d'exagération homérique ! [Le TdS]

ressemble beaucoup à `\DocInput` de Doc — elle l'utilise même en interne — mais possède les caractéristiques complémentaires suivantes :

– `\PrintIndex` est manipulé correctement de façon automatique ;
– chaque fichier incorporé par `\DocInclude` reçoit une page de titre ;
– `\tableofcontents` fonctionne comme on l'espère. Les noms des fichiers `.dtx` sont utilisés comme titre de « chapitre ».

Notez que `\DocInclude`, au contraire de `\DocInput`, présume que le nom de fichier comporte l'extension `.dtx`.

L'annexe A.5 présente le squelette d'un document maitre qui utilise `\DocInclude` pour composer un document unique à partir de ⟨*file1*⟩`.dtx`, ⟨*file2*⟩`.dtx` et ⟨*file3*⟩`.dtx`. Si l'on préfère une approche plus artisanale — si p. ex. on n'aime pas la page de titre que `\DocInclude` crée pour chaque fichier —, on peut toujours utiliser `\DocInput`. Il faudra toutefois redéfinir `\PrintIndex` pour qu'elle ne fasse rien, sinon, chaque fichier aurait son propre index. Lorsque tous les fichiers `.dtx` ont été typographié, on appelera la commande originelle `\PrintIndex` pour composer un index unifié :

```
\begin{document}
  \let\origPrintIndex=\PrintIndex \let\PrintIndex=\relax
  \DocInput{<file1>.dtx}
  \DocInput{<file2>.dtx}
  \DocInput{<file3>.dtx}
  \origPrintIndex
\end{document}
```

5.2 Distribution à fichier unique

Bien que les extensions LaTeX soient distribuées ordinairement sous la forme conjointe d'un fichier `.ins` et d'un fichier `.dtx`, on peut distribuer une extension dans un unique fichier. Le truc est d'inclure tout le fichier `.ins` au début du fichier `.dtx`, juste après les lignes préfixées par `%`⟨*package*⟩ :

```
%<*batchfile>
\begingroup
```

```
        ⋮
⟨Contenu entier du fichier .ins⟩
        ⋮
\endgroup
%</batchfile>
```

On omettra le `\endbatchfile` pour permettre à LaTeX de continuer avec la suite du fichier `.dtx`. On pourra également, pour éviter le message « `File` ⟨*fichier-sty*⟩ `already exists on the system. Overwrite it? [y/n]` [20] », placer un « `\askforoverwritefalse` » avant la première commande `\generate`. Cela permettra de remplacer automatiquement le fichier `.sty` déjà présent. Enrober les commandes `\generate` dans un « `\IfFileExists{`⟨*fichier-sty*⟩`}{}{`... `}` » supprimera le remplacement. Il faudrait également placer les instructions d'installation du `.sty` à la fin du fichier `.dtx` afin qu'elles ne déroulent pas en dehors de l'écran de l'utilisateur. On aura besoin de `\typeout` car `\Msg` ne sera pas définie :

20. Ce qui signifie : « Le fichier `fichier-sty` existe déjà sur le système. Faut-il le remplacer ? [o/n] ». [Le TdS]

```
% \Finale
%
% \typeout{*************************************************}
% \typeout{*}
% \typeout{* To finish the installation you have to move the}
% \typeout{* following file into a directory searched by TeX:}
% \typeout{*}
% \typeout{* \space\space skeleton.sty}
% \typeout{*}
% \typeout{* Documentation is in skeleton.dvi.}
% \typeout{*}
% \typeout{* Happy TeXing!}
% \typeout{*************************************************}
\endinput
```

5.3 Fichiers de style et de classe partageant des informations de version

Certaines extensions contiennent à la fois un fichier `.cls` et un `.dtx`. On peut vouloir les extraire tous les deux d'un même fichier `.ins` et

partager les mêmes chaines d'information de version. La documentation de DocStrip explique comment extraire plusieurs fichiers avec un seul appel à `\generate`

```
\generate{\file{<extension>.cls}{\from{<extension>.dtx}{class}}
          \file{<extension>.sty}{\from{<extension>.dtx}{package}}}
```

Pour utiliser une seule chaine d'information de version dans les fichiers `.cls` et `.sty` on changera les lignes suivantes du fichier `.dtx` présenté à l'annexe A.4 :

```
%<<class>>\NeedsTeXFormat{LaTeX2e}[1999/12/01]
%<<class>>\ProvidesClass{<extension>}
%<<*class>>
    [<AAAA>/<MM>/<JJ> v<version> <description brève>]
%<</class>>
```

Le nouveau code précise quelle ligne appartient au fichier de classe et laquelle appartient au fichier de style :

```
%<<class|package>>\NeedsTeXFormat{LaTeX2e}[1999/12/01]
%<<class>>\ProvidesClass{<extension>}
%<<extension>>\ProvidesPackage{<extension>}
%<<*class|package>>
    [<AAAA>/<MM>/<JJ> v<version> <description brève>]
%<</class|package>>
```

A Squelettes de fichiers

Cette section contient des squelettes complet de types de fichiers présentés ci-avant. On peut utiliser ces squelettes comme gabarit pour créer ses propres extensions.

A.1 Un squelette de fichier `.ins` pour créer un fichier `.sty`

```
%%
%% Copyright (C) <année> by <votre nom>
%%
```

```
%% This file may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either
%% version 1.2 of this license or (at your option) any later
%% version.  The latest version of this license is in:
%%
%%    http://www.latex-project.org/lppl.txt
%%
%% and version 1.2 or later is part of all distributions of
%% LaTeX version 1999/12/01 or later.
%%

\input docstrip.tex
\keepsilent

\usedir{tex/latex/<extension>}

\preamble

This is a generated file.

Copyright (C) <année> by <votre nom>
```

```
This file may be distributed and/or modified under the
conditions of the LaTeX Project Public License, either
version 1.2 of this license or (at your option) any later
version.  The latest version of this license is in:

   http://www.latex-project.org/lppl.txt

and version 1.2 or later is part of all distributions of
LaTeX version 1999/12/01 or later.

\endpreamble

\generate{\file{<extension>.sty}{\from{<extension>.dtx}{package}}}

\Msg{************************************************************}
\Msg{*}
\Msg{* To finish the installation you have to move the}
\Msg{* following file into a directory searched by TeX:}
\Msg{*}
\Msg{* \space\space <extension>.sty}
```

```
\Msg{*}
\Msg{* To produce the documentation run the file <extension>.dtx}
\Msg{* through LaTeX.}
\Msg{*}
\Msg{* Happy TeXing!}
\Msg{*************************************************************}

\endbatchfile
```

A.2 Un squelette de fichier `.ins` pour créer un fichier `.cls`

```
%%
%% Copyright (C) <année> by <votre nom>
%%
%% This file may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either
%% version 1.2 of this license or (at your option) any later
%% version.  The latest version of this license is in:
%%
%%    http://www.latex-project.org/lppl.txt
```

```
%%
%% and version 1.2 or later is part of all distributions of
%% LaTeX version 1999/12/01 or later.
%%

\input docstrip.tex
\keepsilent

\usedir{tex/latex/<extension>}

\preamble

This is a generated file.

Copyright (C) <année> by <votre nom>

This file may be distributed and/or modified under the
conditions of the LaTeX Project Public License, either
version 1.2 of this license or (at your option) any later
version.  The latest version of this license is in:
```

```
   http://www.latex-project.org/lppl.txt

and version 1.2 or later is part of all distributions of
LaTeX version 1999/12/01 or later.

\endpreamble

\generate{\file{<extension>.cls}{\from{<extension>.dtx}{class}}}

\Msg{************************************************************}
\Msg{*}
\Msg{* To finish the installation you have to move the}
\Msg{* following file into a directory searched by TeX:}
\Msg{*}
\Msg{* \space\space <extension>.cls}
\Msg{*}
\Msg{* To produce the documentation run the file <class>.dtx}
\Msg{* through LaTeX.}
\Msg{*}
\Msg{* Happy TeXing!}
\Msg{************************************************************}
```

```
\endbatchfile
```

A.3 Un squelette de fichier `.dtx` pour créer un fichier `.sty`

```
% \iffalse meta-comment
%
% Copyright (C) <année> by <votre nom>
% ---------------------------------
%
% This file may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.2
% of this license or (at your option) any later version.
% The latest version of this license is in:
%
%    http://www.latex-project.org/lppl.txt
%
% and version 1.2 or later is part of all distributions of LaTeX
% version 1999/12/01 or later.
```

```
%
% \fi
%
% \iffalse
%<<package>>\NeedsTeXFormat{LaTeX2e}[1999/12/01]
%<<package>>\ProvidesPackage{<extension>}
%<<package>>   [<AAAA>/<MM>/<JJ> v<version> <brief description>]
%
%<<*driver>>
\documentclass{ltxdoc}
\usepackage{<extension>}
\EnableCrossrefs
\CodelineIndex
\RecordChanges
\begin{document}
  \DocInput{<extension>.dtx}
\end{document}
%<</driver>>
% \fi
%
% \CheckSum{0}
```

```
%
% \CharacterTable
%  {Upper-case    \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
%   Lower-case    \a\b\c\d\e\f\g\h\i\j\k\l\m\n\o\p\q\r\s\t\u\v\w\x\y\z
%   Digits        \0\1\2\3\4\5\6\7\8\9
%   Exclamation   \!     Double quote  \"     Hash (number) \#
%   Dollar        \$     Percent       \%     Ampersand     \&
%   Acute accent  \'     Left paren    \(     Right paren   \)
%   Asterisk      \*     Plus          \+     Comma         \,
%   Minus         \-     Point         \.     Solidus       \/
%   Colon         \:     Semicolon     \;     Less than     \<<
%   Equals        \=     Greater than  \>>     Question mark \?
%   Commercial at \@     Left bracket  \[     Backslash     \\
%   Right bracket \]     Circumflex    \^     Underscore    \_
%   Grave accent  \`     Left brace    \{     Vertical bar  \|
%   Right brace   \}     Tilde         \~}
%
%
% \changes{v1.0}{<AAAA>/<MM>/<JJ>}{Première version}
%
% \GetFileInfo{<extension>.sty}
```

```
%
% \DoNotIndex{<list of control sequences>}
%
% \title{L'extension \textsf{<extension>}\thanks{Ce document
%   correspond à la version~\fileversion de \textsf{<extension>},
%   datée du~\filedate.}}
% \author{<votre nom> \\ \texttt{<votre addresse électronique>}}
%
% \maketitle
%
% \begin{abstract}
%   Placer ici le texte du résumé.
% \end{abstract}
%
% \section{Introduction}
%
% Du texte ici.
%
% \section{Usage}
%
% \DescribeMacro{\VotreMacro}
```

```
% Description de |\VotreMacro| ici.
%
% \DescribeEnv{VotreEnvir}
% Description de |VotreEnvir| ici.
%
% \StopEventually{\PrintIndex}
%
% \section{Implementation}
%
% \begin{macro}{\VotreMacro}
%  Explication de l'implémentation de |\VotreMacro| ici.
%    \begin{macrocode}
\newcommand{\VotreMacro}{}
%    \end{macrocode}
% \end{macro}
%
% \begin{environment}{VotreEnvir}
% Explication de l'implémentation de |VotreEnvir| ici.
%    \begin{macrocode}
\newenvironment{VotreEnvir}{}{}
%    \end{macrocode}
```

```
% \end{environment}
%
% \Finale
\endinput
```

A.4 Un squelette de fichier `.dtx` pour créer un fichier `.cls`

```
% \iffalse meta-comment
%
% Copyright (C) <année> by <votre nom>
% ------------------------------------
%
% This file may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.2
% of this license or (at your option) any later version.
% The latest version of this license is in:
%
%    http://www.latex-project.org/lppl.txt
%
% and version 1.2 or later is part of all distributions of LaTeX
```

```
% version 1999/12/01 or later.
%
% \fi
%
% \iffalse
%<<*driver>>
\ProvidesFile{<extension>.dtx}
%</driver>>
%<<class>>\NeedsTeXFormat{LaTeX2e}[1999/12/01]
%<<class>>\ProvidesClass{<extension>}
%<<*class>>
    [<AAAA>/<MM>/<JJ> v<version> <description brève>]
%</class>>
%
%<<*driver>>
\documentclass{ltxdoc}
\EnableCrossrefs
\CodelineIndex
\RecordChanges
\begin{document}
  \DocInput{<extension>.dtx}
```

```
\end{document}
%<</driver>>
% \fi
%
% \CheckSum{0}
%
% \CharacterTable
%  {Upper-case    \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
%   Lower-case    \a\b\c\d\e\f\g\h\i\j\k\l\m\n\o\p\q\r\s\t\u\v\w\x\y\z
%   Digits        \0\1\2\3\4\5\6\7\8\9
%   Exclamation   \!     Double quote  \"     Hash (number) \#
%   Dollar        \$     Percent       \%     Ampersand     \&
%   Acute accent  \'     Left paren    \(     Right paren   \)
%   Asterisk      \*     Plus          \+     Comma         \,
%   Minus         \-     Point         \.     Solidus       \/
%   Colon         \:     Semicolon     \;     Less than     \<<
%   Equals        \=     Greater than  \>>     Question mark \?
%   Commercial at \@     Left bracket  \[     Backslash     \\
%   Right bracket \]     Circumflex    \^     Underscore    \_
%   Grave accent  \`     Left brace    \{     Vertical bar  \|
%   Right brace   \}     Tilde         \~}
```

```
%
%
% \changes{v1.0}{<AAAA>/<MM>/<JJ>}{Première version}
%
% \GetFileInfo{<extension>.dtx}
%
% \DoNotIndex{<list of control sequences>}
%
% \title{L'extension \textsf{<extension>}\thanks{Ce document
%   correspond à la version~\fileversion de \textsf{<extension>},
%   datée du~\filedate.}}
% \author{<votre nom> \\ \texttt{<votre addresse électronique>}}
%
% \maketitle
%
% \begin{abstract}
%   Placer ici le texte du résumé.
% \end{abstract}
%
% \section{Introduction}
%
```

```
% Du texte ici.
%
% \section{Usage}
%
% \DescribeMacro{\VotreMacro}
% Description de |\VotreMacro| ici.
%
% \DescribeEnv{VotreEnvir}
% Description de |VotreEnvir| ici.
%
% \StopEventually{\PrintIndex}
%
% \section{Implementation}
%
% \begin{macro}{\VotreMacro}
%  Explication de l'implémentation de |\VotreMacro| ici.
%    \begin{macrocode}
\newcommand{\VotreMacro}{}
%    \end{macrocode}
% \end{macro}
%
```

```
% \begin{environment}{VotreEnvir}
% Explication de l'implémentation de |VotreEnvir| ici.
%    \begin{macrocode}
\newenvironment{VotreEnvir}{}{}
%    \end{macrocode}
% \end{environment}
%
% \Finale
\endinput
```

A.5 Un squelette de fichier document maitre (`.tex`)

```
\documentclass{ltxdoc}
\usepackage{<file1>}
\usepackage{<file2>}
\usepackage{<file3>}

\title{<titre>}
\author{<vous>}
```

```
\EnableCrossrefs
\CodelineIndex
\RecordChanges

\begin{document}
  \maketitle

  \begin{abstract}
    <résumé>
  \end{abstract}

  \tableofcontents

  \DocInclude{<file1>}
  \DocInclude{<file2>}
  \DocInclude{<file3>}
\end{document}
```

Références

[1] Frank Mittelbach and Michel Goossens with Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion Second Edition.* Addison Wesley, Pearson Education, Inc. 2004. ISBN 0-201-36299-6.

[2] Frank Mittelbach et Michel Goossens *Le LaTeX Companion 2e édition.* Pearson Education, 2006. ISBN 2-744-07182-X. *traduction française de [1]*

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison Wesley, Reading, Massachusetts, October 1, 1994. ISBN 0-201-54199-8.

[4] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, May 1984. British Computer Society. Available from http://www.literateprogramming.com/knuthweb.pdf.

Index