Production de fiches de solutions avec LaTeX 2_ε*

31 octobre 2001

Table des matières

1 Introduction

Ce *package* est une modification de l'ancienne option de style `answers` de l'auteur, qui a été utilisée pendant quelques années, basée sur l'idée du TEXBook de lier solutions et exercices. J'ai saisi l'opportunité de cette révision pour modifier le format des solutions, afin qu'elles soient maintenant présentées comme un environnement LaTeX plutôt que débutées par une commande et terminées par la fin de l'environnement environnant, une manière fort peu en accord avec LaTeX !

*Traduit en français par Jean-Pierre Drucbert et Mathieu Goutelle le 15 septembre 2000. Titre original « Production of solution sheets in LaTeX 2_ε ».

L'autre changement important est que plusieurs descripteurs de fichiers peuvent être actifs en même temps. Cela permet de placer certaines solutions en annexe dans un livre (par exemple) et d'autres d'être placées dans un fichier séparé pour être imprimées et distribuées aux étudiants au fur et à mesure de la progression du cours.

De plus, les fichiers physiques actuels ouverts avec chaque descripteur peuvent maintenant être changés en cours de travail, permettant ainsi de créer plusieurs fichiers avec le même format. Ainsi, par exemple, chaque chapitre d'un livre peut créer son propre fichier de solutions, permettant à l'utilisateur d'utiliser `\include` pour les chapitres et les solutions.

Enfin, n'importe quel nombre de types de solutions peut être attaché à un fichier et non plus uniquement les deux anciens, solution et indication. Le format de chaque type de solution est sous le contrôle total de l'utilisateur.

2 Le fichier du gestionnaire de documentation

Voici le fichier du gestionnaire qui produit la documentation. On utilise la classe de document fournie par la distribution LaTeX 2_ε pour produire la documentation.

```
⟨*driver⟩
\documentclass{ltxdoc}
\usepackage[T1]{fontenc}       % ajouté par le traducteur
\usepackage[latin1]{inputenc}  % ajouté par le traducteur
\usepackage[frenchb]{babel}    % ajouté par le traducteur
\RecordChanges
\begin{document}
  \DocInput{f-answers.dtx}
  \PrintIndex
  \PrintChanges
\end{document}
⟨/driver⟩
```

3 Interface utilisateur

Le *package* doit être inclus avec la commande :

```
\usepackage[nosolutionfiles]{answers}
```

Si l'argument optionnel est spécifié, les solutions apparaissent directement dans le texte au lieu d'être écrites dans un autre fichier. Cela permet de produire une version de démonstration.

`\Newassociation` Après cela, il doit y avoir plusieurs déclarations de la forme :

```
\Newassociation{xxx}{yyy}{zzz}
```

où `xxx` est un environnement dans le document et `yyy` un environnement qui entourera le contenu de `xxx` lorsqu'il est écrit dans le descripteur de fichier symbolique `zzz`. Les noms `xxx`, `yyy` and `zzz` doivent contenir uniquement des lettres, pas de chiffre, de signe de ponctuation ou d'espaces.

`\solutionextension` Par défaut, la sortie se fait dans `zzz.tex` si `zzz` est ouvert. La commande `\solutionextension` peut être redéfini pour remplacer l'extension `tex` par une autre. Alternativement, le fichier de sortie peut être modifié en tant que paramètre de `\Opensolutionfile` et chaque `\Opensolutionfile` sur le même descripteur peut utiliser un fichier physique différent. Par défaut, `\solutionpoint` est ajouté avant `\solutionextension`. Pour le supprimer, il faut le redéfinir (la valeur par défaut est évidemment un point).

`\Opensolutionfile` `\Closesolutionfile` À un certain point, l'utilisateur tape :

```
\Opensolutionfile{zzz}
...
\Closesolutionfile{zzz}
```

pour créer un fichier de solution écrit par les environnements `xxx` jusqu'aux environnements `yyy`. Si cette construction est utilisée plusieurs fois, plusieurs fichiers de solutions vont être créés. L'utilisateur peut souhaitér que ces fichiers aient des noms différents. S'il utilise `\Opensolutionfile{zzz}[www]`, alors `www.tex` est utilisé comme nom de fichier de sortie actuel au lieu de `zzz.tex`. Cela permet au descripteur de fichier `zzz` de créer plusieurs fichiers `www.tex`, par exemple un pour chaque chapitre d'un livre ou pour chaque feuille de problème. Ensuite, on peut les utiliser grâce à la commande `\include`. La même valeur de `\solutionextension` est utilisé pour les arguments principal et optionnel. Le nom `www` doit respecter les conventions habituelles sur les noms de fichier.

`\Writetofile` De plus, des données peuvent être écrites directement dans un fichier au moyen de `\Writetofile`. Son premier argument est le descripteur de fichier et le second la ligne de texte à écrire. Il est très important de se souvenir que chaque mot de contrôle dans la ligne doit être précédé par `\protect`, autrement la primitive TEX `\write` va les développer. Aussi, comme l'argument est lu à la manière habituelle de TEX, tout espace suivant un mot de contrôle disparaîtra à moins que des précautions soient prises. Donc, pour écrire `\xx yyy` dans le fichier, l'utilisateur doit taper `\protect\xx\space yyy`.

`Filesave` Un bloc de texte peut être sauvé dans un descripteur de fichier grâce à :

```
\begin{Filesave}{zzz}
  ....
\end{Filesave}
```

autour de celui-là, dès que `zzz` a été ouvert. Les restrictions précédentes s'appliquant à `\Writetofile` n'existent pas pour cet environnement.

`\Readsolutionfile` L'un des fichiers générés peut être lu grâce à :

```
\Readsolutionfile{zzz}
```

à condition que le fichier n'ait pas été fermé sans être rouvert. On peut aussi utiliser simplement `\input` ou `\include` au choix.

Aucune de ces opérations E/S n'ont d'effet si le descripteur de fichier n'a pas été ouvert ou si le paramètre `nosolutionsfiles` est spécifié.

4 Un exemple simple

Voici un exemple direct pour illustrer comment on peut utiliser ces macros.

```
⟨*ex1⟩
\documentclass[12pt,a4paper]{article}
\usepackage{answers}
\Newassociation{sol}{Solution}{ans}
\newtheorem{ex}{Exercice}
\begin{document}
\Opensolutionfile{ans}[ans1]
\section{Problèmes}
\begin{ex}
Premier exercice
\begin{sol}
  Première solution.
\end{sol}
\end{ex}
\begin{ex}
Deuxième exercice
\begin{sol}
  Deuxième solution.
\end{sol}
\end{ex}
\Closesolutionfile{ans}
\section{Solutions}
\input{ans1}
\end{document}
⟨/ex1⟩
```

5 Un exemple complexe

Ce qui suit est un exemple (très complexe) de l'usage du *package* `answers`. Il utilise certains des raffinements décrits plus haut.

```
⟨*ex2⟩
\documentclass[12pt,a4paper]{article}
\usepackage{answers}%\usepackage[nosolutionfiles]{answers}
```

D'abord, un environnement qui contient des problèmes et qui les numérote. Il est basé sur un `theorem` de LaTeX mais avec un corps en roman plutôt qu'en italique.

```
\newtheorem{Exc}{Exercice}
\newenvironment{Ex}{\begin{Exc}\normalfont}{\end{Exc}}
```

Trois types de solutions sont écrites dans deux fichiers différents. Le descripteur de fichier `test` contiendra les solutions et indications que l'étudiant verra ; `testtwo` contient les solutions aux problèmes qu'ils remettront vraisemblablement et qui doivent donc être préparées séparément.

```
\Newassociation{solution}{Soln}{test}
\Newassociation{hint}{Hint}{test}
\Newassociation{Solution}{sSol}{testtwo}
```

Parce que nous voulons repérer deux types distincts de problèmes dans le fichier principal, nous définissons ceci :

```
\newcommand{\prehint}{~[Indication]}
\newcommand{\presolution}{~[Solution]}
\newcommand{\preSolution}{~[Devoir à la maison]}
```

Nous fournissons un paramètre supplémentaire quand nous ouvrons un descripteur de fichier `test` car nous voulons écrire la commande `\section` dans le fichier solution. Il s'agit seulement ici d'une illustration ; cela serait plus pertinent si le fichier solution était inclus par `\include`

```
\newcommand{\Opentesthook}[2]%
{\Writetofile{#1}{\protect\section{#1: #2}}}
```

Le texte par défaut, que LaTeX écrit lorsqu'il rencontre un environnement solution est modifié ici.

```
\renewcommand{\Solnlabel}[1]{\emph{Solution #1}}
\renewcommand{\Hintlabel}[1]{\emph{Indication #1}}
\renewcommand{\sSollabel}[1]{\emph{Solution à #1}}

\begin{document}
```

Nous ouvrons le descripteur `test` en tant que fichier `test1.tex`.

```
\Opensolutionfile{test}[ans2]{Solutions}
```

et nous écrivons un peu de texte dedans.

```
\Writetofile{test}{\protect\subsection{Quelques solutions}}
```

Le descripteur `testtwo` est ouvert en tant que `testtwo.tex`.

```
\Opensolutionfile{testtwo}[ans2x]
\Writetofile{testtwo}{%
\protect\subsection{Solutions supplémentaires}}
```

Maintenant les problèmes.

```
\section{Exercises}
\begin{Ex}
  Un exercice avec une solution.
  \begin{solution}
  Voilà une solution.
  \relax{}
  \end{solution}
\end{Ex}
\begin{Ex}
  Un exercice avec une indication et une solution cachée.
  \begin{hint}
  Voilà une indication.
```

```
  \end{hint}
  \begin{Solution}
  Voilà une solution cachée.
  \end{Solution}
\end{Ex}
\begin{Ex}
  Un exercice avec une indication.
  \begin{hint}
  Voilà une indication.
  \end{hint}
\end{Ex}
```

Nous fermons les deux fichiers de solution et incluons immédiatement leur contenu. Nous aurions pu utiliser ici `\include`.

```
\Closesolutionfile{test}
\Readsolutionfile{test}
\clearpage
\Closesolutionfile{testtwo}
\Readsolutionfile{testtwo}
\end{document}
⟨/ex2⟩
```

6 Un exemple supplémentaire

Voici une application à une situation non envisagée au départ, suggérée à l'auteur par Martin Osborne. Ici, les exercices et solutions ne sont pas numérotés ; ils sont *décrits*.

```
⟨*ex3⟩
\documentclass[12pt,a4paper]{article}
\usepackage{answers}
\newenvironment{Ex}[1]{\begin{trivlist}\item \emph{#1} %
\renewcommand{\Currentlabel}{#1}}{\end{trivlist}}
\Newassociation{solution}{Soln}{solutions}

\renewenvironment{Soln}[1]{\begin{trivlist}\item
Solution à \emph{#1} }{\end{trivlist}}

\begin{document}
\section*{Problèmes}
\Opensolutionfile{solutions}[ans3]
\begin{Ex}{Premier exercice}
  Un exercice avec une solution.
  \begin{solution}
  Voici la solution.
  \relax{}
  \end{solution}
\end{Ex}
\begin{Ex}{Second exercice}
  Un second exercice avec une solution.
```

```
 \begin{solution}
 Voici une autre solution.
 \end{solution}
\end{Ex}
\Closesolutionfile{solutions}
\section*{Solutions}
\Readsolutionfile{solutions}
\end{document}
⟨/ex3⟩
```

7 Identification

Ce *package* peut être utilisé uniquement avec $\LaTeX\,2_\varepsilon$, un message idoine est donc affiché quand un autre format est utilisé.

```
⟨*answers⟩
\NeedsTeXFormat{LaTeX2e}[1996/06/01]
```

On déclare le nom du *package* et sa version :

```
\ProvidesPackage{answers}[1996/07/10]
```

et on l'affiche sur le terminal (et le fichier log) :

```
\typeout{Package `answers' <\filedate>.}
\typeout{\Copyright}
```

8 Options

Il y a une seul option `nosolutionfiles` qui bascule dans le mode où les solutions ne sont pas écrites dans des fichiers mais produites immédiatement.

```
\newif\ifanswerfiles \answerfilestrue
\DeclareOption{nosolutionfiles}{\answerfilesfalse
\typeout{No answer files being produced}}%
\ProcessOptions
```

Comme ce *package* utilise fortement le *package* `verbatim`, on s'assure que celui-ci est chargé.

```
\RequirePackage{verbatim}
```

9 Manipulation de fichiers

`\solutionextension` L'extension par défaut des fichiers solution est définie ici.

```
\newcommand{\solutionpoint}{.}
\newcommand{\solutionextension}{tex}
```

Elle peut être changée par `\renewcommand`.

`Filesave` On définit un environnement **`Filesave`** avec un paramètre, le descripteur de fichier. Il est similaire à l'exemple de Schöpf dans la description de **`verbatim`**.

```
\newenvironment{Filesave}[1]{%
\@bsphack
\def\verbatim@processline{}%
\Iffileundefined{#1}{}{%
  \Ifopen{#1}{%
  \def\verbatim@processline{%
 \Ifanswerfiles{%
\immediate\write\@nameuse{#1@file}%
{\the\verbatim@line}%
 }{}%
  }%
  }{}%
}%
\let\do\@makeother\dospecials
\catcode`\^^M\active \catcode`\^^I=12\relax
\verbatim@start
}{\@esphack}
```

`\Writetofile` Il est aussi utile d'avoir une macro pour écrire dans un fichier. Dans celle-ci, on a besoin de placer `\protect` devant chaque mot de contrôle dans l'argument qui pourrait être développé prématurément et ainsi provoqué des problèmes.

```
\newcommand{\Writetofile}[2]{%
\@bsphack
\Iffileundefined{#1}{}{%
  \Ifopen{#1}{%
  {%
 \let\protect\string
 \Ifanswerfiles{%
\immediate\write\@nameuse{#1@file}{#2}%
 }{}%
  }%
  }{}%
}%
\@esphack
}
```

`\Ifopen` On a besoin de tester si un fichier est déjà ouvert ou non.

```
\newcommand{\Ifopen}[3]{%
\csname if#1open\endcsname#2\else#3\fi}%
```

`\Iffileundefined` On a aussi besoin de tester si une variable de fichier a déjà été définie pour un descripteur de fichier donné.

```
\newcommand{\Iffileundefined}[3]{%
\csname ifx\expandafter\endcsname
  \csname #1@file\endcsname\relax
  #2\else#3\fi}
```

Enfin, on a besoin de tester si on doit faire la sortie sur un fichier ou non.

```
\newcommand{\Ifanswerfiles}[2]{%
\ifanswerfiles #1\else #2\fi}
```

10 L'interface fichier

\Opensolutionfile

Avant de pouvoir écrire des solutions, nous devons ouvrir un (ou des) fichier de solutions. La commande pour faire cela prend un seul paramètre, qui est d'ordinaire un nom de fichier sans extension. Ainsi, ce sera sans doute une chaîne de 8 caractères pour la portabilité. Cette opération ne tronquera pas un fichier ouvert existant. Néanmoins, si le second paramètre optionnel est spécifié, ce dernier représente le nom de fichier actuel et le premier paramètre est alors un nom de descripteur symbolique de fichier arbitraire.

```
\def\Opensolutionfile#1{%
\@ifnextchar[{\define@filename{#1}}%
  {\define@filename{#1}[#1]}}%
\def\define@filename#1[#2]{%
\global\@namedef{#1@filename}{#2\solutionpoint\solutionextension}%
\Ifanswerfiles{%
  \typeout{Output from handle #1 going
  to #2.\solutionextension}%
}{}%
\Iffileundefined{#1}{%
  \expandafter\newwrite\csname #1@file\endcsname
  \csname newif\expandafter\endcsname
  \csname if#1open\endcsname
  \global\csname #1openfalse\endcsname
  \expandafter\ifx\csname Open#1hook\endcsname\relax
  \global\@namedef{Open#1hook}##1{}%
  \fi
  \expandafter\ifx\csname Close#1hook\endcsname\relax
  \global\@namedef{Close#1hook}##1{}%
  \fi
}{}%
\let\Tmp\relax
\Ifopen{#1}{\typeout{File #1 already open}}{%
  \Ifanswerfiles{%
  \immediate\openout\@nameuse{#1@file}=%
  \@nameuse{#1@filename}%
  }{}%
  \global\csname#1opentrue\endcsname
  \def\Tmp{\@nameuse{Open#1hook}{#1}}%
}%
\Tmp
}
```

\Closesolutionfile

Nous avons aussi une commande pour fermer un fichier préalablement ouvert.

```
\def\Closesolutionfile#1{%
\let\Tmp\relax
```

```
\Iffileundefined{#1}{}{%
  \Ifopen{#1}{%
  \Ifanswerfiles{%
 \immediate\closeout\@nameuse{#1@file}%
  }{}%
  \global\csname #1openfalse\endcsname
  \def\Tmp{\@nameuse{Close#1hook}{#1}}%
  }{}%
}%
\Tmp
}
```

On notera que ces deux commandes fournissent un point d'entrée qui leur permet de réaliser des tâches supplémentaires. Par exemple, l'opération d'ouverture peut écrire des informations supplémentaires en redéfinissant le point d'entrée idoine. L'opération de fermeture peut, si nécessaire, réaliser un **\input** immédiat du contenu du fichier de solutions. Par exemple :

```
\newcommand{\Openxxxhook}[2]{%
  \Writetofile{#1}{\protect\section{#2}}%
}%
\newcommand{\Closexxxhook}[1]{%
  \Readsolutionfile{#1}%
}
```

et ensuite :

```
\Opensolutionfile{xxx}{Réponses aux problèmes sélectionnés}
...
\Closesolutionfile{xxx}
```

Le comportement par défaut est d'ignorer le premier paramètre. Notez que si vous le modifier, vous devez garder à l'esprit que le premier paramètre est toujours le descripteur de fichier.

\Readsolutionfile L'opération de lecture du fichier de solutions peut être accomplie avec la commande suivante.

```
\def\Readsolutionfile#1{%
\Ifanswerfiles{%
  \Iffileundefined{#1}{}{%
  \Ifopen{#1}{%
 \typeout{WARNING: attempt to read open file #1}%
  }{%
 \edef\Tmp{%
\noexpand\InputIfFileExists
{\@nameuse{#1@filename}}{}%
{\noexpand\message{File
\@nameuse{#1@filename}%
  \space not found}}%
 }%
 \Tmp
```

```
  }%
  }%
}{}%
}

```

11 L'interface solution

\Newassociation

Plusieurs descripteurs de fichier de solutions peuvent être définis. Vous êtes seulement limités par le nombre que TEX met à votre disposition. Chaque environnement de solution, pour écrire dans l'un de ces descripteurs, doit connaître dans quel descripteur il doit écrire ainsi que les informations à ajouter au fichier, indépendamment de son contenu. On réalise ceci en créant une association entre l'environnement source, l'environnement destination et le descripteur de fichier.

```
\newcommand{\Newassociation}[3]{%
\newsolution{#2}%
\expandafter\ifx\csname #3opentrue\endcsname\relax
  \expandafter\newif\csname if#3open\endcsname
\fi
\newenvironment{#1}{%
  \Ifanswerfiles{%
  \let\Tmp\relax
  \Iffileundefined{#3}{}{%
 \Ifopen{#3}{%
\immediate\write\@nameuse{#3@file}%
{\string\begin{#2}\@nameuse{#2params}}%
 \def\Tmp{\Filesave{#3}}%
 }{}%
  }%
  }{%
  \edef\Tmp{\noexpand\begin{#2}\@nameuse{#2params}}%
  }%
  \csname pre#1\endcsname
  \Tmp
}%
{%
  \Ifanswerfiles{%
  \Iffileundefined{#3}{}{%
 \Ifopen{#3}{%
\endFilesave%
\immediate\write\@nameuse{#3@file}%
{\string\end{#2}}%
\csname post#1\endcsname
 }{}%
  }%
  }{%
  \end{#2}%
  }%
```

```
}%
}
```

`\newsolution` L'environnement destination par défaut, dans le fichier de solution, est défini pour prendre un unique paramètre, un nombre de référence reçu de l'environnement source. Il est inséré avec le style `\solutionstyle`, valant par défaut `\textbf`. En plus, on peut ajouter au type de solution `yyy` des balises avant et après en définissant respectivement `\preyyy` et `\postyyy`, par exemple pour ajouter une ligne de la largeur de la page ou un symbole. Si quelque chose de plus sophistiqué est désiré, il est probablement mieux de redéfinir l'environnement par `\renewenvironment{yyy}`.

`\Currentlabel`

```
\newcommand{\newsolution}[1]{%
\@ifundefined{#1}{%
  \global\@namedef{#1params}{{\Currentlabel}}%
  \newenvironment{#1}[1]%
  {%
  \csname pre#1\endcsname
  \trivlist
  \item[\hskip\itemsep{\@nameuse{#1label}{##1}}]}%
  {\csname post#1\endcsname\endtrivlist}%
  \global\@namedef{#1label}##1{\solutionstyle{##1}}%
}{\typeout{WARNING: environment #1 already in use}}%
}
\newcommand{\solutionstyle}[1]{\textbf{#1}}
\newcommand{\Currentlabel}{\@currentlabel}
```

Le format de l'étiquette pour l'environnement solution `xxx` est défini par la commande `\xxxlabel`, qui, par défaut, prend un argument. L'argument lui est passé par la commande `\xxxparams`, qui se développe en `{\Currentlabel}`, synonyme de `{\@currentlabel}` ; cette argument est écrit automatiquement par l'environnement source. L'étiquette apparaît en gras par défaut. Nous pouvons facilement changer le comportement de cet environnement en modifiant ces deux commandes. Par exemple :

```
\renewcommand{\xxxlabel}[1]{\emph{Solution to #1}}
\renewcommand{\xxxparams}{{\Currentlabel(p.\thepage)}}
```

fourniront un nombre et une référence de page en italique.

Des comportements plus compliqués peuvent être obtenus en redéfinissant l'environnement `xxx` lui-même, pour qu'il accepte différents paramètres. Notez cependant que `\xxxparams` doit être redéfinit pour fournir ces paramètres.

Voilà qui nous amène à la fin du *package* `answers`.

```
⟨/answers⟩
```

Historique des changements