Hot plugging in a macro-definition
Remplacement à chaud dans la définition d'une macro

Benjamin BAYART*

2001/01/27

Contents
Table des matières

1 Foreword
Introduction

I'm French. I speak a few words of english. As most of the TEX users are not french, I wrote (or tried to) the documentation in english. But this package is a bit tricky to use, with a lots of side effects, so I prefere to *also* write the documentation in french. In case the french documentation states something, and the english one states the opposite, the french one is probably right, and you should send me an e-mail to help me improve my english.

*bayartb@edgard.fdn.fr

2 Hot plugging
Remplacement à chaud

The basic idea of this package is rather simple: you want to slightly modify a macro-command you didn't write (one from the kernel, or from a third party package). The usual way is that you copy the definition and change what you want to change. In some cases, there is no better solution. But in simple cases, there is a way to have a better solution. This package is to deal with the simple case.

This simple case is when the original macro (say `\bibliography`, the macro command which stand behin `\begin{bibliography}`) call one macro (say `\chapter`) and you want it to call another one (say `\section`). For this simple example, if you copy the definition, then your document might not work with another version of the class. With the `hotplug` package, you can *hotplug* the `\section` into `\bibliography` where was previously `\chapter`.

To solve this problem, the solution can be:

```
\HotReplace\bibliography\chapter\section
```

L'idée à la base de ce package est assez simple : tu souhaites modifier une macro dont tu n'es pas l'auteur (une macro du noyau LaTeX, ou prise dans un autre package). La manière habituelle est de copier la définition de cette macro dans ton document, puis d'altérer à loisir la copie. Dans les cas complexes, on ne peut pas faire autrement. Mais dans les cas simples, une méthode plus élégante est possible. Ce package présente la méthode élégante pour le cas simple.

Le cas simple c'est quand la macro originale (mettons `\bibliography`, la macro qui se cache derrière `\begin{bibliography}`) appelle une macro (disons `\chapter`) alors que tu veux qu'elle en appelle une autre (disons `\section`). Pour cet exemple simple, si tu copies la définition de `\bibliography`, ton document risque de ne plus compiler avec une autre version de la classe. Avec le package `hotplug`, tu peux *remplacer à chaud* le `\chapter` qui est dans `\bibliography` par `\section`.

Pour résoudre ce problème, la solution pourra être :

3 File limit
Nombre maximum de fichier

During `\HotReplace` we need to have a write file descriptor available. If you give no options to the package, or the `file` option, which is the default, then a `\newwrite` will be issued. It you're getting short of such things (one can only have 16 `\newwrite` within TeX, and LaTeX already uses several), you can use the option `nofile`, then the macro `\HotFile` will be defined to use the file 15, which is choosen arbitrarily. You can then choose any other file which suits you more. It will be use *only* during `\HotReplace`. It will be open, write and close during each single call to `\HotReplace`, so you can choose the file descriptor of *any* file that is not used at the

Durant le `\HotReplace`, on a besoin d'un descripteur de fichier en écriture. Si tu ne passes pas d'option au package, ou l'option `file` qui est celle pas défaut, le package fera appel à `\newwrite` pour se réserver *ad vitam etœrnam* un tel descripteur. Si tu n'en as plus assez (tu ne peux en avoir que 16 dans TeX, et, LaTeX en consomme déjà quelques uns), tu peux utiliser l'option `nofile`, en pareil cas la macro `\HotFile` sera définie pour utiliser le descripteur numéro 15, arbitrairement choisi. Tu peux alors choisir n'importe quel autre descripteur qui te convienne. Il ne sera utilisé *que* pendant `\HotReplace`. Il sera ouvert, écrit, puis refermé, durant chacun des

same time. For example, if you do your calls to `\HotReplace` *before* the `\begin{document}`, then, you can call:

appels à `\HotReplace`, tu peux donc choisir n'importe quel descripteur qui n'est pas utilisé en même temps que tu fais tes remplacements. Par exmeple, si tu fais tes appels à `\HotReplace` *avant* le `\begin{document}`, alors, tu peux faire :

```
\usepackage[nofile]{hotplug}
\makeatletter
\let\HotFile\ext@toc
\makeatother
```

4 How it works Comment ça marche

A technical point you might already know: suppose a `\toto` macro defined with:

Petit rappel technique : soit une commande `\toto` définie par :

```
\def\toto(#1,#2){bla \bla bla}
```

The TEX primitive `\meaning\toto` produces:

La primitive TEX `\meaning\toto` produit :

```
macro:(#1,#2)->bla \bla bla
```

We will heavily use it.

On va abondament s'en servir.

5 Implementation Implémentation

```
1 ⟨*hotplug⟩
2 \ProvidesPackage{hotplug}[2004/05/31 v1.00 Hot replace in a command def]
3 ⟨/hotplug⟩
```

`\keep@prefix`

The LATEX macro `\strip@prefix` reads the result of a `\meaning` and only keep the body of the macro (removing the arguments, it only keeps what is after >). In the same spirit, we define here a `\keep@prefix` which keeps this prefix.

La macro `\strip@prefix` de LATEX lit le genre de choses produites par `\meaning` et ne garde que le corps de la macro (tout ce qui est après le >). Dans le même esprit, on définit `\keep@prefix` qui garde la définition des arguments.

```
4 ⟨*hotplug⟩
5 \def\keep@prefix #1:#2->#3\finkeep{#2}
6 ⟨/hotplug⟩
```

`\HotReplace`

This macro will do the whole job. It uses a lot of `\expandafter`, thus it's almost im-

C'est cette macro qui fait tout le travail. Elle utilise énormément de `\expandafter`. Elle

possible to read it. It will be commented in details, but remains hard to understand.

In the following explanations, we will suppose that the replace is

```
\HotReplace\commande\from\to
```

est donc presque illisible. Elle sera très commentée, mais pas forcément compréhensible pour autant.

Dans les explications, on suppose qu'on fait le changement

```
7 ⟨*hotplug⟩
8 \def\HotReplace#1#2#3{%
9 % Change #2 pour #3 dans #1
```

We store the string `##1` in a token register.

On stocke la chaîne `##1` dans un registre de token.

```
10   \@temptokena{##1}
```

The following lines are hard to read. The first part expanded is `\string#2`, which produces `\from` as a string (not the token `\from`). Next is expanded the `\the\@temptokena` which produces the string `##1`. Finally, the affectation is donce. So, this is more or less similar to

Les lignes suivantes sont difficiles à lire. Le premier morceau exécuté est `\string#2`, qui produit `\from` sous forme d'une chaîne (pas le token `\from`). Ensuite est exécuté le `\the\@temptokena` qui produit la chaine `##1`. Enfin, l'affectation à lieu, c'est donc vaguement équivalent à

```
\@temptokena{##1\from##2\endHplug}
```

with this restriction that we have the *string* `\from` and the token `\endHplug`.

à cette nuance qu'on a affaire à la chaîne `\from` et au token `\endHplug`.

```
11   \expandafter\expandafter\expandafter\@temptokena
12   \expandafter\expandafter\expandafter{%
13    \expandafter\the
14    \expandafter\@temptokena
15    \string#2 ##2\endHplug}
```

What we are trying to do is to define a macro which look like that:

Ce qu'on cherche à faire, c'est définir une macro qui soit de cette forme là :

```
\def\preHplug#1\from#2\endHplug{\def\tmp@pre{#1}}
```

where we have the string `\from`. Such that if this macro reads the body of `\commande`, as it is given by `\meaning` (it gives back a string), we will have in `\tmp@pre` all the part of the body that is before `\from`.

The kind of argument expected by this macro is stored in `\@temptokena`, as we've seen before. So, those lines will do the job and define our strange `\preHplug`.

où l'on à la chaîne `\from`. De telle sorte que si cette macro lit le corps de `\commande`, tel qu'il est donné par `\meaning` (la primitive renvoie un chaîne), on récupère dans `\tmp@pre` toute la partie du corps de la macro qui se trouve avant `\from`.

La forme d'argument attendue par cette macro est contenue dans `\@temptokena`, comme on vient de voir. Donc ces quelques lignes vont faire le boulot et définir notre `\preHplug`.

```
16   \expandafter\def
```

```
17  \expandafter\preHplug
18  \the\@temptokena{\def\tmp@pre{##1}}
```

Similarily, we build a second macro, but one which stores what is after `\from` in the body of `\commande`.

On crée une seconde macro sur le même modèle mais qui garde tout ce qui se trouve après `\from` dans le corps de `\commande`.

```
19  \expandafter\def
20  \expandafter\postHplug
21  \the\@temptokena{\def\tmp@post{##2}}
```

We now have two funny macros, `\preHplug` and `\postHplug`, which if called properly will cut the body of `\commande` in two pieces.

On se trouve donc à la tête de deux macros `\preHplug` et `\postHplug` qui, convenablement appellées, vont nous découper le corps de `\commande` en deux morceaux.

We want to call something like

On cherche donc à appeler un truc du genre

```
\preHplug\strip@prefix\meaning\commande\endHplug
```

to take the beginning of the body (and the same with `\post`), but it needs to be expanded in a strange order: first expand `\meaning`, which will produce the whole text, then `\strip@prefix` which will remove the part `macro:...->`, then finaly expand `\preHplug`. This is why there are so many `\expandafter`.

pour récupérer le début (et la même chose avec `\post`), mais il faut exécuter les choses dans un ordre bizare : d'abord `\meaning`, qui va produire le texte complet, puis `\strip@prefix` qui va retirer la partie `macro:...->`, puis enfin exécuter la macro `\preHplug`. C'est ce qu'on fait avec la multitude de `\expandafter`.

```
22  \expandafter\expandafter
23  \expandafter\preHplug
24  \expandafter\strip@prefix
25              \meaning#1\endHplug
```

We now have in `\tmp@pre` the body of the macro `\commande` up to `\from`, without it. We do the same with `\postHplug` to get `\tmp@post`.

On a donc dans `\tmp@pre` le corps de la macro jusqu'à `\from` exclus. On fait le même appel, avec `\postHplug` pour récupérer `\tmp@post`.

```
26  \expandafter\expandafter
27  \expandafter\postHplug
28  \expandafter\strip@prefix
29              \meaning#1\endHplug
```

We now want to get the kind of arguments expected by `\commande`, which is a bit easier: `\keep@prefix` have been previously defined to do that. It works a bit like `\preHplug` but was easier do define, the required delimiter in the arguments (namely the strings `:` and `->`) where already known, which is not the case for `\preHplug` where the delimiter (the string `\from`) was only learned recently.

On cherche maintenant à récupérer la forme d'argument attendue par `\commande`, c'est un tout petit peu plus simple, `\keep@prefix`, qui fonctionne un peu comme `\preHplug` a déjà été définie (on pouvait la définir avant, on connaissait les délimiteurs nécessaires dans ses arguments, ce qui n'est pas le cas pour `\preHplug`).

```
30  \edef\tmp@args{\expandafter\keep@prefix\meaning#1\finkeep}
```

We now have in `\tmp@args` the kind of arguments of `\commande`, in `\tmp@pre` the beginning of its definition, and in `\tmp@post` the end of its definition.

We want TEX to read a brand new definition of the macro, but now, it have been transformed in a string. It cannot anymore be transformed straightly in a real macro. For that purpose, we are going to write in a file the *string* which defined the macro, and then ask TEX to read this brand new file.

We are going to write in a temporary file a string like

```
\def\commande\tmp@args{\tmp@pre\to\tmp@post}
```

string in which all the macros a expanded to their definitions, except `\def`, `\commande`, and `\to`, and where the braces are kept.

First, we open the file:

```
31  \immediate\openout\HotFile hotplug.tmp\relax
```

Then, we write, with `\noexpand` before the part to be kept unchanged (the three macros and the braces).

```
32  \immediate\write\HotFile{%
33    \noexpand\def
34    \noexpand#1%
35      \tmp@args\noexpand{\tmp@pre\noexpand#3\tmp@post\noexpand}}\relax
36  \immediate\closeout\HotFile
```

Now, the file contains TEX code which, if read, will redefine `\commande` while changing the first appearance of `\from` by `\to`. So we do, and the job is ended.

```
37  \input{hotplug.tmp}
38 }
39 ⟨/hotplug⟩
```

The last thing to be done is to allocate a write handle for our temporary file. Our file is *really* temporary, since it is only used during the use of the macro itself. If you run short of files in your document (which is often my case), then you can bypass this by using any other file descriptor which is not used when

On a donc maintenant dans `\tmp@args` la forme d'arguments de `\commande`, dans `\tmp@pre` le début de la définition, et dans `\tmp@post` la fin de cette définition.

On veut que TEX lise une nouvelle définition de la macro, mais maintenant que tout cela est sous forme de chaînes, ça n'a aucune chance de re-devenir facilement une vraie macro. Pour cette raison, on va écrire le *texte* de notre nouvelle définition dans un fichier, puis demande à TEX de bien vouloir lire ce nouveau fichier.

On va écrire dans un fichier temporaire la forme suivante

forme dans laquelle toutes les macros sont remplacées par leur définition sauf `\def`, `\commande`, et `\to` et où les accolades sont conservées.

On commence par ouvrir le fichier :

Puis on fait l'écriture, en mettant des `\noexpand` devant tout ce qu'on souhaite conserver tel quel (les trois commandes et les deux accolades).

Maintenant, le fichier contient du code TEX qui, s'il était lu redéfinirait `\commande` en ayant remplacé la première occurence de `\from` par `\to`. On va donc s'empresser de lire le fichier, et le tour sera joué.

Il ne nous reste plus qu'à définir le fichier temporaire dans lequel on écrite. Ce fichier est *vraiment* temporaire, il ne peut être utilisé que pendant l'exécution de la macro `\HotReplace`. Si tu es as cours de fichiers (comme c'est souvent mon cas), alors du peux éviter cette allocation, et utiliser tout autre descripteur de

you call \HotReplace.

fichier qui n'est pas utile pendant que tu utilises \HotReplace.

```
40 ⟨*hotplug⟩
41 \DeclareOption{nofile}{\hotfilefalse}
42 \DeclareOption{file}{\hotfilefalse}
43 \newif\ifhotfile
44 \hotfiletrue
45 \ProcessOptions
46 \ifhotfile
47   \newwrite\HotFile
48 \else
49   \chardef\HotFile15\relax
50 \fi
51 ⟨/hotplug⟩
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

W