Compte Rendu
Projet de compilation 2015-2016

January 11, 2016

Date :	January 11, 2016
Rédigé par :	Thomas Capet Yohann Henry

Contents

1 Introduction

Le projet consiste en la création d'un exécutable permettant la gestion de références contenues dans des fichiers `bibtex` ou LaTeX. L'exécutable devait notamment extraire les clés de fichiers d'extension `.tex`, ainsi que les données contenues dans un fichier `.bib`. L'application effectuerait ensuite selon les options différents traitements et afficherait le résultat sur la sortie standard ou dans un fichier.

2 Notice d'utilisation

2.1 Option -t

`mybib commande -t type nomFichier.bib`
Affiche sur la sortie standard le contenu du fichier `nomFichier.bib` en le privant de tous les blocs différents de `type`.

2.2 Option -c

`mybib commande -c nomFichier.bib`
Affiche sur la sortie standard les clés en doublon dans le fichier `.bib`.

2.3 Option -b

`mybib -b toto.tex`
Si l'option `-b` est choisie le fichier `.tex` est parsé pour extraire les inclusions de fichier `.bib` et `.tex` afin d'en extraire les clés bibliographiques. Les références tirées du `.bib` correspondant aux clés sont ensuite affichées sur la sortie standard.

2.4 Option -s

`mybib commande -s fichier.bib`
Avec cette option, le fichier `.bib` est parsé avant de produire sur la sortie le fichier en transformant les champs `publisher`, `organization`, `series` et `journal` en des variables définies et factorisées au début du fichier. Chaque type de variable est définie dans un @String différent.

2.5 Option -o

`mybib commande -o nomFichier`
Par défaut, l'affichage se fait sur la sortie standard. Avec l'option -o, l'affichage sur la sortie standard est à présent reporté sur le fichier `nomFichier`.

3 Structures de données utilisées

3.1 Collections

Pour travailler, nous avons d'abord développé plusieurs structures de données génériques de type collection :

- Les listes ordonnées
- Les sets ordonnées (ainsi que son homologue non ordonné)

- Les maps sous la forme d'une liste ordonnée
- Les tables de hachage

Toutes ces structures sont ensuite utilisées dans Flex et Yacc pour stocker les données de manière optimisée.

3.2 Références

Nous avons ensuite créé un type Référence. Ce dernier représente toutes les données nécessaires pour enregistrer une référence d'un bloc `Bibtex`.

```
struct _ref {
    TypeReference type;
    char* id;
    char* champs[NB_CHAMP_REF];
};

typedef struct _ref * Reference;
```

Les attributs `type` et `id` sont trivialement le type et l'id de la référence. L'attribut `champs` est un tableau qui attribue à un type de champ une valeur en chaîne de caractères. Par défaut, cette valeur est initialisée avec une chaine de caractères vide. Plusieurs fonctions sont présentes dans `references.h` pour simplifier les différentes options demandées. Notamment, une des fonctions permet d'écrire dans un flux le contenu d'une référence avec tous les champs obligatoires ou optionnels de ce type.

3.3 Les gestionnaire de références

Nous avons finalement créé un type `RefManager`, un gestionnaire de références. Ce dernier nous permet de stocker toutes les références contenues dans les `.tex` et les `.bib`. Une fois le gestionnaire rempli, on traite les données selon les options demandées.

```
struct _ref_manager {
        HashMap map;
        int onlyUpdateMode;
};

typedef struct _ref_manager* RefManager;
```

Le gestionnaire de références est une simple table de hachage avec pour clé l'id d'une référence et en valeur la référence correspondante. Comme pour `Référence`, plusieurs fonctions utilitaires ont permis de simplifier les différentes options.

4 La grammaire

4.1 Les tokens

La grammaire commence dans un mode selon le type de fichier qu'elle étudie. Elle reconnait donc soit un fichier Bibtex, soit un fichier Latex, mais jamais les deux en même temps. Ainsi, pour l'option `-b`, la grammaire parse tous les fichiers `.tex` puis passe au `.bib` automatiquement.

4.1.1 Les tokens pour parser les .tex

- `CITE : \cite{[^}]+}`
- `NOCITE : \nocite{[^}]+}`
- `BIBNAME : \bibname{[^}]+}`
- `INCLUDE : \include{[^}]+}`
- `INPUT : \input{[^}]+}`

4.1.2 Les tokens pour parser les .bib

- `TYPEREF : @Article|@Book|@Booklet|@Conference|@Inbook|@Incollection |@Inproceedings|@Manual|@Mastersthesis|@Misc|@Phdthesis |@Proceedings|@Techreport|@Unpublished`
- `TYPECHAMP: address|abstract|annote|author|booktitle|chapter|crossref |edition|editor|eprint|howpublished|institution|isbn|journal|key|month|note |number|organization|pages|publisher|school|series|title|type|url|volume|year`
- `KEY : {[a-zA-Z0-9]+(:[a-zA-Z0-9]+)*,`
- `VAL : {.*},|{.*}\n`

4.2 Définition de la grammaire

```
file: bloc file
    |;

bloc: TYPEREF KEY champs '}';

champs: TYPECHAMP champs
    |TYPECHAMP VAL champs
    |;
```