Tame the BeaST

BibTEX, de B à X...

Nicolas Markey
markey@lsv.ens-cachan.fr

J'ai essayé, dans cette documentation de 44 pages, de présenter de manière la plus exhaustive et la plus claire possible, les différentes possibilités qu'offre BibTEX pour faire une bibliographie. En effet, les documentations existantes, essentiellement celles de l'auteur de BibTEX [Pat88a, Pat88b] et celles des documentations LATEX [Lam97, GMS98, Des00, ...], sont très incomplètes.
Le titre de cette doc, qui signifie *grosso modo* « maîtrise la bête », contient trois lettres majuscules : BST, qui est le nom d'extension des fichiers définissant les styles de bibliographie. Le sous-titre signifie que j'ai essayé d'être aussi complet que possible...

Table des matières

Première partie

Faire une bibliographie avec LaTeX sans se prendre la tête

BibTeX, et plus globalement tout ce qui concerne la bibliographie dans LaTeX, est souvent vu comme quelquechose de magique, que beaucoup utilisent sans chercher à le comprendre. Mais une bibliographie, ce n'est rien de plus qu'une liste de documents auquels ont fait référence dans un article. Sans réfléchir plus loin, on ferait donc simplement ceci :

```
Ceci est le corps du document, dans lequel je fais des r\'ef\'erences \`a
[\ref{doc1}], mais aussi \`a [\ref{doc2}].
\section*{R\'ef\'erences}
\begin{enumerate}
  \renewcommand\labelenumi{[\theenumi]}  %% un \item sera numerote entre crochets...
  \item \label{doc1} Michel Goossens, Franck Mittelbach et Alexander
        Samarin, \emph{The \LaTeX Companion}, Addison Wesley, 1998.
  \item \label{doc2} Leslie Lamport, \emph{\LaTeX: A Document Preparation
        System}, Addison Wesley, 1997.
\end{enumerate}
```

qui donnerait

> Ceci est le corps du document, dans lequel je fais des références à [1], mais aussi à [2].
>
> **Références**
>
> [1] Michel Goossens, Franck Mittelbach et Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1998.
>
> [2] Leslie Lamport, *LaTeX : A Document Preparation System*, Addison Wesley, 1997.

C'est, à quelques détails près, ce qui est fait avec l'environnement `thebibliography` — qui est un environnement de liste comme `enumerate` — et avec `\bibitem` — dont la définition est basée sur `\item`. La principale différence est, finalement, que les références croisées basées sur `\label` et `\ref` ne peuvent se faire que sur des chiffres, alors qu'on voudrait pouvoir avoir des références plus générales, comme par exemple [GMS98].

Tout cela est l'objet de cette première partie : « Comment faire une bibliographie *sans* utiliser BibTeX ? » Même si ce n'est pas le but de cette doc, c'est, je pense, une étape indispensable pour comprendre la suite.

1 L'environnement `thebibliography`

L'environnement `thebibliography` n'est pas défini dans LaTeX (ni dans TeX, bien sûr). Il doit être défini dans la classe qui est chargée au début du document, comme article.cls ou report.cls. Comme annoncé précédemment, c'est *grosso modo* un environnement de type `list`, précédé par le commencement d'une nouvelle section (ou d'un nouveau chapitre, ça dépend de la classe utilisée[1]). Un problème

[1] C'est pour cette raison que cet environnement n'est pas défini dans LaTeX. Mais la plupart des commandes servant à la bibliographie sont définies dans LaTeX.

qui peut se poser mais que l'exemple précédent ne montre pas, c'est l'indentation : on voudra éviter, par exemple, le résultat produit, par défaut, par l'environnement `enumerate` :

Références

[GMS98] Michel Goossens, Franck Mittelbach et Alexander Samarin, *The LATEX Companion*, Addison Wesley, 1998.

[Lam97] Leslie Lamport, *LATEX : A Document Preparation System*, Addison Wesley, 1997.

C'est pour éviter ce problème que l'environnement `thebibliography` prend un argument : cet argument doit indiquer le « label » le plus long qui apparaît dans la liste.

Pour constater tout cela, regardons la définition de `thebibliography`, dans la classe article.cls par exemple :

```
\newenvironment{thebibliography}[1]
     {\section*{\refname
        \@mkboth{\MakeUppercase\refname}{\MakeUppercase\refname}}%
```

Comme prévu, on commence une section[2]. Les en-têtes sont également définies, en cas de besoin.

```
       \list{\@biblabel{\@arabic\c@enumiv}}%
            {\settowidth\labelwidth{\@biblabel{#1}}%
```

Comme prévu également, on commence une liste[3] qui prend deux arguments :

- le premier défini le format de la « clef » qui sera utilisé, par défaut, pour chaque élément de la liste. Ici, il est défini à partir du compteur `enumiv`, avec la commande `\@biblabel`[4]. Ça va donc donner, par défaut, [1], [2], ... Bien entendu, l'argument optionnel de `\item` permet de modifier ces valeurs.
- le deuxième argument est une liste de commandes qui sont effectuées au commencement de la liste, et qui décrivent les valeurs des différentes longueurs qu'utilise un environnement `list`. C'est ici que l'on indente la liste en fonction de l'argument de `thebibliography`, et c'est pour cette raison que l'argument doit être *le plus long label* utilisé dans la bibliographie. Ainsi, on voit souvent des commandes comme `\begin{thebibliography}{99}`, mais cela n'a lieu d'être que lorsque les labels sont numériques, et qu'il y a entre 10 et 99 documents cités. Et on pourrait remplacer l'argument `99` par `88`, par exemple, ou par n'importe quel nombre à deux chiffres, si (comme c'est le cas pour les fontes cmr) les chiffres ont tous la même largeur.

La définition de `thebibliogrphy` se poursuit avec la fin des déclaration des longueurs et compteurs utilisés par la liste :

```
             \leftmargin\labelwidth
             \advance\leftmargin\labelsep
             \@openbib@code
             \usecounter{enumiv}%
             \let\p@enumiv\@empty
             \renewcommand\theenumiv{\@arabic\c@enumiv}}%
```

La commande `\@openbib@code`, qui est vide par défaut, permet de modifier ces paramètres si nécessaire, en particulier avec l'option `openbib` des classes de document usuelles.

Enfin, à l'intérieur de cet environnement de liste, on ajuste l'espacement ainsi que quelques pénalités :

[2] C'est une section « étoilée », elle n'apparaît donc pas dans la table des matières. Voir le package tocbibind.sty pour pallier ce problème.

[3] La commande `\list` est équivalente à un `\begin{list}`.

[4] `\@biblabel` est définie dans LATEX. Elle prend un argument et ajoute simplement des crochets autour. Sa définition précise est `\def\@biblabel#1{[#1]}`.

```
        \sloppy
        \clubpenalty4000
        \@clubpenalty \clubpenalty
        \widowpenalty4000%
        \sfcode`\.\@m}
```

C'est tout pour ce qui est de l'initialisation de l'environnement `thebibliography`. La fermeture de l'environnement est beaucoup plus simple : on met un message d'alerte si rien n'a été écrit dans la bibliographie, et on termine l'environnement `list`.

```
       {\def\@noitemerr
         {\@latex@warning{Empty `thebibliography' environment}}%
        \endlist}
```

2 La commande \bibitem

À l'intérieur de l'environnement de type `list` décrit ci-dessus, il faut, comme de coutume, mettre des éléments commençant par `\item`. Encore une fois, c'est un `\item` un peu modifié, et nommé `\bibitem`. Son rôle sera non seulement d'écrire une nouvelle entrée dans la bibliographie, mais aussi de définir le « label » qui sera écrit dans le texte, au moment où on cite cette entrée. Comme nous l'avons vu précédemment, par défaut, le « label » sera `\@biblabel{\@arabic\c@enumiv}`, ce qui donne par exemple [1], mais il est bien entendu possible de modifier cela (comme avec un `\item`), pour obtenir plutôt [GMS98] par exemple. La commande `\bibitem` admettra donc un argument optionnel. Voici la définition précise de tout cela :

```
\def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

`\bibitem` appelle `\@lbibitem` si un argument optionnel est fourni, et `\@bibitem` sinon. Voyons maintenant comment ces deux commandes sont définies :

```
\def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
      {\let\protect\noexpand
       \immediate
       \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

Cette commande est utilisée lorsqu'il y a l'argument optionnel. Raisonnons, par exemple, sur la commande `\bibitem[GMS98]{companion}`. Cette commande commence par créer un `\item` pour l'entrée courante, avec comme « label » l'argument optionnel, entouré de crochets par `\@biblabel`. Elle écrit ensuite, dans le fichier `.aux` [5], la commande `\bibcite` suivie des deux arguments. Cette commande `\bibcite` est définie comme suit[6] :

```
\def\bibcite{\@newl@bel b}
```

De cette façon, lorsque le fichier `.aux` est lu par LaTeX, les `\bibcite` sont lus et exécutés, et ainsi le label `\b@companion` est créé, dans notre exemple, et renvoie `GMS98`.

Dans le cas où il n'y a pas d'argument optionnel, c'est très semblable :

```
\def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
       {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

[5] Plus précisément, dans le fichier désigné par `\@auxout`, mais en général, c'est le fichier `.aux` correspondant au fichier que l'on compile.

[6] La commande `\@newl@bel` prend trois arguments, `#1`, `#2` et `#3`, et crée une commande nommée `#1@#2` (où, bien sûr, `#1` et `#2` sont remplacés par leurs valeurs) contenant le troisième argument.

On crée un `\item`, et on écrit le `\bibcite` dans le fichier `.aux`. La seule chose à savoir, mais qui est évidente si on a compris comment ça marche, c'est que la commande `\@listctr` indique le compteur qui est utilisé par la liste courante (par la bibliographie, donc).

Tout le texte qui suit une commande `\bibitem` (et ses arguments) est écrit à la suite du « label », comme dans le cas d'un `\item` classique, jusqu'à la prochaine commande `\bibitem` ou la fin de la bibliographie[7].

Pour conclure, voici un petit exemple de bibliographie, avec deux entrées, comme elle pourrait apparaître dans un document :

```
\begin{thebibliography}{GMS98}   %% GMS98 est le plus long label.
\bibitem[GMS98]{companion} Michel Goossens, Franck Mittelbach et Alexander
     Samarin, \emph{The \LaTeX Companion}, Addison Wesley, 1998.
\bibitem[Lam97]{lamport} Leslie Lamport, \emph{\LaTeX: A Document Preparation
     System}, Addison Wesley, 1997.
\end{thebibliography}
```

Et voici le résultat :

Références

[GMS98] Michel Goossens, Franck Mittelbach et Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1998.

[Lam97] Leslie Lamport, *LaTeX : A Document Preparation System*, Addison Wesley, 1997.

3 La commande `\cite`

La commande `\cite`, c'est l'équivalent de la commande `\ref` pour les références croisées, toujours en gardant l'analogie entre la bibliographie et une liste avec référence croisées. Cette commande prend en argument la « clef » associée à l'élément auquel on fait référence. Cette commande prend également un argument optionnel, permettant d'ajouter une précision sur la référence. Par exemple quand on parle de bibliographie, on cite citer [GMS98, Chap. 13], en écrivant `\cite[Chap.~13]{companion}`.

Cette commande est donc définie de la façon suivante[8] :

```
\DeclareRobustCommand\cite{%
  \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
```

Dans ce cas, donc, s'il n'y a pas d'argument optionnel, on en ajoute un, vide... On garde aussi en mémoire, pour plus tard, une variable booléenne `@tempswa` indiquant si un argument optionnel a été donné.

Avant de voir en détail ce que fait `\@citex`, et pour comprendre comment est utilisée la variable booléenne `\@tempswa`, jetons un œil rapidement à la commande `\@cite`, qui sera utilisée par `\@citex` :

```
\def\@cite#1#2{[{#1\if@tempswa , #2\fi}]}
```

Cette commande affiche donc son premier argument, et éventuellement le second, si `\@tempswa` est vraie.

Attaquons donc maintenant la définition de `\@citex` :

```
\def\@citex[#1]#2{%
  \let\@citea\@empty
  \@cite{\@for\@citeb:=#2\do
```

[7] Avec certains packages, tout cela peut-etre un peu différent. Voir la section 4.4 à ce sujet.

[8] Ne nous préoccupons pas, pour l'instant, de `\DeclareRobustCommand`, et considérons que c'est un simple `\newcommand`. Voir le paragraphe 4.1 pour (un peu) plus de détails.

On appelle donc `\@cite`, et on va calculer son premier argument. Plusieurs « clefs » pouvant être passées à la commande `\cite`, il faut faire une boucle pour les traiter toutes, une à une.

```
      {\@citea\def\@citea{,\penalty\@m\ }%
```

À partir du deuxième passage dans la boucle, on ajoute une virgule... La pénalité interdit de couper la ligne à cet endroit. De façon générale, même si ça doit dépasser, une référence ou une liste de références ne sera jamais coupée.

```
       \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
```

Cette ligne définit `\@citeb`, la variable utilisée dans la boucle, comme étant égal à, *grosso modo*, elle-même, mais un peu trafiquée. À quoi sert-ce ? Principalement à supprimer les espaces initiaux, que la commandes `\@for` laisse dans `\@citeb`. C'est un peu sioux, mais ça marche...

```
       \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
```

Ici, on écrit, dans le fichier `.aux`, qu'on a fait une référence à `\@citeb`. C'est pas indispensable dans le processus actuel, mais c'est vital pour les bibliographies avec BibTEX, comme on le verra à la section 5.

```
       \@ifundefined{b@\@citeb}{\mbox{\reset@font\bfseries ?}%
         \G@refundefinedtrue
         \@latex@warning
           {Citation `\@citeb' on page \thepage \space undefined}}%
```

Au cas où la référence n'existe pas, on remplace le « label » par un point d'interrogation, et on prévient l'utilisateur...

```
         {\hbox{\csname b@\@citeb\endcsname}}}}{#1}}
```

...et sinon, on ajoute une `\hbox` avec le contenu du « label » créé précédemment par `\bibcite` (*cf.* page 4). On exécute donc tout cela pour toutes les références demandées. Et on n'oublie pas, bien sûr, de donner le deuxième argument que réclame `\@cite`, qui est ici l'argument optionnel `#1` de `\cite`.

En pratique, la complexité de la définition est complètement transparente pour l'utilisateur, qui peut simplement taper `\cite{companion, lamport}` pour obtenir [GMS98, Lam97].

4 Petits trucs en vrac

4.1 Qu'est-ce que `\DeclareRobustCommand` ?

Une commande qui a un argument facultatif, comme `\cite`, est un exemple de commande *fragile*. Cela pose un problème lorsque ces commandes sont utilisées à l'intérieur d'autres commandes (comme par exemple si on place un `\cite` dans une commande `\section`). Pour éviter ces problèmes, on peut définir ces commandes avec `\DeclareRobustCommand` au lieu de `\newcommand`. Une autre solution serait de faire précéder la commande `\cite`, dans les cas qui posent problème, par `\protect`.

4.2 Comment changer le titre de la bibliographie ?

En regardant la définition de l'environnement `thebibliography`, ça doit sauter aux yeux : il suffit de redéfinir la commande `\refname`. Cependant, attention... Dans la classe report.cls, c'est effectivement `\refname` qui est utilisé, et il faut effectivement la redéfinir (la valeur par défaut est « Références »). Mais dans les classes article.cls et book.cls, c'est `\bibname`, qu'vaut « Bibliographie » par défaut.

Par exemple, donc, avec la classe report.cls, on écrira

```
\renewcommand{\refname}{Quelques r\'ef\'erences}
```

alors qu'avec book.cls et article.cls, on devra mettre

```
\renewcommand{\bibname}{Quelques r\'ef\'erences}
```

4.3 Comment ajouter du texte avant la première entrée ?

Si on place directement du texte après le début de l'environnement `thebibliography`, on déclenche une erreur, puisque tout environnement `list` nécessite en premier lieu un `\item`. L'idée est donc simplement de mettre un `\item` vide, puis de reculer un peu pour revenir au niveau de la marge de gauche :

```
\begin{thebibliography}{AAA00}
\item[]
\hskip-\leftmargin
\begin{minipage}{\textwidth}
\hskip15pt Voici quelques r\'ef\'erences utiles pour tout savoir sur \LaTeX, et
qui se trouvent dans toute bonne librairie. On pourra trouver de nombreuses
autres sources d'information sur le web, comme par exemple la \textsc{f.a.q.}
du groupe \textsf{fr.comp.text.tex}.
\end{minipage}
\bigskip
\bibitem[GMS98]{companion} Michel Goossens, Franck Mittelbach et Alexander
     Samarin, \emph{The \LaTeX Companion}, Addison Wesley, 1998.
\bibitem[Lam97]{lamport} Leslie Lamport, \emph{\LaTeX: A Document Preparation
     System}, Addison Wesley, 1997.
\end{thebibliography}
```

Ce qui donnera :

Références

Voici quelques références utiles pour tout savoir sur LaTeX, et qui se trouvent dans toute bonne librairie. On pourra trouver de nombreuses autres sources d'information sur le web, comme par exemple la F.A.Q. du groupe fr.comp.text.tex.

[GMS98] Michel Goossens, Franck Mittelbach et Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1998.

[Lam97] Leslie Lamport, *LaTeX : A Document Preparation System*, Addison Wesley, 1997.

4.4 Redéfinition de `\bibitem`

Certains styles redéfinissent la commande `\bibitem`, et en fait surtout `\@bibitem` et `\@lbibitem`, de telle sorte que chaque entrée de la bibliographie doive obligatoirement se terminer par un changement de paragraphe (ou une ligne vide). Un exemple de package qui fait cela est backref.sty.

D'autres packages le redéfinisse de telle sorte que l'argument optionnel devient obligatoire. Par exemple, apalike.sty.

J'ai rien à ajouter à ce sujet, il faut juste savoir que ça arrive, et que, si on l'oublie, l'erreur n'est pas toujours facile à identifier.

4.5 Comment changer les crochets en parenthèses, dans les références ?

Comme on l'a vu, c'est la commande `\@biblabel` qui est chargée d'ajouter les crochets dans la liste des références. Il suffit de la redéfinir, par exemple comme ça :

```
\makeatletter % pour que le @ ne soit pas "interprete"
\def\bibleftdelim{(}
\def\bibrightdelim{)}
```

```
\def\@biblabel#1{\bibleftdelim #1\bibrightdelim}
\makeatother % pour que le @ soit a nouveau "interprete"
```

De cette manière, on peut changer facilement les crochets en parenthèses, par exemple, en redéfinissant les délimiteurs.

4.6 Comment changer les crochets en parenthèses, dans les citations ?

Dans ce cas, il suffit de changer la commande `\@cite`. Par exemple, en reprenant ce qui précède :

```
\makeatletter
\def\@cite#1#2{\bibleftdelim{#1\if@tempswatrue , #2\fi}\bibrightdelim}
\makeatother
```

4.7 Que peut-on mettre comme « clef » pour les références bibliographiques ?

Je ne sais pas exactement, à vrai dire. Si je mets cette question, c'est surtout pour dire que ça ne sert à rien d'aller mettre des caractères exotiques, y en a bien assez avec les 26 lettres de l'alphabet, non ? Quoi qu'il en soit, les virgules, accolades, « contre-oblique[9] » sont interdits. Le reste, tu essayes, je sais pas et encore une fois (mais plus crûment, ce coup-ci) je trouve la question débile.

Conclusion

En théorie, cette doc pourrait s'arrêter là : on sait faire une bibliographie, maintenant, inutile d'aller voir plus loin... Sauf que c'est un peu lourd de recopier à chaque nouvel article toutes les références dont on a besoin. Il serait plus pratique d'avoir une *base de données* de références, dont un programme extrairait le contenu, et le mettrait en page. Évidemment, si je dis ça, c'est parce que ça existe, et c'est une brillante transition vers la suite...

[9] On dit aussi (surtout) backslash...

Deuxième partie
Utiliser BibTEX

5 Comment ça marche ?

BibTEX est donc, pour parler de façon générale, un utilitaire qui gère une base de données : il sait extraire, trier et mettre en page des éléments d'une base de données, afin d'en faire un résultat compilable par LATEX.

Évidemment, dit comme ça, c'est le rêve. La réalité est légèrement plus complexe : pour que BibTEX sache quelles entrées il doit extraire de la base de données, il faut le lui dire. Il faut donc passer un coup de LATEX avant. Et forcément un coup après, pour que LATEX prenne en compte le résultat. Voici comment ça marche de façon précise :

- la première étape est donc la compilation du document LATEX. À ce moment, si rien n'a été extrait de la base de données, les références ne sont pas indiquées, bien entendu. Par contre, à chaque fois qu'une référence bibliographique est utilisée, c'est indiqué dans le fichier `.aux` [10]. Au cours de cette compilation, LATEX va également écrire dans le fichier `.aux` quelle base de données est utilisée, et quel «style » il faut appliquer pour la mise en page de ses éléments ;
- On lance alors BibTEX, qui va regarder dans le fichier `.aux` les informations qui le concernent : le style (fichier d'extension `.bst`) qu'il doit utiliser, la ou les bases de données (dont l'extension est `.bib`) d'où il doit extraire les renseignements bibliographiques, ainsi que les éléments à extraire. Il va ensuite faire ce qu'on lui a demandé, c'est à dire l'extraction et la mise en forme des données, et mettre le tout dans un fichier d'extension `.bbl`. Les messages d'avertissements et d'erreur sont mis dans un fichier d'extension `.blg` ;
- Afin de prendre en compte ces données, on relance LATEX sur le document. Le fichier `.bbl` qui vient d'être créé est inclus. Les `\bibitem` sont donc exécutés, donc LATEX définit les labels à utiliser dans le fichier `.aux` [11]. Cependant, il les écrit, mais il ne les « crée » pas pour l'exécution actuelle (il n'a aucun moyen de les connaître au début de l'exécution), les références ne seront donc pas bonnes après cette étape ;
- Il faut donc un dernier passage par LATEX, qui permet d'avoir les bonnes références dans l'ensemble du document.

Il faut donc, dans le meilleur des cas, une compilation de LATEX, une exécution de BibTEX et deux compilation de LATEX pour que le document soit correct. Il y a cependant des cas (si, par exemple, un élément de la bibliographie en cite un autre) où il faudra relancer BibTEX et LATEX. Pour ceux qui comprenne ce que ça veut dire, on va faire ça :

$$\text{LATEX (BibTEX LATEX)}^{+}\ \text{LATEX}.$$

Regardons maintenant plus précisément les commandes à utiliser pour faire marcher cette petite merveille. En fait, rien de bien nouveau si on a compris à peu près ce qui se passe : c'est pareil que dans la partie précédente, sauf que l'environnement `thebibliography` et son contenu sont générés par BibTEX. Il y a cependant deux nouvelle commandes à étudier : celles qui définissent les bases de données et le style bibliographique à utiliser :

- `\bibliographystyle` est la commande qui indique le style à utiliser par BibTEX. Sa définition est assez simple :

```
\def\bibliographystyle#1{%
  \ifx\@begindocumenthook\@undefined\else
    \expandafter\AtBeginDocument
```

[10] C'est pour cette raison que la commande `\cite` écrit une ligne `\citation` dans le fichier `.aux` (*cf.* page 6).
[11] Voir la définition de `\bibitem` à la page 4.

```
  \fi
    {\if@filesw
       \immediate\write\@auxout{\string\bibstyle{#1}}%
     \fi}}
```

Que fait-ce ? Ça écrit simplement dans le fichier `.aux` la commande `\bibstyle` suivie du nom du style à utiliser. Et `\bibstyle` est une commande qui prend un argument et ne fait rien. En effet, la style de bibliographie à utiliser est une indication que BibTEX doit prendre en compte, mais LATEX n'en a que faire.

– `\bibliography` est la commande qui indique quelle(s) base(s) de données utiliser[12]. Là encore, ça va être écrit dans le fichier auxiliaire. Cette commande fait quand même autre chose : c'est elle qui inclut le fichier `.bbl` créé par BibTEX :

```
\def\bibliography#1{%
  \if@filesw
    \immediate\write\@auxout{\string\bibdata{#1}}%
  \fi
  \@input@{\jobname.bbl}}
```

Comme précédemment, la commande `\bibdata` prend un argument et ne fait rien, puisque l'argument ne concerne que BibTEX. La commande `\jobname`, pour ceux qui n'auraient pas compris, renvoie (par défaut) le nom du fichier qui est en train d'être compilé, sans son extension.

Pour boucler la boucle, et accessoirement, pour me répéter, le fichier `.aux` indique aussi à BibTEX quelles références il doit extraire de la base de données. Cela se fait avec les `\citation` que `\cite` écrit dans le fichier `.aux`. Tout comme `\bibstyle` et `\bibdata`, la commande `\citation` prend un argument et ne fait rien.

6 Les packages et styles qui existent déjà...

C'est là que cette doc va devenir un peu moins exhaustive : il existe une multitude de styles bibliographiques, parce que suivant la discipline en question, le type d'article et l'âge du capitaine, les règles sont différentes. Je vais essayer néanmoins de présenter quelques styles avec leurs particularités :

6.1 Les styles bibliographiques « classiques »

Ces quatre styles ont été créés par Oren PATASHNIK, que j'ai oublié de présenter, qui est en fait l'auteur de BibTEX. Ces styles s'appellent plain.bst, alpha.bst, unsrt.bst et abbrv.bst. Pour sélectionner le style plain.bst, par exemple, on écrira donc `\bibliographystyle{plain}`.

De manière globale, un style bibliographique doit s'occuper de tout. « Tout », ici, ça peut se décomposer en trois points : les différents types d'entrée possibles, avec les champs jugés pertinents pour chaque entrée[13] ; l'ordre et la mise en page de ces éléments ; et enfin, la façon de trier les entrées et d'y faire référence.

Le premier rôle du fichier de style, c'est donc de définir les types d'entrée possibles, nommées **@book**, **@article** ou encore **@inproceedings** dans les styles classiques, et pour chaque type d'entrée, les champs (*i.e.* les données) pertinents, qui seront soit obligatoires, soit facultatifs, soit simplement ignorés[14]. Le tableau suivant décrit le rôle des champs, bien que leurs noms soient généralement assez explicites. Les descriptions sont assez sommaires, mais ce tableau se retrouve dans n'importe quel bouquin sur LATEX :

[12]Dans le cas où on utilise plusieurs bases de données, leurs noms doivent être séparés par des virgules, mais sans saut de ligne, ni espace.

[13]J'insiste ici sur ce point : beaucoup pensent que les différents types d'entrée et les champs autorisés sont définis dans BibTEX, mais ils dépendent uniquement du style bibliographique.

[14]La règle générale est la suivante : un champ qui n'est ni obligatoire ni facultatif est ignoré. On peut donc ajouter autant de champs qu'on veut même s'ils ne sont pas dans la liste ci-dessous, puisqu'ils seront ignorés par les styles classiques. Il se peut, par contre, que ces champs supplémentaires soient utilisés par d'autres styles bibliographiques.

address	La ville (généralement) ou l'adresse complète de l'éditeur.
author	Les noms des auteurs. Le format d'entrée de ce champ est assez complexe, puisqu'il faut que BibTEX puisse distinguer le nom du prénom et de la particule. La section 12 y est entièrement consacrée.
booktitle	Comme son nom l'indique, c'est le titre d'un livre. C'est utilisé dans le cas où on cite une partie dudit livre.
chapter	Le numéro du chapitre (ou n'importe quelle type de sectionnement) que l'on cite dans un livre. Si ce n'est pas un chapitre, le champ `type` permet de spécifier le type de sectionnement.
crossref	Celui-la, c'est un peu particulier. Ça permet de faire des références à l'intérieur de la bibliographie. De cette façon, on peut citer un document et une partie de ce document, et dans ce cas avoir une référence croisée dans la bibliographie, ou faire hériter à une entrée de la bibliographie les champs d'une autre entrée. C'est un peu complexe, je détaillerai tout ça un peu plus loin (section 13).
edition	Indique le numéro de l'édition, ou plutôt son ordinal. On mettra par exemple `edition = "Premi\‘ere"`. C'est là que commencent à surgir les problèmes de multilinguisme, nous y reviendrons.
editor	Indique le nom du « rédacteur en chef », et non pas de l'éditeur comme on pourrait le croire. C'est juste un faux-ami de la langue anglaise, mais *editor* se traduit bien par *rédacteur en chef* (cependant, on laisse généralement *(éd.)* après le nom des éditeurs, même en français).
howpublished	Ne sert que dans les cas particuliers où le document cité n'est pas d'un type classique, comme un livre, un article de journal ou de conférence, ...
institution	Dans le cas d'un rapport, indique l'institution qui l'a publié.
journal	Le nom du journal dont est extrait un article. Souvent défini par une abréviation : BibTEX permet en effet de créer des raccourcis, et généralement, plutôt que d'entrer le nom complet à chaque fois, on préfèrera définir et utiliser ce genre d'abréviations. Voir la section 10 pour quelques détails supplémentaires.
key	permet de définir le label d'une entrée, dans le cas où il ne peut pas être « calculé » par BibTEX. Il ne sert donc pas à *forcer* le label, juste à en donner une valeur *par défaut*.
month	Bah... Le mois de parution du document, s'il est connu. On retrouve le problème des langues étrangères : il faut mettre le mois en toutes lettres, ce qui oblige à avoir deux bibliographies différentes si on écrit parfois en français et parfois en anglais. Plusieurs moyens plus ou moins évolués permettent de contourner cette difficulté, comme nous le verrons par la suite.
note	Une remarque additionnelle quelconque. Ces styles de base datant de 1985 environ, ils n'ont pas de champ `url` alors que c'est souvent bien pratique. Le champ `note`, utilisé conjointement avec le package url.sty, est souvent utilisé pour pallier ce manque.
number	Un numéro... Mais pas un numéro quelconque : ça peut être le numéro dans une série, ou le numéro d'un rapport. Pour les numéros de volume, on utilisera le champ `volume`.
organization	L'organisateur (pas une personne, plutôt une institution) d'une conférence.
pages	Les pages qui nous intéressent. Ça permet d'éviter au lecteur de se faire tout un bouquin pour trouver l'information à laquelle on fait référence. Cette information peut aussi être passée en argument optionnel de la commande `\cite` (voir page 5), auquel cas elle apparaîtra dans le texte, à l'endroit de la citation, mais pas dans la liste des références.

publisher	Pour le nom de la maison d'édition ou de l'organisme qui a publié le document cité.
school	Pour un mémoire ou une thèse, l'école où il a été réalisé.
series	Le nom d'une série d'ouvrages, d'une collection de bouquins, ...
title	Le titre du document que l'on cite. Il y a certaines règles à respecter, notamment pour les styles de base, parce que les titres sont passés en minuscules. Il y a moyen d'y remédier, je détaille ça plus loin (section 11).
type	Le type. Le type de quoi ? Ça dépend : le type de publication, au cas où c'est pas clair. Par exemple, on verra plus tard que pour une thèse de doctorat ou une thèse d'habilitation, il n'y a qu'un type d'entrée possible. C'est dans le champ `type` que l'on peut apporter la nuance supplémentaire. Mais ça peut aussi être le type de sectionnement utilisé dans l'argument du champ `chapter`, si l'entrée est `@inbook` ou `@incollection`.
volume	Le numéro de volume dans une série ou dans une collection.
year	L'année de publication du document cité.

Le tableau suivant décrit les différents types d'entrée. Là encore, ce tableau est un classique, je ne m'y attarde pas.

Type d'entrée	Champs obligatoires	Champs optionnels
@article : Article paru dans un journal.	`author`, `title`, `year`, `journal`	`volume`, `number`, `pages`, `month`, `note`
@book : Bah... Un livre.	`author` ou `editor`, `title`, `publisher`, `year`	`volume` ou `number`, `series`, `address`, `edition`, `month`, `note`
@booklet : Un *petit* livre, qui n'a pas de champ `publisher`.	`title`	`author`, `howpublished`, `address`, `address`, `month`, `year`, `note`
@conference : Article dans les actes d'une conférence, d'un colloque, d'une rencontre...	`author`, `title`, `booktitle`, `year`	`editor`, `volume` ou `number`, `series`, `pages`, `address`, `month`, `organization`, `publisher`, `note`
@inbook : Partie (un chapitre, souvent) d'un livre.	`author` ou `editor`, `title`, `chapter` ou `pages`	`volume`, `number`, `series`, `type`, `address`, `edition`, `month`, `note`
@incollection : Grosse partie d'un livre. Pas juste un petit chapitre... Il faut en particulier que cette partie ait son propre titre.	`author`, `title`, `booktitle`, `publisher`, `year`	`editor`, `volume` ou `number`, `series`, `type`, `chapter`, `pages`, `address`, `edition`, `month`, `note`
@inproceedings	Exactement pareil que **@conference**.	
@manual : Une petite doc, comme celle-ci, par exemple.	`title`	`author`, `organization`, `year`, `address`, `edition`, `month`, `note`
@masterthesis : Mémoire de D.E.A., ou équivalent.	`author`, `title`, `school`, `year`	`type`, `address`, `month`, `note`

Type d'entrée	Champs obligatoires	Champs optionnels
@misc : Quand on ne sait pas quoi mettre, on met **@misc**...	un des champs optionnels	`author`, `title`, `howpublished`, `year`, `month`, `note`
@phdthesis : Thèse de doctorat, d'habilitation, ou un autre gros truc dans le même genre.	`author`, `title`, `school`, `year`	`type`, `address`, `month`, `note`
@proceedings : Actes d'une conférence.	`title`, `year`	`editor`, `volume` ou `number`, `series`, `address`, `month`, `organization`, `publisher`, `note`
@techreport : Rapport technique, publié par un labo, un centre de recherche, ...	`author`, `title`, `institution`, `year`	`type`, `address`, `number`, `month`, `note`
@unpublished : Un document non publié. C'est souvent assez proche de l'entrée **@misc**, sauf que là, il y a un auteur et un titre.	`author`, `title`, `note`	`month`, `year`

C'est donc BibTEX qui s'occupe de la mise en ordre et en page des différents champs pour chaque entrée. L'ordre d'apparition des champs et leur mise en page sont identiques dans les quatre styles de base. Ça dépend bien entendu du type d'entrée, mais globalement, ça commence toujours par les auteurs, puis le titre. Ensuite, si c'est un article de journal, ses références ; si c'est un livres, l'éditeur et l'année ; si c'est une publication dans une conférences, les références des actes de la conférences... Là encore, je n'insiste pas, il suffit d'essayer pour voir, ou de prendre un bouquin sur LATEX, c'est souvent détaillé.

Jusqu'à maintenant, mis à part leur nom, rien ne distingue les styles bibliographiques de base. C'est donc dans le tri et les « labels » que ça va se jouer...

- Le style plain.bst a les deux caractéristiques suivantes : il trie les entrées suivant le nom de leur(s) auteur(s) (dans l'ordre alphabétique, ça va sans dire), puis, en cas d'égalité, suivant l'année de parution (dans l'ordre croissant), puis, en cas de nouvelle égalité, par ordre alphabétique du titre. S'il y a encore égalité, c'est l'ordre d'apparition dans le document qui départage, mais il faut vraiment le faire exprès... Concernant le « label », c'est du numérique, de 1 à n, dans l'ordre croissant ;
- le deuxième style s'appelle alpha.bst parce que le label qu'il calcule est alphanumérique : c'est les trois premières lettres du nom de l'auteur (ou les initiales des auteurs s'il y en a plusieurs), suivi des deux derniers chiffres de l'année. Le tri se fait en premier lieu en fonction de ce label, puis, en cas d'égalité, suivant les mêmes critères que le style plain.bst ;
- le troisième style s'appellant unsrt.bst, un zeste de perspicacité permet de comprendre que la particularité de ce style est de ne pas trier les entrées de la bibliographie : elles apparaissent dans l'ordre où elles sont citées dans le texte. Les labels sont numériques, comme pour le style plain.bst ;
- enfin, abbrv.bst a la particularité d'abréger les prénoms, et les définitions des mois et des journaux. En effet, je ne l'ai pas précisé, mais les quatre styles classiques contiennent les définitions des mois et de certains noms de journaux (pas au sens *Le Monde*, mais plutôt journaux scientifiques. Mais pas non plus *Science et Vie Junior...*) À part ça, le comportement de abbrv.bst est similaire à celui de plain.bst.

Voila donc pour les styles classiques. Ces styles ont de nombreux défauts, comme le fait qu'ils ne sont pas polyglottes, qu'ils n'ont pas de champ `url`, le tri se fait selon des critères discutables, ... Par ailleurs, il n'y a pas vraiment de règles typographiques très précises, toutes ces raisons font qu'il existe de nombreux autres styles. Nous allons en voir quelques-uns de ce pas.

7 D'autres styles bibliographiques

Il existe un nombre incalculable[15] de styles bibliographiques. Je n'ai pas l'intention de les décrire tous, pas plus que d'en décrire certains en détail. J'en donne juste quelques-uns, peut-être les plus utilisés.

7.1 Le style apalike.bst

La principale particularité du style bibliographique apalike.bst, écrit par Oren Patashnik également, est le calcul des « labels » : apalike.bst utilise un style dit *author-year*. Le plus simple, pour comprendre, c'est de regarder :

Dans (Goossens et al., 1998), il y a, à la page 384, un exemple complet du résultat que produit apalike.bst.

Références

Goossens, M., Mittelbach, F., and Samarin, A. (1998). *The LaTeX Companion.* Addison-Wesley.

Lamport, L. (1997). *LaTeX : A Document Preparation System.* Addison-Wesley.

Il est *très* important de ne pas oublier d'inclure le package apalike.sty lorsqu'on utilise le style bibliographique apalike.bst. En effet, si tu te souviens de comment marchent `\@biblabel` et `\cite`, tu comprendras qu'il a fallu les redéfinir. De plus, les labels peuvent être longs avec le style apalike.bst, et donc il a été prévu qu'elles puissent être coupées, ce qui n'était pas le cas avec la commande `\cite` de LaTeX (*cf.* section 3).

Pour terminer, je ne peux m'empêcher de citer quelques lignes du fichier apalike.bst lui-même :

```
% Many journals require a style like 'apalike', but I recommend that you
% not use it if you have a choice---use something like 'plain' instead.
```

Il existe des variantes du style apalike.bst, nommées authordate1.bst, authordate2.bst, authordate3.bst, authordate4.bst, qui sont tous de légères variations sur le style *author-year*, et qui doivent être utilisés avec le package authordate1-4.sty.

Une dernière remarque : apalike.sty redéfinit `\bibitem` de telle sorte que l'argument optionnel devient obligatoire (même s'il reste entre crochets).

7.2 Le package natbib.sty

Le package natbib.sty de Patrick W. Daly va un peu plus loin : son but principal est de redéfinir la commande `\cite`, afin de pouvoir faire des références de type *author-year* ou numériques, mais de manière plus flexible.

Les styles classiques[16] ont été adaptés pour natbib.sty. Ils s'appellent alors plainnat.bst, abbrvnat.bst et unsrtnat.bst. Un point important : ces trois styles ont un champ `url` qui permet de faire référence à des articles qui sont sur le web. Mais natbib.sty peut aussi être utilisé avec les styles apalike.bst ou authordate1.bst à authordate4.bst.

Enfin, natbib.sty possède une documentation [Dal99c] très bien faite, très complète, et je n'en dis pas plus ici pour t'inciter à aller la lire.

---

[15] Enfin, presque...

[16] Sauf alpha.bst puisque les labels sont de type *author-year*.

7.3 custom-bib

Face à autant de variations dans les styles bibliographiques, Patrick W. Daly s'est dit qu'il serait pas mal de faire un petit programme qui permettrait à chacun de créer son propre style bibliographique, suivant une vingtaine de critères.

Comme d'hab, quand un truc est fait par Daly, la documentation est super bien faite. Je ne vais donc pas m'attarder.

Pour créer son propre style bibliographique, donc, il suffit d'exécuter `latex makebst.tex`, puis de répondre aux questions. Au fil des réponses, le fichier de style bibliographique va être créé. C'est assez simple à faire, je te laisse essayer, c'est assez intuitif.

8 Problèmes et solutions

Cette section termine la partie « LaTeX » de cette doc, et je vais ensuite pouvoir attaquer la partie « BibTeX ». Ici, donc, en vrac, je vais évoquer certaines difficultés qui pourraient se présenter, et quelques packages qui permettent d'y remédier.

8.1 Comment avoir plusieurs bibliographies différentes ?

C'est un problème que je n'ai pas encore évoqué, mais, par défaut, les commandes `\bibliography` et `\bibliographystyle` doivent apparaître au plus une fois dans un fichier LaTeX. Si elles apparaissent plusieurs fois, LaTeX ne s'en plaindra pas, c'est BibTeX qui ne va pas savoir quel style et quelle base de données utiliser, puisqu'il y aura plusieurs `\bibdata` et plusieurs `\bibstyle` dans le fichier auxiliaire.

La solution, comme les plus rusés l'auront compris, serait de créer plusieurs fichiers auxiliaires, chacun contenant une fois la commande `\bibdata` et une fois la commande `\bibstyle`. C'est exactement ce que font les packages multibib.sty, chapterbib.sty et bibunit.sty.

Le package multibib.sty, de Thorsten Hansen, permet, au moment où on fait une référence, de dire dans quelle bibliographie doit aller cette référence. Pour ce faire, on utilise des commandes `\cite` différentes pour chaque bibliographie, et le tour est joué. Pour cette doc, par exemple, j'aurais pu demander à multibib.sty de définir les commandes `\citelatex` et `\citebibtex` pour faire deux bibliographies séparées, l'une au sujet de LaTeX et l'autre au sujet de BibTeX [17]. Bien entendu, quand on fait ça, il faut ensuite lancer BibTeX sur les différents fichier `.aux` créés. Pour le reste, la doc [Han00b] est bien faite (aussi), donc je te laisse t'informer par toi-même. La doc décrit elle-même le principal défaut de cette méthode : il est tout à fait possible qu'un même « label » apparaisse dans plusieurs bibliographies différentes, sans pour autant désigner le même document. Il n'y a pas grand'chose à faire contre ça...

Le package chapterbib.sty, écrit principalement par Donald Arseneau, permet de faire une bibliographie par fichier inclus. Souvent, dans un document assez long (s'il n'est pas long, c'est pas la peine de faire plusieurs bibliographies), on découpe le document en chapitres, et chaque chapitre est mis dans un fichier `.tex` différent, et le tout est inclus dans un fichier principal par la commande `\include`. chapterbib.sty s'arrange alors que soit associé à chaque fichier inclus, un fichier `.aux` qui contiendra les commande `\bibstyle`, `\bibdata` et `\citation`. Pour plus d'infos, le package est documenté dans le source, à la fin.

Le package bibunit.sty, que l'on doit également à Thorsten Hansen, est assez similaire à chapterbib.sty : il permet de faire une bibliographie par « unité », une unité étant une portion de document commençant par `\begin{bibunit}` et se terminant par `\end{bibunit}`. À l'intérieur d'une telle unité, toutes les références sont faites à la bibliographie qui est incluse dans cette même unité. Voir [Han00a] pour en savoir plus.

Il existe encore d'autres packages qui font ce genre de choses : camel.sty, bibtopic.sty. Je te laisse aller voir de quoi il retourne, puisqu'encore une fois, ce n'est pas vraiment le sujet de cette documentation.

[17] Pour ceux qui connaissent, le principe est le même que pour le package multind.sty, qui permet d'avoir des index séparés.

8.2 Et si je ne veux pas la bibliographie, juste les références ?

Il existe principalement deux solutions à ce problème : la première consiste à mettre les références en note de bas de page. C'est pratique pour le lecteur, à chaque fois qu'il rencontre une citation, il sait de quoi ça parle sans devoir tourner les pages. Ça peut être pratique, surtout pour les gros bouquins où il y a plein de références importantes. Pour faire ça, donc, c'est le package footbib.sty. Lui aussi a une doc assez explicite [Dom97].

Parfois enfin, certains mettent des références dans le texte, au milieu de tout. Plutôt que de faire référence à [GMS98], on fait alors référence à Michel Goossens, Franck Mittelbach et Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1998. Ça donne un style... Si toi aussi, tu veux faire ça, je te laisse aller lire la doc [Dal99a] du package bibentry.sty.

Ces deux packages peuvent poser des problèmes s'ils sont utilisés conjointement à d'autres, comme par exemple hyperref.sty. Simplement parce que chacun veut imposer sa propre définition de l'environnement `thebibliography`, de `\bibitem`, et celui qui perd finit souvent par se plaindre. Il n'y a pas de recette miracle, c'est au cas par cas...

8.3 Et si je ne veux pas les références, juste la bibliographie ?

C'est possible aussi, grace à la commande `\nocite`. Cette commande permet de dire à BibTeX d'inclure une entrée, sans que la référence à cette entrée apparaisse dans le document. Ça s'utilise exactement de la même façon que `\cite`.

Une variante qui peut être intéressante, c'est `\nocite{*}` : ça permet, d'un seul coup, de dire à BibTeX d'inclure toutes les entrées des fichiers `.bib` inclus dans la bibliographie. Ces entrées seront ajoutées dans leur ordre d'apparition dans BibTeX, sauf évidemment si elles ont déjà été `\cite`-ées auparavant.

Voilà, j'en ai terminé avec ce qui ne m'intéressait, au départ, pas trop. Je vais maintenant entrer dans le vif du sujet, en comançant par expliquer comment créer un fichier `.bib` (à peu près) proprement.

Troisième partie
Le fichier .bib

Le fichier .bib, c'est la base de données. Son contenu dépend en grande partie du style qui va lui être appliqué, et de ce qu'on veut en faire. Ce que je vais décrire dans cette partie, c'est dans le cas d'une bibliographie, avec un style pas trop excentrique[18]. Mais (je l'ai déjà dit, mais j'insiste) BibTEX peut faire beaucoup d'autres choses, comme nous le verrons un peu dans la partie 5.

9 La structure du fichier .bib

Voici un exemple d'entrée que peut contenir un fichier .bib :

```
@book{companion,
  author        = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander",
  title         = "The {{\LaTeX}} {C}ompanion",
  publisher     = "Addison-Wesley",
  year          = 1998
}
```

Pour le moment, on ne va pas s'intéresser au contenu, aux accolades dans le titre, tout ça, on verra plus tard. Pour le moment, on regarde juste la forme. Globalement, on a ça :

```
@type_d_entrée{clef_interne,
  champ_1       = "valeur_1",
  champ_2       = "valeur_2",
  ...
  champ_n       = "valeur_n"
}
```

Quelques remarques :
- une nouvelle entrée doit toujours commencer par un @. Cela donne un moyen assez simple de « commenter » une entrée : il suffit de supprimer son @ initial. Tout ce qui n'est pas à l'intérieur de la définition d'une entrée est considéré comme étant du commentaire, et BibTEX l'ignore. Il ne faut pas hésiter, comme à chaque fois qu'on peut mettre des commentaires, à bien documenter la base de données, à bien la structurer. Inversement, comme BibTEX ne peut pas savoir à l'avance si un nom est bien le nom d'un type d'entrée, tout ce qui commencera par un @ sera considéré comme le début d'une nouvelle entrée[19]
- BibTEX mélange les majuscules et les minuscules. Ainsi, cette distinction importe peu pour les noms des types d'entrée et des champs. Pour la clef interne, BibTEX va se plaindre s'il y a deux entrées dont les clefs internes sont identiques à la casse près. Ainsi, si on définit une entrée de clef `Exemple`, on ne peut pas en définir une de clef `exemple`. Par contre, pour y faire référence dans LATEX, il faut bien mettre `\cite{Exemple}`, et non `\cite{exemple}`, puisque LATEX fait la différence entre majuscules et minuscules, lui...
- les espaces et sauts de ligne ne sont pas importants. Juste pour l'esthétique et la lisibilité. Les virgules sont par contre obligatoires ;

[18] Donc, en utilisant les champs et les entrées décrits aux pages 10 et suivantes.

[19] Il existe un type d'entrée spécial, **@comment**, dont le contenu n'est pas interprété non plus par BibTEX. Au vu de ce qui précède, la seule utilisation que j'en vois est de supprimer facilement un grand nombre d'entrées, de manière temporaire.

– les valeurs (*i.e.* les parties droites des assignations) peuvent être mises entre accolades, à la place des guillemets. La seule différence est que, entre deux accolades, on peut mettre des guillemets, alors qu'on ne peut pas en mettre directement entre des guillemets. Par exemple, pour citer l'article intitulé *Comments on "Filenames and Fonts"* de Franck MITTELBACH, on pourra utiliser une des possibilités suivantes :

```
title         = "Comments on {"}Filenames and Fonts{"}",
title         = {Comments on "Filenames and Fonts"},
```

Les accolades doivent se correspondre, puisqu'elles vont former des groupes pour LaTeX. Évidemment, pour qu'une accolade apparaisse dans le texte, il faut la coder avec `\{` (ou `\}`), et on ignorera le message d'avertissement de BibTeX si une accolade n'est pas fermée (ou pas ouverte).
– Lorsqu'une valeur est numérique, les guillemets et accolades peuvent être omis (mais ils peuvent également être mis...)

Je crois que je l'ai déjà dit, mais on peut définir des champs même s'ils ne sont pas utilisés par le style bibliographique que l'on va appliquer. Ainsi, par exemple, on pourra définir l'entrée pour le LaTeX companion de la façon suivante :

```
@book{companion,
  author        = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander",
  title         = "The {{\LaTeX}} {C}ompanion",
  booktitle     = "The {{\LaTeX}} {C}ompanion",
  publisher     = "Addison-Wesley",
  year          = 1998,
  month         = "Septembre",
  ISBN          = "0-201-54199-8",
  bibliotheque  = "Oui",
}
```

si on veut apporter des précisions supplémentaires[20], ou se souvenir que le livre en question est disponible à la bibliothèque. Il ne faut donc pas hésiter à dire tout ce que l'on sait, en donnant des noms de champs explicites pour être sûr qu'ils ne seront pas utilisés, par erreur, par d'autres styles bibliographiques, même exotiques[21].

10 Les entrées @string et @preamble

Ce ne sont pas à proprement parler des entrées, mais ça y ressemble. L'entrée **@string** permet de définir des abréviations. Par exemple, depuis le début, on cite deux livres publiés par Addison-Wesley. Il pourrait être utile de définir un raccourci pour cette maison d'édition. On modifie donc notre fichier `.bib` de la façon suivante :

```
@string{AW      = "Addison-Wesley"}

@book{companion,
  author        = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander",
  title         = "The {{\LaTeX}} {C}ompanion",
  booktitle     = "The {{\LaTeX}} {C}ompanion",
  publisher     = AW,
  year          = 1998,
  month         = "Septembre",
  ISBN          = "0-201-54199-8",
```

[20]Le fait de renseigner le champ `booktitle`, qui n'est pas utilisé pour les entrées de type **@book**, peut avoir un réel intérêt lorsqu'on utilise les références croisées, voir la section 13 pour plus de détails.

[21]« À quoi bon ? », me diras-tu. D'une part, ça coûte pas bien cher de le faire, et d'autre part, il se peut que tu aies besoin de ces infos plus tard. Il te suffira alors d'adapter ou de créer un style bibliographique, sans avoir à aller ajouter toutes les infos pour chaque entrée.

```
  bibliotheque  = "Oui",
}
```

Cela permet non seulement de gagner un peu de temps, mais surtout d'homogénéiser la base de données, notamment ici, par exemple, de mettre le tiret partout ou nulle part, mais pas juste une fois de temps en temps... C'est aussi pratique, je trouve, pour les noms d'auteurs, ça évite, en plus, de faire des fautes dans les noms, ce qui ne fait jamais très sérieux.

Concernant **@preamble**, ça permet d'insérer des commandes (ou du texte) dans le fichier créé par BibTEX. En fait, tout ce qui se trouve dans les commandes **@preamble** est concaténé et placé dans une variable nommée `preamble$` dans le fichier `.bst`. Dans les styles classiques, le contenu de cette variable est placé juste avant le début de l'environnement `thebibliography`, et permet de (re)définir éventuellement certaines commandes. Un petit exemple, et on n'en parle plus (enfin, si, on s'en reservira plus loin) :

```
@preamble{ "\def\gobble#1{} "}
```

Dans cet exemple, on définit une fonction qui prend un argument et ne fait rien. Contrairement aux apparances, ça peut servir. Et de manière générale, donc, si tu as besoin d'une commande LATEX particulière dans un champ de la base de données, il faut définir cette commande dans un **@preamble**. Une dernière remarque : il ne faut jamais mettre de commandes de mise en page, ou du même genre, dans un **@preamble**. La mise en page n'est pas le rôle du fichier `.bib`.

11 Le titre

Voyons maintenant comment remplir le champ `title` d'une référence bibliographique. Regardons tout d'abord ce que j'avais écrit pour le *LATEX Companion* :

```
  title         = "The {{\LaTeX}} {C}ompanion"
```

Avant d'aller plus loin, quelques définitions seront nécessaires : la *profondeur d'accolades* d'un élément est le nombre d'accolade qui l'entourent. Enfin, c'est pas très formel, comme définition, mais par exemple, dans le titre ci-dessus, `\LaTeX` est à une profondeur 2, le `C` est à profondeur 1, et le reste à une profondeur 0[22]. On définit un *caractère spécial* comme étant un élément commençant par une accolade située à profondeur 0, immédiatement suivie d'une contre-oblique, et se terminant par l'accolade qui correspond à la première. Je sais pas si j'ai été très clair. Par exemple, dans le titre ci-dessus, il n'y a pas de caractère spécial, puisque `{\LaTeX}` n'est pas situé à profondeur 0. Noter que les lettres qui apparaissent dans un caractère spécial sont considérées comme étant à une profondeur d'accolades nulle. Voilà pour les définitions.

De manière générale, le titre peut être traité de plusieurs manières différentes par le style bibliographique :

– tout d'abord, il sert pour le tri. Pour trier, BibTEX calcule une « valeur » pour chaque entrée, nommée `sort.key$`, qui est une longue (en général) chaîne de caractères, puis trie les entrées dans l'ordre alphabétique de `cite.key$`. Dans `cite.key$`, seuls des caractères alphanumériques peuvent intervenir. Lors du calcule de `cite.key$`, donc, on se sert des noms des auteurs, de l'année, du titre, et on doit supprimer les caractères non alphanumériques de ces éléments. Une fonction, nommée `purify$`, permet de faire cela. Cette fonction retire tout ce qui n'est pas alphanumérique[23], excepté dans les caractères spéciaux. Pour les caractères spéciaux, globalement, elle retire tous les espaces et toutes les commandes LATEX du caractère spécial, même ceux qui sont contenu dans des accolades à l'intérieur du caractères spécial. Elle laisse tout le reste, par contre. Par exemple, `t\^ete`, `t{\^e}te` et `t{\^{e}}te` deviendront `tete`, `Bib{\TeX}` deviendra `Bib` et `Bib\TeX` deviendra `bibTeX`. Quelques commandes, au nombre de 13, font exception à la

[22]Évidemment, si le titre avait été codé avec `{The {{\LaTeX}} {C}ompanion}`, c'aurait été pareil, puisque les accolades les plus à l'extérieur sont équivalentes à des guillemets.

[23]Sauf les tirets, qu'elle remplace par des espaces.

règle : `\i`, `\j`, `\oe`, `\OE`, `\ae`, `\AE`, `\aa`, `\AA`, `\o`, `\O`, `\l`, `\L`, `\ss`, qui sont des commandes LaTeX correspondant respectivement à ı, ȷ, œ, Œ, æ, Æ, å, Å, ø, Ø, ł, Ł, ß. Ces fonctions sont remplacées, respectivement, par `i`, `j`, `oe`, `OE`, `ae`, `AE`, `aa`, `AA`, `o`, `O`, `l`, `L`, `ss`. Toute autre commande est irrémédiablement supprimée ;
– une deuxième chose qui peut arriver à un titre, c'est d'être mis en minuscules (excepté la première lettre). La fonction qui change la casse des lettres s'appelle `change.case$`, et n'agit que sur les lettres qui sont à une profondeur d'accolade 0, sauf dans un caractère spécial. Dans un caractère spécial, les commandes sont laissées telles qu'elles, le reste est mis en minuscules.

Ça, c'était la théorie. Essayons maintenant la mise en pratique. Essayons pour commencer de définir le champ `title` du *LaTeX Companion*. Plusieurs possibilités viennent à l'esprit :

– `title = "The \LaTeX Companion"` : cette solution ne convient pas, parce qu'en passant en minuscules, on obtiendrait `The \latex companion`, ce que LaTeX va apprécier modérément...
– `title = "The {\LaTeX} {C}ompanion"` : cela résout le problème évoqué ci-dessus. Cependant, si on applique la fonction `purify$` à ce titre, on obtient `The  Companion`. Le tri risque donc d'être erroné ;
– `title = "The {{\LaTeX}} {C}ompanion"` : c'est la solution que j'ai utilisée.
– `title = "The { \LaTeX} {C}ompanion"` : le problème, c'est que cet espace n'est pas supprimé, et donc l'application de `purify$` va donner `The  LaTeX Companion`, avec deux espaces entre `The` et `LaTeX`. Ça peut donc aussi poser problème lors du tri ;
– `title = "The{ \LaTeX} {C}ompanion"` : c'est une solution qui marche également. Je la trouve cependant un peu moins élégante.
– `title = "The {\csname LaTeX\endcsname} {C}ompanion"` : d'une part, c'est un peu violent, et d'autre part, ca ne marche pas lorsque ce titre sera passé en minuscules : les commandes sont alors inchangées, mais le texte, lui, est passé en minuscule. On va se retrouver avec `\latex`, qui va immanquablement provoquer une erreur.

Pour coder un accent dans un titre, par exemple le *É* (en majuscule) de *École*, on mettra `{\'E}cole` ou `{{\'E}}cole`, suivant que l'on veut que le *E* passe en minuscule lors du changement de casse ou pas. Noter que le deuxième cas n'est pas un caractère spécial. Cela n'a pas d'importance ici, mais si, pour une raison ou pour une autre, on veut compter le nombre de *caractères* qui apparaissent dans le titre, ou en extraire le premier caractère, ce `{{\'E}}` comptera pour 7 caractères, alors que `{\'E}` compte pour 1. Il faut noter, par contre, qu'a l'intérieur d'un caractère spécial, tout est considéré comme étant a profondeur d'accolade nulle, même quand il y a beaucoup d'accolades.

Voilà donc pour les subtilités liées au titre. Passons maintenant, et c'est une autre paire de manches, aux noms d'auteurs...

12 Le champ `author`

Partons, encore une fois, de ce que j'ai écrit pour le *LaTeX Companion* :

```
author  = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander"
```

La première chose à constater, c'est que deux auteurs sont séparés par le mot-clef `and`. Ensuite, pour un auteur, on met d'abord son nom, une virgule, et son prénom. Enfin, ça, c'est dans le cas simple...

Avant d'aller plus loin, remarquons un point crucial : BibTeX doit pouvoir distinguer le nom du prénom. Pour le tri, tout d'abord, et pour pouvoir, comme le font certains styles, abréger le prénom. Un autre point crucial est que certaines personnes ont plusieurs prénoms, ou des prénoms composés, d'autres ont plusieurs noms, d'autres ont une particule, et d'autres ont une « extension » (comme le *Fils* dans *Alexandre Dumas, Fils*). Dans l'explication un peu technique qui suit, le prénom sera appelé `First`, la particule sera notée `von`, le nom `Last` et l'extension, `Jr`[24].

Il faut donc que BibTeX puisse distinguer les différentes partie d'un nom. Dans ce but, BibTeX reconnaît trois formats pour un nom :

– `First von Last` ;

[24] Ce sont les appellations proposées par Patashnik dans sa documentation [Pat88b].

– `von Last, First`;
– `von Last, Jr, First`.

Il lui suffit de compter le nombre de virgules pour savoir dans quelle situation il se trouve.
Voici les caractéristiques de ces différentes possibilités :

– `First von Last` : imaginons qu'on ait entré `Jean de La Fontaine`. La première chose est que le `Last` ne peut pas être vide. Donc le dernier élément sera toujours considéré comme faisant partie de `Last`. Ensuite, BibTEX regarde la première lettre de chaque élément[25].
  – S'il y a des éléments commençant par une minuscule, excepté le dernier qui est déjà dans le `Last`, ils iront dans le `von`. Si, entre deux éléments commençant par une minuscule (sauf le dernier, toujours), il y en a un qui commence par une majuscule, il va dans le `von` quand même. Tout ce qui se trouve à gauche va alors dans le `First`, tout ce qui est à droite va dans le `Last`.
  – S'il n'y a aucun élément commençant par une minuscule, alors seul le dernier élément va dans le `Last`, tout le reste va dans le `First`.

  Avec `Jean de La Fontaine`, donc, le `First` est `Jean`, le `Last` est `La Fontaine`. Voici quelques variations sur le sujet, tu peux vérifier si t'as bien compris comment ça marche...

entrée	`First`	`von`	`Last`
`jean de la fontaine`		`jean de la`	`fontaine`
`Jean de la fontaine`	`Jean`	`de la`	`fontaine`
`Jean {de} la fontaine`	`Jean de`	`la`	`fontaine`
`jean {de} {la} fontaine`		`jean`	`de la fontaine`
`Jean {de} {la} fontaine`	`Jean de la`		`fontaine`
`Jean De La Fontaine`	`Jean De La`		`Fontaine`
`jean De la Fontaine`		`jean De la`	`fontaine`
`Jean de La Fontaine`	`Jean`	`de`	`La Fontaine`

  Cette dernière ligne est bien ce que l'on veut : le nom de cet auteur est bien *La Fontaine*, et son prénom *Jean*. Évidemment, y en a qui voudront que le `von` ait une majuscule. On verra ça plus tard, à la section 14

– `von Last, First` : l'idée est un peu la même, sauf que le `First` est plus facile à identifier ici, c'est tout ce qui est à droite de la virgule. Pour ce qui est à gauche, la règle est assez semblable au cas précédent : si cette partie gauche est non vide, son dernier élément va nécessairement dans le `Last`. Pour le reste, c'est en fonction des majuscules et des minuscules, sachant que s'il y a un élément commençant par une minuscule, si ce n'est pas le dernier, alors tout ce qui est à sa gauche est dans le `von`. Bon, là encore, le mieux est sûrement de faire quelques exemples :

entrée	`First`	`von`	`Last`
`jean de la fontaine,`[26]		`jean de la`	`fontaine`
`de la fontaine, Jean`	`Jean`	`de la`	`fontaine`

[25] Dans toute la suite, quand je dis « première lettre » d'un élément, ça veut dire « la première lettre qui est à une profondeur d'accolades nulle, s'il y en a, les lettres (pas les commandes) d'un caractère spécial étant considérées comme à profondeur nulle, même si elles sont entourées par quinze accolades. » Lorsqu'il n'y a pas de lettres à la profondeur zéro, alors tout l'élément va faire comme ses voisins, avec une priorité pour le `First`, puis pour le `Last`, en cas d'ambiguïté. Il ne sera donc dans le `von` que s'il est entouré par deux `von`. Enfin, tous les éléments dans un même groupe d'accolades iront au même endroit, même pour un caractère spécial. Enfin, mais là, vraiment, j'arrête après, pour une commande LaTEX en dehors d'un caractère spécial, BibTEX vire la contre-oblique et considère le mot qui reste. Si t'as pas tout compris, c'est un peu gênant, je te conseille d'aller prendre deux aspirines et de relire cette note.

entrée	First	von	Last
De La Fontaine, Jean	Jean		De La Fontaine
De la Fontaine, Jean	Jean	De la	fontaine
de La Fontaine, Jean	Jean	de	La Fontaine

– `von Last, Jr, First` : bah c'est exactement pareil, sauf qu'il y a une deuxième virgule, et que ce qui est entre les deux virgules va aller dans le `Jr`. Pas d'exemple dans ce cas-là, donc, c'est facile.

La séparation entre deux noms n'est pas nécessairement un espace : il arrive en effet que deux prénoms soient séparés par un tiret, comme dans « Jean-François » par exemple. BibTEX reconnaît alors bien la séparation entre les deux prénoms, et ainsi, par exemple, la version abrégée de ce prénom est « J.-F. ». Un tilde est également reconnu comme séparateur. Il résulte de tout cela que si on écrit `Jean-baptiste Poquelin`, la partie `baptiste`, qui commence malencontreusement par une minuscule, sera placée dans le `von`...

Il me paraît opportun de revenir ici sur les abréviations : comme on l'a vu, c'est pas forcément évident de mettre les noms dans le bon format, sans parler des risques de fautes de frappe. Je conseille donc, pour chaque auteur, de définir une abréviation. Tout ce qu'il est nécessaire de savoir, c'est que pour concaténer deux chaînes de caractères, on utilise le `#`. Ainsi, par exemple, j'ai un fichier `auteurs.bib` qui contient (entre autres) ceci :

```
@string{goossens        = "Goossens, Michel"}
@string{mittelbach      = "Mittelbach, Franck"}
@string{samarin         = "Samarin, Alexander"}
```

un autre fichier `editeurs.bib` contenant entre autres

```
@string{AW              = "Addison-Wesley"}
```

et mon fichier contenant les références bibliographiques à proprement parler contient[27]

```
@book{companion,
  author        = goossens #" and "# mittelbach #" and "# samarin,
  title         = "The {{\LaTeX}} {C}ompanion",
  booktitle     = "The {{\LaTeX}} {C}ompanion",
  year          = 1998,
  publisher     = AW,
  month         = "Septembre",
  ISBN          = "0-201-54199-8",
  bibliotheque  = "Oui",
}
```

Et ça va beauuuuuuuuuuucoup plus vite à taper, quand on s'y est habitué.

13 Les références croisées (`crossref`)

Comme je l'ai déjà dit plus haut (je sais plus où... Ah si, à la page 11), BibTEX permet de faire des références croisées. C'est aussi extrêmement pratique si on veut citer une partie d'un livre, un article issu des actes d'une conférence. Par exemple, je peux définir

[26]Dans ce cas, BibTEX renvoie un message d'erreur, disant qu'il a trouvé un nom se terminant par une virgule. Bah oui, parce que souvent, les gens qui connaissent pas, ils séparent les noms par des virgules.

[27]En fait, pas exactement, parce que le `"Septembre"` pose encore des problèmes d'internationalisation.

```
@incollection{companion-bib,
  crossref      = "companion",
  title         = "Bibliography Generation",
  chapter       = 13,
  pages         = "371-420",
}
```

C'est ici qu'on se rend compte de l'intérêt d'avoir renseigné le champ `booktitle` de l'entrée `companion` : ce champ ne sert pas pour l'entrée **@book**, mais il est quand même récupéré ici, et il est obligatoire pour une entrée de type **@incollection**. On aurait bien sûr pû l'ajouter dans l'entrée ci-dessus, mais on devrait alors l'ajouter autant de fois qu'on cite de chapitres du livre en question.

Par ailleurs, un autre intérêt de ce mécanisme est que, dans le cas où on cite plusieurs chapitres du livre sans citer le livre, BibTeX peut « factoriser », c'est à dire mettre une référence (avec la commande `\cite`) dans les entrées concernées, et ajouter le livre. Par contre, si on ne cite qu'un chapitre du livre, BibTeX n'ajoute pas le livre. Ainsi, on peut obtenir les deux présentations suivantes, suivant les goûts :

Références

[1] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Bibliography generation. In *The LaTeX Companion* [4], chapter 13, pages 371–420.

[2] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Higher mathematics. In *The LaTeX Companion* [4], chapter 8, pages 215–258.

[3] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Index generation. In *The LaTeX Companion* [4], chapter 12, pages 345–370.

[4] Michel Goossens, Franck Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Septembre 1998.

ou

Références

[1] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Bibliography generation. In *The LaTeX Companion*, chapter 13, pages 371–420. Addison-Wesley, Septembre 1998.

[2] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Higher mathematics. In *The LaTeX Companion*, chapter 8, pages 215–258. Addison-Wesley, Septembre 1998.

[3] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Index generation. In *The LaTeX Companion*, chapter 12, pages 345–370. Addison-Wesley, Septembre 1998.

On peut préciser à BibTeX la limite à partir de laquelle il doit inclure le livre : il faut pour cela utiliser l'argument `-min-crossrefs` de BibTeX. Par exemple, pour le premier exemple, j'ai tapé simplement `bibtex biblio`, alors que pour le deuxième, j'ai tapé `bibtex -min-crossrefs=5 biblio`.

Encore une remarque importante : lorsqu'on fait une référence croisée, il faut impérativement que l'entrée à laquelle on fait référence soit située *après* l'entrée qui contient le `crossref`. De plus, on ne peut pas « emboîter » deux `crossref`, c'est à dire qu'on ne peut pas faire référence à une entrée qui contient déjà un champ `crossref`.

Dernière chose, le champ `crossref` est un peu particulier, c'est un champ qui existe toujours, quel que soit le style bibliographique utilisé. Si l'entrée à laquelle on fait référence est incluse dans la bibliographie (soit par une commande `\cite` dans le document, soit par BibTeX s'il a « factorisé » cette entrée), alors les entrées qui y font référence ont un champ `crossref`. Par contre, si elle n'est pas incluse, le champ `crossref` est « supprimé » de l'entrée.

14 Petits trucs

14.1 Comment faire pour que *Philippe* soit abrégé en *Ph.* ?

La fonction qui abrège les prénoms est en fait la même que celle qui extrait les parties `First`, `von`, `Last` et `Jr` des noms. Lorsqu'on lui demande d'abréger un prénom, il va garder uniquement le premier caractère (qui peut être un « caractère spécial »). Donc, bien entendu, *Philippe* sera abrégé en *P.*, ce que certains n'aiment pas trop. Pour contourner ce problème, on va utiliser un caractère spécial : on écrira donc `"{\relax Ph}ilippe"`. Ainsi, la version abrégée sera `{\relax Ph}.`, ce qui donnera, en LaTeX, *Ph.* La version non abrégée sera `{\relax Ph}ilippe`, qui donne bien Philippe.

14.2 Comment mettre des majuscules dans le `von` ?

Il peut arriver que le `von` doive commencer par une majuscule. Pour une raison que j'ignore, l'exemple classique est *Maria De La Cruz*.

Une première solution consiste à entrer `"{\uppercase{d}e La} Cruz, Maria"`. En effet, pour distinguer les différentes parties de ce nom, BibTeX va commencer par placer le `Cruz` dans le `Last`, et le `Maria` dans le `First`. En ce qui concerne le `{\uppercase{d}e La}`, c'est un caractère spécial qui commence par un `d` minuscule, il va donc dans le `von`.

Cette solution a cependant le désavantage suivant : certains styles, pour créer le label qui sera affiché dans le document pour faire référence à une entrée, prennent les premières lettres de tous les éléments du `von` et du `Last`. Ainsi, tel qu'on l'a entré ci-dessus, ce nom d'auteur donnera le label `{\uppercase{d}e La}C`, donc `De LaC`, alors qu'on voudrait `DLC`. La première idée qui vient à l'esprit serait de mettre

```
author = {\uppercase{d}}e {\uppercase{l}}a Cruz, Maria.
```

C'est effectivement une solution correcte, mais qui a l'inconvénient de donner le label `dlC` alors qu'on aimerait mieux encore `DLC`.

La solution sera alors de mettre les lettres entre accolades. En effet, elles ne seront alors pas prises en compte, et c'est la deuxième lettre, en minuscules, qui déterminera :

```
author = {D}e {L}a Cruz, Maria.
```

Cette solution résout tous les problèmes évoqués ci-dessus.

14.3 Comment mettre des minuscules dans le `Last` ?

C'est un peu le problème inverse, et on pourrait penser que la solution est similaire. Prenons pour exemple le célèbre ingénieur espagnol *Juan de la Cierva y Codorníu*. Si, si, il existe. La première idée serait

```
author = de la Cierva {\lowercase{Y}} Codorn{\'\i}u, Juan
```

ou

```
author = de la Cierva {y} Codorn{\'\i}u, Juan
```

Le défaut de ces solutions, c'est que, avec des styles comme `alpha.bst`, le « label » qui sera calculé contiendra `CyC` ou `CYC`, alors qu'on préfèrerait `CC`.

La solution ultime me semble être la suivante :

```
author = de la Cierva{ }y Cordon{\'\i}u, Juan
```

ou aussi

```
author = de la {Cierva y} Cordon{\'\i}u, Juan
```

Ces deux possiblités font ce que l'on veut : dans le premier cas, BibTEX ne voit même pas le `y`, qu'il considère comme « attaché » au `Cierva`. Dans le deuxième cas, `Cierva y` est un nom complètement entre accolades, il va dans la `Last`, qui est prioritaire sur le `von`.

Il faut noter que, dans le cas de styles bibliographiques qui appliquent des modifications au nom (mise en petites capitales ou en majuscules) mais pas à la particule, la solution ci-dessus n'est pas parfaite, puisque la deuxième particule sera traitée différemment de la première. C'est inévitable, puisque BibTEX ne peux pas traiter plusieurs particules dans un même nom. Une solution serait de remplacer la deuxième particule par une commande imposant une particule en minuscules. Mais inversement, si la particule doit être mise en capitales, cela posera problème... Ce cas ne peut, à ma connaissance, pas être résolu indépendamment du style bibliographique utilisé.

14.4 Et quand le `von` doit être collé au `Last`...

Le premier exemple qui me vient est *Jean d'Ormesson*. La meilleure solution semble être la suivante :

```
author = "d'\relax Ormesson, Jean"
```

La commande `\relax` ne fait rien. Sauf que, comme toute commande LATEX qui se respecte, elle mange les espaces qui la suivent, ce qui est l'effet voulu.

15 Le champ `key`

Il arrive que certains documents n'aient pas d'auteurs. Dans ce cas, les styles bibliographiques qui doivent calculer un « label » sont un peu perdus, ils ne savent pas à partir de quoi calculer ce label. Ils utilisent alors le champ `key` (seulement ses trois premières lettres, pour les styles classiques, mais tout le champ `key` pour `apalike.bst`, par exemple) pour calculer ces labels. Il faut donc ne pas oublier de renseigner ce champ le cas échéant.

Voila, je pense que j'ai dit tout ce que j'avais à dire. Si personne n'a de question, on va passer à la partie plus intéressante, je trouve, de tout ça : comment créer ou modifier un style bibliographique...

Quatrième partie
Le fichier `.bst`

16 Pourquoi et comment ?

Le fichier `.bst` est responsable de la mise en page des entrées qui sont citées dans le document. Il est primordial de savoir ce qui est du ressort du style bibliographique, et de le distinguer de ce qui est fait par BibTEX ou de ce qui doit être fait dans le fichier `.bib`.

En gros, BibTEX lit trois fichiers : le premier est le fichier `.aux` qui lui est passé en argument. C'est dans ce fichier qu'il trouve le nom du style à utiliser le nom des bases de données, et la liste des entrées à extraire de ces bases de données. Il va ensuite lire le fichier `.bst`, et le fichier `.bib`.

Le fichier `.bst` indique à BibTEX ce qu'il doit faire avec les données qui se trouvent dans le fichier `.bib`. Il va donc dire à BibTEX

– quelles sont les types d'entrées qui sont acceptés ;
– quels sont les champs qui seront utilisés ;
– enfin, globalement, que faire avec tous ces champs, suivant les types d'entrées.

Pour ce faire, le langage utilisé contient deux types d'instructions. Les premières seront appelées *commandes*, les autres seront appelées *fonctions internes*. Je vais parler des commandes dans la prochaine section, et des fonctions internes dans les suivantes.

17 Structure globale d'un fichier `.bst`

Voici, grossièrement, la structure d'un fichier `.bst` :

```
ENTRY
  {  ...  }
  {  ...  }
  {  ...  }

INTEGERS {  ...  }
STRINGS {  ...  }

MACRO {  ...  }{  ...  }
FUNCTION {  ...  }{  ...  }

READ
EXECUTE {  ...  }
ITERATE {  ...  }
SORT
ITERATE {  ...  }
REVERSE {  ...  }
EXECUTE {  ...  }
```

On voit apparaître, dans cette structure, les dix commandes que comprend BibTEX. En voici une succinte description :

`ENTRY` : c'est la commande qui définit les champs qui seront utilisés par ce style. Et même un peu plus. Cette commande prend en fait trois arguments : le premier est la liste de l'ensemble des champs qui seront utilisés par ce style[28]. Une liste, c'est une suite de noms alphabétiques séparés par des espaces et/ou sauts de ligne. Une chose que je n'ai pas précisé : BibTEX, globalement, ne fait pas attention aux minuscules et aux majuscules, pour lui, c'est tout pareil. Généralement, pour s'y retrouver, les noms des commandes sont en majuscules, le reste en minuscules.

Le deuxième argument est la liste des variables *entières* que l'on va utiliser pour chaque entrée, et le troisième, la liste des variables de type *chaîne de caractère*. Par exemple, certains styles bibliographiques seront chargés de calculer, pour chaque entrée, le label qui lui sera assigné, et tous devront calculer la longueur maximale des labels de toutes les entrées citées. Il faudra donc définir une chaîne de caractères qui contiendra ce label. Par exemple, le style plain.bst contient :

```
ENTRY
  { address
    author
    booktitle
    ...
    volume
    year
  }
  {}
  { label }
```

Il définit ainsi tous les champs classiques, et une chaîne de caractères qui contiendra le label qui sera calculé par la suite. Ce label est toujours considéré comme une chaîne de caractères, puisqu'on va, par la suite, en calculer la longueur. Dernière remarque : cette commande `ENTRY` doit apparaître une et une seule fois dans le fichier `.bst`.

`INTEGERS` : cette commande permet de définir des variables entières[29]. L'argument est encore une liste de noms du même format que pour `ENTRY`.

`STRINGS` : cette commande permet de définir des variables de type chaîne de caractères, de la même façon que `INTEGERS` définit des variables entières. Il est important de ne pas gâcher ces variables, BibTEX n'autorise pas d'en utiliser plus de vingt. Les limitations sont beaucoup moins sévères pour les autres variables, même celles déclarées dans `ENTRY`.

`MACRO` : cette commande permet de définir une abréviation, de la même façon que **@string** [30] en définit. Si une abréviation est définie à la fois dans le fichier `.bst` et dans le fichier `.bib`, c'est la définition du fichier `.bib` qui prime.

Le premier argument est l'abréviation, le deuxième donne sa signification, qui doit nécessairement être une chaîne de caractères, et à ce titre, être entourée de guillemets. Par exemple, dans plain.bst, on trouve :

```
MACRO {jan} {"January"}
MACRO {feb} {"February"}
MACRO {mar} {"March"}
...
```

[28] Le champ `crossref`, comme je l'ai déjà dit, est défini par défaut, il ne faut donc pas le redéfinir.

[29] La différence avec le deuxième argument de `ENTRY`, c'est que là, la variable est « globale », alors que les variables entières déclarées dans `ENTRY` sont associées aux entrées (par exemple, la variable `label` est associées à l'entrée courante). C'est peut-être un peu pas très clair, mais ça ira mieux après.

[30] À ne pas confondre avec `STRINGS`...

`FUNCTION` : c'est, de loin, la commande la plus utilisée. Elle permet, comme son nom peut le laisser penser, de définir des fonctions très générales, qui seront utilisées par la suite. Le premier argument est le nom de la fonction, et le deuxième, sa définition. Je ne donne pas d'exemple ici, on en verra bien assez par la suite, t'as qu'à tourner quelques pages...

`READ` : cette commande, tout comme `ENTRY`, doit apparaître une et une seule fois dans le fichier `.bst`. Enfin, il faut dire que c'est elle qui dit à BibTEX d'aller lire les bases de données, donc si on ne la met pas, on va pas aller bien loin, et la mettre deux fois, je vois pas l'intérêt. Comme je viens de le dire, c'est donc cette commande qui va chercher dans la base de données les entrées qui doivent être citées, c'est à dire celle qui sont listées dans le fichier `.aux`. Une autre remarque : la commandes `ENTRY`, ainsi que les éventuelles commandes `MACRO`, doivent impérativement être placées *avant* la commande `READ`, et les commandes `ITERATE` et `REVERSE` doivent être placées *après*.

`EXECUTE` : c'est la commande qui permet d'exécuter une fonction. Ladite fonction, passée en argument, aura impérativement été définie *avant* d'être exécutée. Pour le reste, rien de particulier. Enfin, si, mais j'en parle ci-dessous, dans la description de `ITERATE`. Juste une chose : quand je dis que la fonction est passée en argument, soit c'est le nom d'une fonction qui a été définie, soit c'est une suite de fonctions internes de BibTEX.

`ITERATE` : cette commande permet elle aussi d'exécuter une fonction. La différence, c'est que la fonction exécutée par l'intermédiaire d'`ITERATE` est exécutée sur toutes les entrées que BibTEX doit prendre en compte (toutes celles qui ont été lues auparavant par la commande `READ`), et tous les champs de chaque entrée sont accessibles pour cette fonction. La commande `EXECUTE`, quant à elle, n'a accès à aucun champ d'aucun entrée, et n'exécute la fonction passée en argument qu'une et une seule fois.

`SORT` : c'est la commande qui trie. Le tri se fait en fonction de la variable nommées `sort.key$`, qui est une variable de type chaîne de caractères déclarée implicitement pour chaque entrée. Le tri se fait dans l'ordre alphabétique pour ces chaînes de caractères. Le calcul de la valeur de `sort.key$` est à la charge de l'utilisateur, qui définira une `FUNCTION` dont ce sera la tâche, et qui sera appliquée à toutes les entrées via `ITERATE`. Évidemment, tout cela ne vaut que pour les styles qui trient. Par exemple, dans le style unsrt.bst, il n'y a pas de commande `SORT`.

`REVERSE` : similaire à `ITERATE`, mais elle est exécutée sur chaque entrée dans l'ordre inverse de la liste des entrées.

Voila pour la liste des commandes. Tu dois commencer à pouvoir imaginer à peu près comment tout ça va se goupiller. Tout ce qui nous reste à voir en fait, c'est comment écrire une `FUNCTION`... C'est l'objet des deux prochaines sections : avant de parler réellement de la définition des fonctions, je dois expliquer un peu la notation utilisée par BibTEX.

18 Notation polonaise inversée

Certains disent que la syntaxe utilisée par BibTEX, c'est du chinois, mais en fait non, c'est du polonais inversé. C'est comme ça que ça s'appelle. Ce genre de notation est basé sur une pile : on place d'abord les arguments sur la pile (comme une pile d'assiettes), puis on applique une opération sur les éléments de cette pile. Par exemple, l'addition va prendre les deux premiers éléments de la pile (les deux assiettes qui sont au dessus) et les additionner. Par exemple, `1 3 5 + 2 3 * -` s'exécute dans l'ordre suivant[31] :

- `1` est mis sur la pile. En supposant que la pile était vide au départ, elle dontient donc maintenant `1` ;
- `3` est mis sur la pile à son tour. La pile contient donc `1` et `3` ;
- `5` est empilé à son tour. La pile contient, dans l'ordre, `1`, puis, au dessus, `3`, puis, sur le sommet de la pile (on peut consevrer l'analogie avec une pile d'assiettes), `5` ;

[31] Je détaille bien ce point-là, c'est super important pour comprendre la suite. Si tu as déjà compris la notation polonaise inversée, tu peux zapper la description de l'exécution, tu vas pas apprendre grand'chose.

– `+` va prendre les deux éléments qui se trouvent au sommet de la pile (si la pile contient plus de deux éléments, sinon, ça va générer une erreur), les additionner, et mettre le résultat sur le sommet de la pile. Après cette opération, donc, la pile contient toujours le `1`, qui n'a pas bougé, et un `8`, qui est le résultat de l'addition de `5` et `3` ;
– `2` est empilé, toujours sur le dessus de la pile. Elle contient donc maintenant `1`, `8` et `2` ;
– `3` est empilé à son tour. La pile contient `1`, `8`, `2` et `3`.
– `*` est appliquée à la pile : cette opération prend ses deux arguments sur le sommet de la pile, ici `2` et `3`, les multiplie[32], et place le résultat sur le sommet de la pile. La pile contient désormais `1`, `8` et `6` ;
– `-` est enfin appliqué à la pile. Le `6` est retranché[33] au `8`, et le résultat est remis au sommet de la pile. La pile contient donc finalement `1`, et `2`.

Voila. Si tu n'as pas compris, c'est pas la peine d'aller plus loin, il est préférable de relire cette partie, et si ça suffit pas encore, il faut que t'ailles te renseigner ailleurs ou que tu en restes aux styles classiques de BibTeX... C'est dur à dire, mais c'est comme ça, c'est un conseil d'ami...

Pour celles et ceux qui ont compris, on va pouvoir attaquer le cœur du sujet.

19 Les fonctions internes

Dans le tableau qui suit, je décris toutes les fonctions internes qui existent. Pour chaque fonction, je précise, à gauche, les éléments qu'elle prend sur la pile, avec pour convention que l'élément le plus à droite est le plus haut sur la pile. À droite de la fonction, j'indique ce que la fonction met sur la pile. Les conventions sont les suivantes : $\mathcal{I}$ désigne un entier, $\mathcal{S}$ désigne une chaîne de caractères, $\mathcal{F}$ une fonction, $\mathcal{N}$ est le nom d'une variable[34], $\mathcal{C}$ est le nom d'un champ déclaré dans `ENTRY`[35], et $\mathcal{E}$ est un élément dont le type n'a pas d'importance.

$\mathcal{I}_1\ \mathcal{I}_2$	`+`	$(\mathcal{I}_1+\mathcal{I}2)$	l'opération d'addition de deux entiers[36](si, si !) ;
$\mathcal{I}_1\ \mathcal{I}_2$	`-`	$(\mathcal{I}_1-\mathcal{I}_2)$	l'opération de soustraction sur les entiers ;
$\mathcal{I}_1\ \mathcal{I}_2$	`>`	$\mathcal{I}$	renvoie 1 si $\mathcal{I}_1$ est *strictement supérieur* à $\mathcal{I}_2$, et 0 sinon[37] ;
$\mathcal{I}_1\ \mathcal{I}_2$	`<`	$\mathcal{I}$	renvoie 1 si $\mathcal{I}_2$ est *strictement inférieur* à $\mathcal{I}_1$, et 0 sinon ;
$\mathcal{I}_1\ \mathcal{I}_2$	`=`	$\mathcal{I}$	le test d'égalité de deux entiers. Le résultat est 1 si les deux entiers ont la même valeur, 0 sinon ;
$\mathcal{S}_1\ \mathcal{S}_2$	`=`	$\mathcal{I}$	le test d'égalité sur les chaînes de caractères ;
$\mathcal{S}_1\ \mathcal{S}_2$	`*`	$(\mathcal{S}_1\mathcal{S}_2)$	la concaténation de deux chaînes de caractères[38] ;
$\mathcal{E}\ \mathcal{N}$	`:=`		l'opération d'assignation. Si la variable est bien du même type que l'élément $\mathcal{E}$, elle prend sa valeur ;
$\mathcal{S}$	`add.period$`	$\mathcal{S}$	ajoute un point à la fin de $\mathcal{S}$, sauf si $\mathcal{S}$ se termine déjà par un point, un point d'exclamation ou un point d'interrogation ;

[32] Enfin, ça, c'est pour l'exemple, mais en fait, il n'y a pas de multiplication en BibTeX, et l'étoile sert pour la concaténation de deux chaînes de caractères.
[33] Cette opération n'est pas commutative : l'ordre des éléments influe sur le résultat. Je suppose ici que le soustration est définie comme ça, mais ça pourrait tout à fait être l'inverse.
[34] Un nom de variable, c'est un nom qui a été déclaré par `STRINGS` ou `INTEGERS`, et précédé d'une apostrophe pour préciser qu'on ne cherche pas à obtenir sa valeur. Par exemple, `'label`.
[35] Ou, bien sûr, le champ `crossref`, qui est implicite.
[36] Pour mettre un entier dans une fonction, il faut le faire précéder d'un `#`. Par exemple, si on veut calculer $2+2$, on écrira `#2 #2 +`.
[37] Il n'y a pas de booléens, donc on se débrouille avec les entiers...
[38] Il n'y a pas de multiplication d'entiers, en BibTeX[39].
[39] Je sais, je l'ai déjà dit.

	`call.type$`		exécute la fonction dont le nom est le type de l'entrée courante. Par exemple, pour une entrée de type **@book**, ça appelle la fonction `book`. Cette fonction `call.type$` ne peut évidemment pas apparaître dans l'argument d'un `EXECUTE`, mais uniquement d'un `ITERATE` (ou `REVERSE`, éventuellement), puisque ça dépend de l'entrée sur laquelle BibTEX est en train de travailler. Évidemment aussi, pour chaque type d'entrée qui est autorisé, il faut qu'une fonction du même nom soit définie, sinon BibTEX émettra un warning. ;
$\mathcal{S}$ `"t"`	`change.case$`	$\mathcal{S}$	convertit la chaîne $\mathcal{S}$ en minuscule, sauf la première lettre. Encore une fois, quand je dis *première lettre*, ça veut dire *première lettre à une profondeur d'accolades nulle*, étant entendu qu'un caractère spécial et tout ce qu'il contient est à profondeur nulle ;
$\mathcal{S}$ `"l"`	`change.case$`	$\mathcal{S}$	convertit la chaîne $\mathcal{S}$ en minuscule, sauf ce qui n'est pas à profondeur nulle ;
$\mathcal{S}$ `"u"`	`change.case$`	$\mathcal{S}$	convertit la chaîne $\mathcal{S}$ en majuscule, sauf... ;
$\mathcal{S}$	`chr.to.int$`	$\mathcal{I}$	convertit une chaîne *contenant un seul caractère* en sa valeur ASCII ;
	`cite$`	$\mathcal{S}$	met sur la pile la clé interne de l'entrée courante. Tout comme pour `call.type$`, cela ne peut être utilisé que dans un `ITERATE` ou `REVERSE` ;
$\mathcal{E}$	`duplicate$`	$\mathcal{E}$ $\mathcal{E}$	duplique l'élément qui se trouve au sommet de la pile ;
$\mathcal{E}$	`empty$`	$\mathcal{I}$	renvoie 1 si $\mathcal{E}$ est la chaîne vide ou le nom d'un champ vide, et 0 sinon ;
$\mathcal{S}_1$ $\mathcal{I}$ $\mathcal{S}_2$	`format.name$`	$\mathcal{S}$	prend le $\mathcal{I}$-ème nom de la chaîne $\mathcal{S}_1$, et le formatte suivant les spécifications indiquées dans la chaîne $\mathcal{S}_2$. C'est assez particulier, j'y reviendrai par la suite ;
	`global.max$`	$\mathcal{I}$	retoure la longueur maximale que peut avoir une chaîne de caractères. C'est utile lorsqu'on veut extraire un suffixe d'une chaîne de caractères, avec la fonction `substring$` ;
$\mathcal{I}$ $\mathcal{F}_1$ $\mathcal{F}_2$	`if$`		exécute la fonction $\mathcal{F}_1$ si l'entier $\mathcal{I}$ est *strictement positif*, et la fonction $\mathcal{F}_2$ sinon ;
$\mathcal{I}$	`int.to.chr$`	$\mathcal{S}$	place sur la pile le caractère dont le code ASCII est $\mathcal{I}$;
$\mathcal{I}$	`int.to.str$`	$\mathcal{S}$	convertit un entier en la chaîne de caractères équivalente[40] ;
$\mathcal{C}$	`missing$`	$\mathcal{I}$	renvoie 1 si le champ est présent dans l'entrée courante, et 0 s'il est absent ;
	`newline$`		passe à la ligne dans le fichier `.bbl` de sortie ;
$\mathcal{S}$	`num.names$`	$\mathcal{I}$	retourne le nombre de noms qui sont présents dans la chaîne $\mathcal{S}$. Plus précisément, puisque les noms sont séparés par `and` dans une liste de noms, cette fonction renvoie le nombre de `and` plus un ;
$\mathcal{E}$	`pop$`		suppime l'élément qui est au sommet de la pile ;

[40] Les scientifiques me diront que je n'ai pas défini la relation d'équivalence que j'utilise. Mais plutôt que de le faire, je donne un exemple que même les matheux comprendront : si $\mathcal{I}$ vaut 147, la chaîne de caractères retournée est `"147"`.

	`preamble$`	$\mathcal{S}$	met au sommet de la pile la concaténation de tous les arguments de `preamble` qui se trouvent dans la base de données ;
$\mathcal{S}$	`purify$`	$\mathcal{S}$	supprime tous les caractères non-alphanumériques de la chaîne $\mathcal{S}$. Plus précisément, cette fonction conserve les lettres sans changer leur casse, conserve les chiffres et les espaces, remplace chaque tabulation, chaque tiret et chaque tilde (~) par un espace, et supprime les autres caractères « standards », *i.e.* les autres caractères qui apparaissent dans la table de caractères de TEX, dont il est question dans l'annexe C du TEXbook. Les caractères spéciaux font exception à ces règles : dans un caractère spécial, toutes les commandes LATEX (qui commencent par un \ ) sont supprimées, les espaces, tabulations, tirets et tildes sont supprimés, de même que les autres caractères standards. D'un point de vue plus pratique, maintenant : cette commande permet de nettoyer les chaînes de caractères afin de pouvoir les trier. C'est pour cela qu'il est important de se ramener uniquement à des caractères connus. En particulier, j'insiste sur le fait que « `é` » et « `\' e` », qui ne seront pas triés de la même façon. ;
	`quote$`	$\mathcal{S}$	met sur la pile le caractère `"` ;
	`skip$`		ne fait rien ;
	`sort.key$`	$\mathcal{S}$	cette fonction est plutôt une variable relative à une entrée, en fait, et sera utilisée pour trier les entrées. Avant d'exécuter la commande `SORT`, il faudra donc définir la valeur de cette variable pour chaque entrée ;
... $\mathcal{E}$ $\mathcal{E}$	`stack$`		vide complètement la pile et affiche son contenu sur la sortie standard (et donc pas dans le fichier `.bbl`) ;
$\mathcal{S}$ $\mathcal{I}_1$ $\mathcal{I}_2$	`substring$`	$\mathcal{S}$	place sur la pile la chaîne de caractère extraite de $\mathcal{S}$, commençant à la position $\mathcal{I}_1$ et de longueur $\mathcal{I}_2$. Le premier caractère est à la position 1. Chaque caractère, même à l'intérieur d'un caractère spécial, compte pour un caractère ;
$\mathcal{E}_1$ $\mathcal{E}_2$	`swap$`	$\mathcal{E}_2$ $\mathcal{E}_1$	permute les deux éléments du sommet de la pile ;
$\mathcal{S}$	`text.length$`	$\mathcal{I}$	renvoie le nombre de caractères d'une chaîne de caractères, un caractère spécial étant compté pour un seul caractère, et les accolades n'étant pas comptées ;
$\mathcal{S}$ $\mathcal{I}$	`text.prefix$`	$\mathcal{S}$	renvoie les $\mathcal{I}$ premiers caractères de la chaîne $\mathcal{S}$, un caractère spécial comptant toujours pour un seul caractère. Attention, les accolades qui ne font pas partie d'un caractère spécial ne comptent pas pour un caractère, mais elles seront retournées quand meme. Ainsi, pour la chaîne `{{\LaTeX}}`, le préfixe de taille 3 sera `{{\La}}` ;
$\mathcal{E}$	`top$`		supprime le premier élément de la pile, et l'affiche sur la sortie standard ;
	`type$`	$\mathcal{S}$	place sur la pile le type de l'entrée courante ;
$\mathcal{S}$	`warning$`		écrit sur la sortie standard le message d'avertissement $\mathcal{S}$;

$\mathcal{F}_1$ $\mathcal{F}_2$	`while$`		exécute successivement $\mathcal{F}_1$ et $\mathcal{F}_2$ tant que $\mathcal{F}_1$ retourne une valeur strictement positive. $\mathcal{F}_1$ doit nécessairement retourner un entier ;
$\mathcal{S}$	`width$`	$\mathcal{I}$	retourne la longueur de la chaîne $\mathcal{S}$, en centièmes de points, pour la fonte *cmr*10, dans sa version de juin 1987. Bon, en fait, tous ces détails sont assez peu importants, ça sert surtout à *comparer* les longueurs des labels, pour l'argument de l'environnement `thebibliography` ;
$\mathcal{S}$	`write$`		écrit la chaîne $\mathcal{S}$ dans le fichier `.bbl`.

Voilà, c'est tout. Avec tout ça, on peut définir plein de fonctions, extraire les entrées de la base de données, les trier, les étiqueter et les mettre en page. Il nous faut maintenant voir comment on se sert de tout ça. Pour commencer, la fonction qui gère les noms...

20 La fonction `format.name$`

Cette fonction est un peu particulière, notamment dans la forme de ses arguments. Je rappelle sa syntaxe :

$\mathcal{S}_1$ $\mathcal{I}$ $\mathcal{S}_2$ `format.name$` $\mathcal{S}$

Le premier argument, la chaîne $\mathcal{S}_1$, est généralement le contenu des champs `author` ou `editor`. C'est donc une liste de noms au format « BibTEX », deux noms étant séparés par `"and"`. Si les noms sont bien formatés, comme je l'ai expliqué à la section 12, BibTEX arrivera à distinguer ce qui fait partie du nom, du prénom, de la particule et du « complément ». Mais cela ne lui dit pas dans quel ordre et dans quel format il doit sortir ces données. C'est l'argument $\mathcal{S}_2$ qui va le lui indiquer le format. Voici un exemple de format possible :

```
"{ff }{vv }{ll}{, jj}"
```

Ça peut sembler obscur, vu comme ça et sans plus de précisions. Mais en voici :
- tout d'abord, le format global : c'est une liste de caractères placés entre accolades, et entourée de guillemets. C'est toujours comme ça ;
- le `ff` représente la partie `First`, donc le prénom, dans le tri qu'a fait BibTEX sur le nom en question. De même, `vv` représente le `von`, *i.e.* la particule. `ll` représente le `Last`, c'est à dire le nom de famille, et enfin, bien sûr, le `jj` représente le `jr`, ce que j'ai appelé le « complément » ;
- la fonction `format.name$` va remplacer, dans chaque accolade, les `ff`, `vv`, `ll` et `jj` par leurs valeurs. Si un élément est vide, tout le contenu de l'accolade est supprimé. Ainsi, avec le format ci-dessus, si le `jr` est vide[41], la virgule qui doit le précéder n'apparaîtra pas.

Voila qui doit clarifier à peu près la situation. Il reste à préciser, si ce n'est pas implicite, le rôle du deuxième argument : il sert simplement à préciser sur quel nom on veut travailler, parmi les différents noms qui figurent dans la chaîne $\mathcal{S}_1$.

Enfin, il existe des formats « abrégés », qui permettent de ne prendre que les initiales des noms ou des prénoms. Par exemple, il suffit de remplacer `ff` par `f` pour obtenir le premier caractère du prénom. En cas de prénoms multiples, le premier caractère de chaque prénom est affiché, deux caractères consécutifs étant séparés par un point. Cela signifie en particulier qu'il n'y a pas de point, par défaut, à la fin de l'abréviation, et qu'il faut donc l'ajouter. Par exemple,

```
"Goossens, Michel and Mittelbach, Franck and Samarin, Alexander"
#2 "{f. }{vv }{ll}{, jj}" format.name$
```

[41] Ce qui arrive assez fréquemment, il faut bien le dire.

retournera *F. Mittelbach*. Regardons encore quelques exemples[42] :

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{vv }{ll}{, f}" format.name$
```

donnera *de La Vallée Poussin, C.J.G.N* sans le point final après le *N*, puisqu'il n'a pas été précisé.

```
"Doppler, {\relax Ch}ristian Andreas"
#1 "{f. }{vv }{ll}" format.name$
```

donnera *Ch. A. Doppler*. Pour terminer,

```
"Jean-Baptiste Poquelin" #1 "{f. }{vv }{ll}" format.name$
```

sortira *J.-B. Poquelin*.

Si tu es perspicace, une subtilité reste à éclaircir : pour créer les labels à partir des noms d'auteurs, c'est bien évidemment ce genre de fonctions qui est utilisé. Pour un livre de La Vallée Poussin, par exemple, on pourra espérer un label de la forme `LVP`. On essaiera donc quelquechose comme ça :

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{l}" format.name$
```

qui donnera lamentablement un *L. V. P* assez loin du résultat escompté. Mais en fait, il existe un moyen de préciser à BibTEX s'il doit ou non mettre le point entre deux initiales, et même de façon plus générale, ce qu'il doit mettre. Par défaut, il met `.~`, (un point et une espace insécable). Mais en faisant :

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{l{}}" format.name$
```

on obtient bien *LVP*. L'accolade après le `l` permet de préciser ce qu'on veut mettre entre les différentes composantes de cette partie (le `Last`, ici) du nom. On peut également ajouter du texte une seule fois : si c'est juste une espace, ça peut se faire directement. Si c'est du texte, pour qu'il ne soit pas confondu avec les spécifications `ff`, `ll`, ... on le mettra entre accolades. Et pour qu'il ne soit pas confondu avec le texte mis entre toutes les composantes, on ajoutera éventuellement une argument vide (ou plutôt, contenant une espace) après le `ff` ou `ll`. Pour clarifier tout ça, un petit exemple :

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{{\scshape\bgroup}ff{ }{\egroup}" format.name$
```

donnera `{\scshape\bgroup}Charles Jean Gustave Nicolas{\egroup}`, et donc « CHARLES JEAN GUSTAVE NICOLAS ». Noter que ça ne marche que parce que le `\bgroup` est fermé par l'accolade qui le suit immédiatement, et que le `\egroup` ferme le groupe commencé par l'accolade qui le précède.

Voila voila... De nouveau, je pourrais m'arrêter là, tu as tous les éléments en main pour attaquer ton propre style bibliographique, ou pour aller modifier un style existant. Mais ne pas faire d'exemple, c'est pas très pédagogique. Je vais donc dans les sections suivantes, donner quelques exemples de fonctions, et des petits trucs pratiques. D'ailleurs, on va commencer par ça.

[42]Le premier est un classique du genre, bien que Oren PATASHNIK se soit trompé dans sa documentation de BibTEX, confondant le père et le fils. Le père était géologue, le fils, mathématicien.

21 Petits trucs pratiques

21.1 Fichier minimal

Le principal petit truc pratique, c'est de savoir se faire un fichier minimal pour tester. Le cas le plus fréquent, c'est pour tester le « codage » d'un nom. Vue la clarté de mon exposé sur le codage des noms, à la section 12, il te sera surement plus utile de savoir comment tester si un codage que tu choisis va marcher. L'idée est donc de faire un fichier minimal, qui affiche sur la sortie standard ce que tu veux tester. Par exemple, disons que tu veux tester comment est découpé, en termes de `First`, `von` et `Last`, le nom suivant :

```
de la Cierva y Codorn{\'\i}u, Juan
```

Le fichier `.bst` minimal doit contenir la commande `ENTRY`, la commande `READ`, et une fonction qui calculera la valeur recherchée. Ce fichier, nommons le `min.bst`, contiendra donc :

```
ENTRY {}{}{}

FUNCTION {test}
{"de la Cierva y Codorn{\'\i}u, Juan"
#1 "{ff - }{vv - }{ll}" format.name$ top$}

READ

EXECUTE{test}
```

De cette façon, les différents éléments seront séparés par des tirets, on saura si le codage est acceptable ou pas.

Il nous faudra également un fichier `.aux`, disons `min.aux`, qui appelle ce style bibliographique :

```
\bibstyle{min}
```

Ce fichier ne contient pas de fonctions `\bibdata`, précisant la base de données à utiliser, ni `\citation`, précisant les entrées à extraire de la base de données. BibTEX ne va pas apprécier, mais tant pis, on fera abstraction des messages d'erreur, le principal étant qu'il réponde à la question qu'on lui pose...

Avec ces deux fichiers, on peut mintenant lancer BibTEX sur le fichier `.aux` :

```
% bibtex min
This is BibTeX, Version 0.99c (Web2C 7.4.5)
The top-level auxiliary file: min.aux
The style file: min.bst
I found no \citation commands---while reading file min.aux
I found no \bibdata command---while reading file min.aux
Warning--I didn't find any fields--line 1 of file min.bst
Juan - de~la Cierva~y - Codorn{\'\i}u
(There were 2 error messages)
```

On a donc nos messages d'erreurs, mais également la réponse : le codage du nom n'est pas correct, puisque le `von` ne devrait contenir que `de la`. Juste une remarque sur les espaces insécables, même si on ne peut pas y toucher de toutes façons : BibTEX les place entre le premier et le deuxième élément, et entre l'avant-dernier et le dernier.

Voila, donc, pour un exemple de fichier minimal de test. Bien entendu, de la même façon, tu peux tester le comportement des autres fonctions un peu bizarres de BibTEX, comme `purify$`, pour vérifier qu'elle se comporte bien comme je l'ai expliqué précédemment... Je te laisse le faire, c'est un bon exercice.

21.2 Lignes trop longues

BibTEX n'aime pas les lignes trop longues. Il les coupe généralement a 80 caractères. Ce n'est généralement pas gênant, parce qu'il ajoute un `%` à la fin de la ligne coupée, donc après un passage de LATEX, on n'y voit que du feu. Sauf pour une URL : la commance `\url` du package `url.sty` ne considère pas le comme un caractère de commentaire, et va donc l'afficher. Pour contourner le problème, on peut, par exemple, user du fait que les espaces et sauts de ligne sont ignorés par `\url` : on peut donc en mettre n'importe où dans une URL, et BibTEX coupera les lignes trop longues à cet endroit-là.

22 Quelques exemples de petites fonctions simples

22.1 Les fonctions booléennes

Comme on l'a vu plus haut, il n'y a pas de booléen en BibTEX, et on se débrouille avec les entiers pour simuler les booléens. C'est exactement comme ça qu'est construite la fonction `if$`, d'ailleurs. Mais quand même, il pourrait être intéressant d'avoir la négation, la conjonction et la disjonction. Ces fonctions sont défininés à la figure 1.

```
FUNCTION {not}
{   { #0 }
    { #1 }
  if$
}

FUNCTION {and}
{   'skip$
    { pop$ #0 }
  if$
}

FUNCTION {or}
{   { pop$ #1 }
    'skip$
  if$
}
```

Fig. 1 – Définitions des fonctions booléennes de base

J'explique juste les deux premières : pour `not`, on suppose qu'on a un booléen (donc `#0` ou `#1`) sur la pile. Ce booléen va être lu par la fonction `if$`. Si c'est `#1`, alors `#0` et retourné, et vice-versa. Ça fait donc bien ce qu'on veut.

Pour le `and` maintenant : on a deux booléens sur la pile, et on doit retourner `#0`, sauf si les deux booléens valent tous les deux `#1`. On fait donc un `if$` sur le premier. S'il vaut `#1`, on laisse le deuxième sur la pile, il est égal au résultat recherché. S'il vaut `#0`, on supprime le deuxième booléen et on le remplace par `#0`.

22.2 Définissons une multiplication

La fonction permettant de calculer le produit de deux entiers est écrite à la figure 2. Je ne détaille pas trop, les commentaires doivent suffire. Mais c'était juste pour montrer que ça peut se faire. Pour t'entrainer, tu peux essayer la division, c'est le même style. Ce sera une division entière, on pourrait aussi programmer le reste de la division, le calcul des PGCD et PPCM, tester si un nombre est premier, tout ça, mais je n'en vois pas trop l'intérêt. La multiplication, elle, va nous servir dans ce qui suit.

```
INTEGERS { a b }

FUNCTION {mult}
{
 'a :=                        %% on memorise la premiere valeur
 'b :=                        %% on memorise la deuxieme valeur

 b #0 <                       %% L'idee ici est de mettre le signe
    {#-1 #0 b - 'b :=}        %% de b sur la pile, et de prendre
    {#1}                      %% la valeur absolue de b.
 if$                          %%

 #0                           %%
 {b #0 >}                     %% tant que b est strictement positif,
 {                            %% on ajoute a au premier element de
   a +                        %% la pile, et on decremente b. A la
   b #1 - 'b :=               %% fin, on a bien le produit de a
 }                            %% par b.
 while$                       %%

 swap$                        %% Enfin, on corrige le signe, si
   'skip$                     %% besoin, en testant le signe de b
   {#0 swap$ -}               %% que l'on avait mis sur la pile au
 if$                          %% depart.
}
```

FIG. 2 – La fonction « multiplication »

22.3 Conversion d'une chaîne de caractères en entiers

Il n'existe pas de fonction interne pour transformer une chaîne de caractères en entiers. L'idée serait de prendre une chaîne de chiffres et la transformer en son équivalent entier. Et de renvoyer un avertissement en cas de problème, bien sûr. Nous allons faire ça récursivement, à l'aide d'une fonction intermédiaire, comme tu peux le constater à la figure 3.

Plusieurs choses à noter, pour cet exemple : tout d'abord, il nécessite d'avoir défini les fonctions des exemples précédents, la multiplication en particulier. Par ailleurs, la fonction `chr.to.value` met en avant un principe qu'il faut s'efforcer de respecter : quoi qu'il arrive, il faut qu'une fonction renvoie

```
FUNCTION {chr.to.value}       %% La valeur ASCII d'un caractere.
{
  chr.to.int$ #48 -           %% en ASCII, "0" -> 48
  duplicate$ duplicate$       %%           "1" -> 49
  #0 < swap$ #9 > or          %%                ...
  {                           %%           "9" -> 57
      #48 + int.to.chr$
      " n'est pas un nombre..." *
      warning$                %% Si c'est pas un chiffre,
      pop$ #0                 %% on retourne 0...
    }
  {}
  if$
}

FUNCTION {str.to.int.aux}     %% La fonction auxiliaire.
{
  {duplicate$ empty$ not}     %% Tant que la chaine n'est pas vide
    {                         %% on prend son premier chiffre
      swap$ #10 mult 'a :=    %% et on le met comme il faut...
      duplicate$ #1 #1 substring$
      chr.to.value a +
      swap$
      #2 global.max$ substring$
    }
  while$
  pop$
}

FUNCTION {str.to.int}
{                             %% Le nombre est-il negatif ?
  duplicate$ #1 #1 substring$ "-" =
    {#1 swap$ #2 global.max$ substring$}
    {#0 swap$}
  if$
                              %% On initialise, puis on appelle
  #0 swap$ str.to.int.aux     %% la fonction auxiliaire.
  swap$
    {#0 swap$ -}              %% On met le signe comme il faut.
    {}
  if$
}
```

Fig. 3 – Conversion d'une chaîne de caractères en entiers

toujours le même type de sortie. Par exemple, ici, si le caractère n'est pas un chiffre, on s'en plaint, mais on retourne néanmoins une valeur. Ce principe permet d'éviter de nombreuses erreurs.

22.4 Inversion de l'ordre chronologique

Une application intéressante de ce qui précède est l'inversion de l'ordre chronologique. Analysons tout d'abord le problème : imaginons que deux publications soient des mêmes auteurs. Ce qui les différenciera dans le tri, c'est alors l'année. Dans la chaîne `sort.key$`, l'année est insérée après les noms d'auteurs, et donc la publication la plus vieille apparaîtra en premier. Imaginons qu'on veuille l'inverse...

Une idée rapidement écartée est d'utiliser la commande `REVERSE` au lieu de `ITERATE` : si on fait ça, on inverse l'ordre alphabétique également, c'est rarement souhaitable.

La deuxième idée, la bonne, est de changer l'année en quelquechose qui diminue lorsque l'année augmente, afin qu'un truc vieux ait une valeur plus grande qu'un truc plus récent. Le candidat sera une constante de laquelle on retirera l'année. La constante ne doit pas être choisie n'importe comment, cependant : il faut que le résultat soit un nombre positif, et qu'il ait toujours le même nombre de chiffres (parce que, dans l'ordre alphabétique, `1000` est plus petit que `1001`, mais `999` est plus grand que `1000`. Bref, on prendra 4000 pour la constante, ça nous laisse le temps de voir venir...

Une version modifiée de `presort` est donnée à la figure 4. Elle suppose bien entendu que l'on ait défini auparavant la commande `str.to.int`.

```
FUNCTION {presort}
{
  ...
  "    "
  *
  % Ca, c'etait pour l'ordre chronologique normal :
  %    year field.or.null sortify
  #4000 year str.to.int - int.to.str$ sortify
  *
  "    "
  ...
}
```

FIG. 4 – Inversion de l'ordre chronologique

Cinquième partie

Autres exemples d'utilisation de BibTEX

Voici quelques exemples d'utilisation détournée de BibTEX. Tous ces exemples sont en cours de développement. Les dernières nouvelles sont sur ma page web, `http://www.lsv.ens-cachan.fr/~markey/bibtex/`.

23 Une liste de publications

Ça commence doucement, puisque cet exemple reste assez proche du but initial de BibTEX : une bibliographie. D'ailleurs, l'idée est ici d'utiliser un fichier `.bib` classique, et d'en extraire les publications d'une personne, en indiquant les éventuels collaborateurs. Par exemple, on aura ceci :

Livres écrits par Michel Goossens

[1] *the LATEX Companion*. Addison-Wesley, 1998. En collaboration avec Frank Mittelbach et Alexander Samarin.

[2] *the LATEX Graphics Companion*. Addison-Wesley, 1999. En collaboration avec Sebastian Rahtz et Frank Mittelbach.

[3] *the LATEX Web Companion*. Addison-Wesley, 1999. En collaboration avec Sebastian Rahtz.

Les différences par rapport à un style classique résident principalement dans trois points :
- il faut virer les entrées qui ne nous intéressent pas, c'est à dire celle où l'auteur en question n'a pas pris part ;
- il faut mettre en forme un peu différemment les noms des auteurs.

Le troisième point annoncé est la méthode de configuration de ce style. Au lieu d'obliger l'utilisateur à aller trifouiller dans le fichier `.bst`, on crée un nouveau type d'entrée, nommé **@config**, dans lequel on définira le nom de l'auteur et quelques petits trucs pour configurer ledit style. Cette entrée contiendra des champs spéciaux, qui ne sont pas utilisés par les autres entrées.

24 Un carnet d'adresses

Bah oui, une liste d'adresses, c'est une base de données. On va la trier, la classer alphabétiquement, éventuellement regrouper plusieurs lettres, puis mettre tout ça dans un tableau. BibTEX est l'outil idéal (enfin... je me comprends...) pour gérer tout ça.

Bien entendu, dans ce cas, le style « bibliographique » n'a plus rien à voir avec un style bibliographique classique, les entrées non plus. Chaque entrée contiendra un nom (obligatoire) et une liste de champs donnant l'adresse, le numéro de téléphone, de portable, l'adresse électronique... BibTEX va remettre tout ça en ordre. Voici ce qu'on obtient, par exemple (avec le signet dans la marge) :

Markey Nicolas	LSV – ENS Cachan 61, avenue du Président Wilson 94235 Cachan Cédex `markey@lsv.ens-cachan.fr`	☎ 🖷	01.47.40.75.32 01.47.40.24.64

mn

Tout cela étant assez configurable : on peut choisir comment seront regroupées les lettres, la taille et l'emplacement des lettres dans la marge, ...

25 Un glossaire

Il existe déjà quelques outils pour créer un glossaire, c'est à dire un « dictionnaire » des différents termes techniques utilisés dans un article, dans un livre, ... BibTEX peut également faire l'affaire, puisque c'est, dans le principe, exactement le même fonctionnement que BibTEX. La différence est l'éventuel ajout des numéros de page où les mots sont utilisés. Cela ne pose pas de réel problème si on utilise le package backref.sty.

26 Francisation des styles bibliographiques

Il existe plusieurs façons de franciser un style bibliographique. Il faut penser, lorsqu'on remplit le fichier `.bib`, à éviter autant que possible tout ce qui est susceptible d'être « traduit » en fonction du style bibliographique. Pas le titre, bien sûr, mais par exemple, le mois de parution. Par ailleurs, BibTEX ajoute lui-même (enfin, c'est le style bibliographique qui le fait, plutôt) des mots, comme par exemple le « and » entre l'avant-dernier et le dernier auteur.

Pour traduire tout ça, plusieurs solutions :

- faire un style bibliographique pour chaque langue. C'est parfois un peu fastidieux, mais finalement, on ne change pas de langue si souvent que ça. Il existe ainsi de nombreux styles bibliographiques traduits dans moult langues différentes.
  Dans les styles francisés disponibles sur `http://www.ctan.org/tex-archive/biblio/bibtex/contrib/bib-fr/`, tous les mots pouvant être traduits sont regroupés au début du fichier de style, ce qui fait que la traduction vers d'autres langues, ou la modification, est facile.
- remplacer tout ce qui peut être traduit par des commandes LATEX. C'est ce que fait en particulier babelbib, disponible sur `http://www.ctan.org/tex-archive/biblio/bibtex/contrib/babelbib/`. Ces styles reposent sur l'utilisation du package babel.sty, et permettent de définir une langue pour chaque entrée (ce qui permet d'avoir les bonnes césures, par exemple). Le textes « statiques » sont définis par l'intermédiaire de commandes LATEX, ce qui permet de n'avoir qu'un seul style bibliographique pour toutes les langues.

Références

[Bay95] Benjamin Bayart. *Joli Manuel pour LaTeX*, décembre 1995.

[Dal99a] Patrick W. Daly. *A LaTeX Package to Place Bibliography Entries in Text*, février 1999.

[Dal99b] Patrick W. Daly. *A Master Bibliographic Style File*, mai 1999.

[Dal99c] Patrick W. Daly. *Natural Sciences Citations and References*, mai 1999.

[Des00] Bernard Desgraupes. *LaTeX : Apprentissage, Guide et Références*. Vuibert, 2000.

[Dom97] Éric Dommenjoud. *The **footbib** package*, mars 1997.

[GMS98] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *the LaTeX Companion*. Addison-Wesley, 1998.

[Han00a] Thorsten Hansen. *The **bibunits** package*, octobre 2000.

[Han00b] Thorsten Hansen. *The **multibib** package*, janvier 2000.

[Lam97] Leslie Lamport. *LaTeX : a Document Preparation System*. Addison-Wesley, mars 1997.

[Pat88a] Oren Patashnik. BibT*EXing*, février 1988.

[Pat88b] Oren Patashnik. *Designing* BibTeX *styles*, février 1988.

LaTeX

BibTEX

Les bst