Personnaliser des fichiers de style bibliographiques

Patrick W. Daly*

This paper describes program `makebst`
version 4.0-beta1 from 1999/08/26†
(including additions by Arthur Ogawa, `ogawa@teleport.com`)

1 Introduction

Ce programme TEX est destiné à être utilisé en association avec des fichiers de styles bibliographiques génériques pour produire des fichiers `.bst` adaptés utilisables avec BIBTEX. Le fichier générique, ou fichier maître, peut être traité par docstrip avec des options choisies pour obtenir le style bibliographique désiré. Dans ce but, un fichier docstrip « batch » devrait être construit. Cependant, à cause du grand nombre d'options disponibles, un système de dialogue interactif serait plus pratique.

Ce programme, `makebst`, réalise cet objectif. Il définit des macros pour fabriquer un tel fichier « batch » docstrip et organiser un menu d'options. Le menu d'options est contenu, cependant, dans le fichier maître lui-même, puisque les deux sont intimement liés. Donc différents fichiers maîtres avec des structures d'options totalement différentes peuvent être utilisés.

Le programme « batch » pourrait en fait être construit avec un éditeur sans appeler `makebst`, mais ce programme facilite vraiment la tâche.

Entre parenthèses, l'exécution du docstrip ne peut être effectuée qu'au moyen d'une tache différée. L'exécution interactive de docstrip insère dans le texte un préambule et un postambule par défaut, ce dernier contenant une commande `\endinput` que BIBTEX ne comprendra pas.

2 Le fichier maître

Le fichier maître est un fichier de style bibliographique BIBTEX contenant dont le contenu est dépendant des options de docstrip. Les options sont choisies lorsque docstrip est exécuté, soit interactivement soit par une tache différée.

*Traduction par Jean-Pierre DRUCBERT, amendée par Nicolas LE NOVÈRE le 12 avril 2000. Titre original : « Customizing Bibliographic Style Files »

†Work on `custom-bib` 4.00 was supported by the American Physical Society

Supposons que l'une des options s'appelle **xyz**. Alors les choix suivants sont possibles :

%<xyz> *une ligne de code*

inclut cette seule ligne de code ;

%<!xyz> *une ligne de code*

exclut cette seule ligne ;

%<*xyz>
plusieurs lignes de code
%</xyz>

inclut toutes les lignes imbriquées ;

%<*!xyz>
plusieurs lignes de code
%</!xyz>

exclut toutes les lignes imbriquées.

Les options peuvent être combinées logiquement : le symbole | est un ou logique, & un et logique, ! un non logique ; des parenthèses (et) peuvent être utilisées pour regrouper des options.

2.1 Utilisation avec docstrip

(La synthaxe des commandes docstrip présentée ici est celles des versions 2.4 ou ultérieures, sorties à partir de décembre 1996.)

Afin d'engendrer un vrai fichier de style BibTeX avec les options choisies à partir du fichier maître, il est nécessaire d'exécuter une tache « batch » docstrip. Supposons que le fichier maître se nomme **maître.mbs**, le fichier de style BibTeX résultant devant être **neuneu.bst**, et le fichier « batch » lui-même s'appelle **neuneu.dbj**. Pour produire ceci avec les options, par exemple, **xyz** et **abc**, le fichier batch ressemblera à :

```
\input docstrip

\preamble
Ceci est destiné au Journal des Résultats Neuneus
\endpreamble

\postamble
Fin du fichier bst personnalisé
\endpostamble

\keepsilent

\askforoverwritefalse

\generate{\file{neuneu.bst}{\from{maître.mbs}{xyz,abc}}}
\endbatchfile
```

Un préambule n'est pas nécessaire, bien qu'il soit conseillé d'inclure des commentaires sur l'application du style bibliographique. Un postambule *est* vital, sinon docstrip ajoutera par défaut `\endinput` à la fin du fichier, ce que BibTeX ne comprendra pas. La commande `\keepsilent` est facultative et ne fait que supprimer les messages de docstrip pendant le traitement. De manière similaire le `\askforoverwritefalse` supprime le message d'avertissement quand un fichier portant le même nom va être écrasé.

2.2 Le fichier menu

Ce programme, `makebst`, simplifie la création du fichier « batch ». Pour le faire, il a besoin d'informations sur les options disponibles. Ces informations doivent être rangées selon un format spécial, dans le fichier maître lui-même. Alternativement, ces informations peuvent être extraites et rangées dans un fichier de même nom de base mais avec l'extension `.opt`. *Cette possibilité n'est pas recommandée car elle peut conduire à des incohérences !* Le format des informations de menu est illustré plus loin dans la section 4.

Dans le fichier maître, ces informations doivent être placées entres les directives docstrip `%<*options>` ... `%</options>` et *doivent* être terminées par une commande `\endoptions`. Elles peuvent aussi comprendre un nombre quelconque de commentaires. Le reste du fichier doit être encadré par des directives `%<*!options>` ... `%</!options>` pour l'exclure lorsque les informations de menu sont extraites par docstrip.

Un menu d'exemple du fichier maître pour sélectionner une ou aucune des options `xyz` *ou* `zyx` ressemblerait à :

```
<*options>
\mes{Choisissez l'une des options suivantes}
\optdef{f}{xyz}{Option avant}{Faire quelque chose en avant}
\optdef{r}{zyx}{Option arrière}{Faire quelque chose en arrière}
\optdef{*}{}{Aucune de celles-ci}{}
\getans
\endoptions
</options>
<*!options>
. . . . . .
</!options>
```

Une explication de ces commandes sera donnée dans la section 4.

Les informations de menu peuvent être extraites du fichier maître au moyen de docstrip et rangées dans un fichier portant l'extension `.opt`. Si ce fichier est présent, `makebst` offre la possibilité de le lire à la place du fichier maître, bien que ceci *ne soit pas* recommandé, comme signalé plus haut.

3 Le programme `makebst`

3.1 Installation du `makebst`

Le programme `makebst` est livré comme un fichier source documenté nommé `makebst.dtx`, qui doit être transformé par docstrip afin d'extraire le vrai 'programme' `makebst.tex`. La manière la plus simple de procéder est simplement de traiter le fichier d'installation `makebst.ins` avec TEX ou LATEX,

`tex makebst.ins`
ou bien
`latex makebst.ins`

trois variants de `makebst` peuvent en fait être extraits : le variant de base produit une liste uniquement des options qui ont été sélectionnées ; Une version plus complète (par défaut) produit une liste de toutes les options offertes, les options rejetées étant transformées en commentaire ; La troisième version ajoute également des commentaires plus nombreux. Cependant, même au sein des deux premières versions, l'utilisateur se verra proposer de manière interactive des options additionnelles appartenant aux autres.

On peut choisir la version désirée en editant `makebst.ins`.

On peut également décider si les fichiers `.dbj` seront conformes à la syntaxe de docstrip version 2.4 ou non. Par défaut, `makebst.ins` teste la version courrante et automatiquement configure `makebst.tex` pour produire la syntaxe correcte. Ceci également peut être changé en éditant `makebst.ins` (Notez que l'ancienne syntaxe est toujours comprise par les nouvelles versions de docstrip).

Rappel : l'ancienne syntaxe requiert `\def\batchfile{`⟨*fichier*⟩`}` sur la première ligne du « batch » docstrip où *fichier* est le nom du fichier « batch » lui-même. La nouvelle syntaxe ne nécessite pas cette ligne mais requiert `\endbatchfile` tout à la fin. L'avantage de la nouvelle syntaxe est qu'on peut éditer et renomer un tel fichier sans avoir à changer son nom sur la première ligne. L'ancienne syntaxe entraine une grande frustration si l'on oublie de changer *filename*.

Une autre difference entre les syntaxes (En fait introduite dans la version 2.3 en Juin 1996) est l'utilisation de la commande `\generate` à la place de `\generateFile`. Son avantage est la possibilité d'extraire plusieurs fichiers en une seule fois, possibilité qui n'est pas exploitée du tout par `makebst`.

3.2 Exécution de `makebst`

C'est en fait un programme TEX, bien qu'il puisse aussi être exécuté sous LATEX. En ce sens, il est comme docstrip lui-même. Donc exécutez ce programme par

`tex makebst` ou bien
`latex makebst`

ou quelque chose d'analogue.

Le programme demande d'abord le nom du fichier maître. C'est le fichier contenant toutes les commandes de styles bibliographiques possibles, avec des directives docstrip pour une sortie sélective. Un nom par défaut est proposé, ainsi qu'une extension par défaut (`.mbs`).

Ensuite, le programme demande le nom du fichier de sortie, c'est-à-dire le fichier `.bst`. Ici l'extension est optionnelle, étant `.bst` par défaut. Ce nom détermine aussi le nom du fichier de travail différé, qui aura le même nom de base avec le suffixe `.dbj`, pour *docstrip batch job*.

Le vrai questionnaire commence alors. Toutes les informations pour les menus sont contenus dans le fichier de style bibliographique maître. La raison en est que les informations de menu doivent se conformer aux options disponibles dans le fichier maître, et il est donc raisonnable qu'un seul fichier doive contenir les deux. Le fichier maître n'est lu que jusqu'à la commande `\endoptions`.

Enfin, le fichier de travail différé est fermé, et on demande à l'usager s'il faut l'exécuter. S'il n'accepte pas cette offre, ou s'il édite ultérieurement ce fichier « batch », alors il doit l'exécuter à la main par une commande telle que

`tex` *bstname*`.dbj`

4 Les informations du menu

L'ensemble de questions dans le questionnaire doit correspondre aux options disponibles dans le fichier maître. Pour cette raison, les informations de menu sont contenues dans le fichier maître lui-même. Le programme `makebst` fournit les macrocommandes qui sont utilisées dans le fichier menu pour simplifier l'écriture et le traitement des informations de menu.

`\mes` Pour imprimer un message sur le terminal, utilisez `\mes{`⟨*texte*⟩`}`. Un retour à la ligne peut être forcé dans le *texte* au moyen de `^^J`.

`\ask` Pour interroger l'usager et lire la réponse, utilisez `\ask{`*com*`}{`⟨*texte*⟩`}`, qui écrit le *texte* sur le terminal et place la réponse dans la commande *com*.

`\optdef` Presque toutes les interrogations consistent en une liste d'options mutuellement exclusives, dont l'une est l'option par défaut. Pour chaque élément de la liste, il faut spécifier la réponse au clavier qui la sélectionnera, le nom effectif de l'option docstrip qui la réalise, et deux morceaux de texte explicatif. Par exemple,

```
\optdef{a}{abr}{Abbreviations}{of such words}
```

signifie que `abr` est le vrai nom d'option docstrip qui est sélectionné en tapant `a`. Les deux textes explicatifs sont écrits sur le terminal immédiatement comme faisant partie du menu, mais seul le premier texte est renvoyé en écho lorsque la sélection est faite (pour confirmation) et est aussi écrit dans le fichier de travail différé (comme commentaire).

L'option par défaut doit avoir la réponse `*`.

`\getans` Un menu est écrit sur le terminal, d'abord avec une commande `\mes` pour afficher le sujet, puis avec une séquence d'instructions `\optdef`, chacune d'elle écrivant leur texte sur le terminal. La réponse est alors lue et traitée par `\getans`, qui écrit la réponse dans la commande `\ans` et écrit l'option docstrip appropriée dans le fichier travail différé. Si la réponse ne correspond à aucune de celles de la liste du menu, elle prend la valeur `*` ; s'il n'y a pas de `*` dans la liste, alors `\ans` prend la dernière valeur entrée. La commande `\ans` est encore disponible par la suite pour toute vérification supplémentaire qui pourrait être nécessaire.

Un menu typique se présente comme suit :

```
\mes{^^JJOURNAL VOLUME NUMBER:}
\optdef{*}{}{Volume plain}{as vol(num)}
\optdef{i}{vol-it}{Volume italic}%
        {as {\string\em\space vol}(num)}
\optdef{b}{vol-bf}{Volume bold}%
      {as {\string\bf\space vol}(num)}
\optdef{d}{vol-2bf}{Volume and number bold}%
      {as {\string\bf\space vol(num)}}
```

```
\getans
```

`\beginoptiongroup` Une autre structure pour l'interrogation est procurée par la séquence `\beginoptiongroup` ...`\endoptiongroup` qui devrait agir comme un récipient pour les commandes `\mes` ...`\optdef` ...`\getans` décrites précédemment. Par exemple :

```
\beginoptiongroup{JOURNAL VOLUME NUMBER:}{}
\optdef{*}{}{Volume plain}{as vol(num)}
\optdef{i}{vol-it}{Volume italic}%
       {as {\string\em\space vol}(num)}
\optdef{b}{vol-bf}{Volume bold}%
      {as {\string\bf\space vol}(num)}
\optdef{d}{vol-2bf}{Volume and number bold}%
      {as {\string\bf\space vol(num)}}
\getans
\endoptiongroup
```

présente le même effet que l'exemple précédent. L'avantage de ce groupe d'option est de procurer un format unique pour toutes les interrogations et d'avoir une apparence homogène au sein du fichier généré.

Cette caractéristique a été ajoutée avec la version 4.0 de `makebst`.

5 Travaux « batch » plus complexes

La version 3.0 de `makebst` permet au fichier maître de définir des travaux différés plus sophistiqués, comme des fichier maîtres additionnels avec leurs propres options. Ceci est rendu possible par le fait que les options ne sont pas écrites directement dans la commande `\generate`, comme dans les versions antérieures, mais dans une commande `\MBopts`. Le fichier « batch » contient alors quelque chose comme :

\def\MBopts{\from{⟨*source.ext*⟩**}{%**
  lignes de la session menu
 }}
\generate{\file{⟨*output.ext*⟩**}{\MBopts}}**

Normalement les *lignes de la session menu* contiennent seulement les options docstrip. Cependant le fichier maître pourrait ajouter d'autres choses à la définition de `\MBopts`, et même la fermer et commencer une nouvelle définition. Il lui suffit de vérifier que les accolades sont appariées.

`\MBaskfile` Un certain nombre de macros sont fournies, qui sont utilisées par `makebst` lui-même, pour simplifier la fabrication de menus complexes. Pour demander le nom d'un fichier de manière interactive,

\MBaskfile{⟨*Texte de demande*⟩**}(**⟨*root.ext*⟩**){**⟨*io*⟩**}***fnom*

peut être utilisé, où *root.ext* est le nom par défaut du fichier, *io* est `i` (pour « input ») si le fichier doit déjà exister, et *fnom* est la commande qui reçoit le nom du fichier. Le nom de base sera mis dans `\froot`, le suffixe dans `\fext`.

`\wr` Le texte est écrit dans le fichier « batch » par

\wr{⟨*texte*⟩**}**

Toute commande dans le *texte* qui doit être écrite telle quelle doit être précédée de `\string`.

`\MBswitch` Puisque toutes les accolades dans le *texte* doivent être appariées, il faut faire quelque chose de spécial pour pouvoir les imprimer comme des caractères normaux. La commande `\MBswitch` effectue cela ; les parenthèses `( )` remplacent `{ }` comme délimiteurs. Ceci doit toujours être fait à l'intérieur de `\begingroup`...`\endgroup`.

Comme exemple, supposons que le fichier maître ne contienne que la moitié du codage pour le fichier `.bst`, l'autre moitié étant dans un parmi plusieurs autres fichiers maîtres. Nous devons demander quel est ce second fichier, l'inclure pour ces options, et nous assurer que `\MBopts` le sait. Le code suivant dans le fichier maître fera tout cela.

```
\MBaskfile{Nom du second fichier fichier maître}(aa.mbs)i\xfile
\begingroup\MBswitch
\wr(\string\MBopta})
\wr(\string\from{\xfile}{\string\MBoptb}})
\wr(\string\def\string\MBopta{\pc)
\endgroup
informations normales de menu pour le premier fichier
\begingroup\MBswitch
\wr(}\string\def\string\MBoptb{\pc)
\endgroup
\input\xfile\relax
\begingroup\MBswitch
\wr({\pc)
\endgroup
\endoptions
```

Le ficher `.dbj` résultant contient

```
\def\MBopts{\from{premier.mbs}{%
\MBopta}
\from{second.mbs}{\MBoptb}}
\def\MBopta{%
premier jeu d'options
}\def\MBoptb{%
second jeu d'options
{%
  }}
\generate{\file{exemple.bst}{\MBopts}}
```