Lex et Yacc

Geoffrey DESBROSSES

Mars 2016

1 Introduction

Dans ce document, nous allons présenter deux outils : Lex et Yacc. Ce sont deux outils utilisés en compilation car ce sont des compilateurs de compilateurs, ils permettent de produire du code en langage C. Lex et Yacc sont souvent utilisés ensemble pour vérifier complètement une grammaire. On utilise ces deux outils au début de la compilation comme l'indique la figure 1.

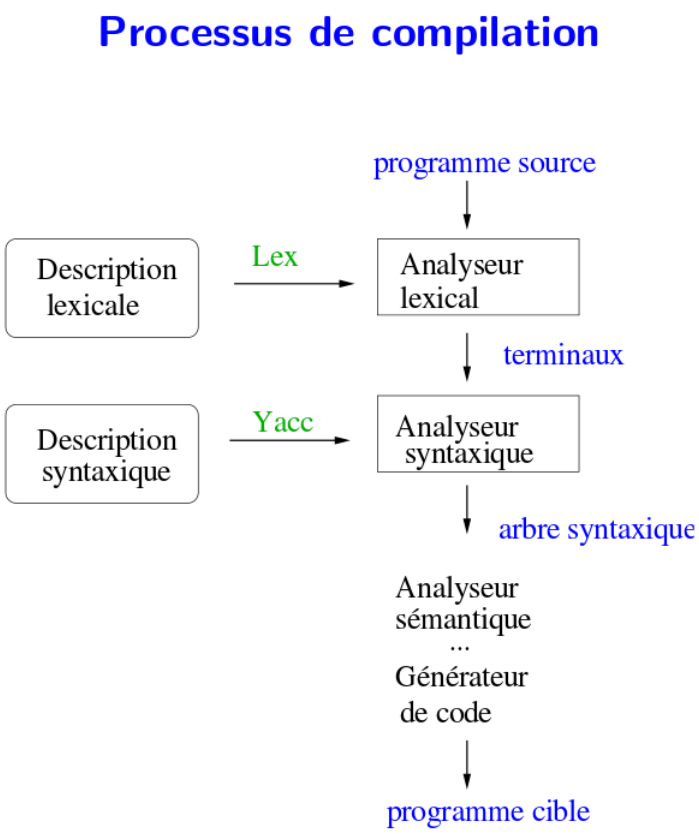

FIGURE 1 – Processus de la compilation

2 L'outil lex

Lex est un générateur d'analyseur lexical, il marche très simplement de la façon suivante : il prend la définition des unités lexicales en entrée, les interprètes puis produit un automate fini et déterministe minimal qui permet de reconnaître les unités lexicales.

Un fichier source lex contient principalement des spécifications sous forme d'expressions relationnelles. Ces expressions comportent des classes de token et des classes d'action. Ces classes d'action s'exécutent à la reconnaissance d'un token.

Un fichier Lex est généralement de la forme suivante :

```
declarations
%%
productions
%%
code additionnel
```

Cependant, seul la partie de productions est obligatoire.

La partie déclaration contient des expressions régulières qui définissent des notions non terminales (lettres, nombres...). Cette partie peut également contenir du code C s'il est encadré de balises %{ et %} ou si la ligne commence par un blanc. Ce code sera intégralement recopié dans le fichier lex.yy.c

La deuxième partie (productions) contient une liste d'expressions régulières et d'actions de la forme :

```
expression_reguliere action
```

Si l'action est absente alors lex recopiera la chaîne de caractères comme tel. Si l'action est présente, elle devra être écrite en langage C. Action est donc un petit fragment de code C à exécuter lorsque le token correspondant à l'expression est reconnu. Par exemple, on peut avoir la règle suivante qui changera toutes les chaînes de caractères "eau" par "elle", ainsi, tourtereau deviendra tourterelle.

```
eau printf("elle");
```

Plusieurs métasymboles existent pour définir facilement ces règles, la liste n'est pas très longue :

```
" \ [ ] ^ ? . * + | ( ) $ / { } %
```

Ils ont chacun une signification, par exemple le symbole "?" désigne une expression optionnelle, par exemple aaa ?b reconnaîtra aaa mais aussi aaab.

La dernière partie du fichier contient du code additionnel écrit en C.

3 L'outil yacc

Yacc sert également à vérifier la grammaire, la différence avec lex c'est que c'est un générateur d'analyse syntaxique et non lexicale.

Un fichier yacc possède lui aussi une structure à respecter, définie de la même manière et avec les mêmes conditions (seule la partie 2 est obligatoire) :

```
declarations
%%
productions
%%
code additionnel
```

La première partie contient des déclarations et/ou des définitions encadrées par les balises %{ et %}.

La deuxième partie contient des productions de la grammaire du langage. Ces productions ont la même partie gauche et sont chacune associée à une action sémantique, elles s'écrivent sous la forme suivante :

```
expression: expression '+' expression { action semantique }
          | expression '-' expression { action semantique }
          | expression '*' expression { action semantique }
          | ...
          ;
```

Le symbole qui constitue la partie de gauche de la première production est appelé axiome de la grammaire. Les expressions peuvent être des notions terminales du langage ou non.

La troisième partie qui comporte le code additionnel contient donc du code écrit en C. Cette dernière partie, si elle existe, doit forcemment contenir les méthodes main() (qui appelle la méthode yyparse()) et yyerror(char *message) qui est appelée lorsqu'une erreur de syntaxe est trouvée.

4 Conclusion

Les outils Lex et Yacc sont donc des compilateurs de compilateurs servant pour le langage C. Ils interviennent au début de la compilation d'un programme pour voir si la grammaire est écrite correctement.

Nous avons également écrit ce genre de programme lorsque nous avons écrit notre grammaire zéro pour voir si elle était correctement écrite. Dans notre projet, lorsque nous avons généré la forêt, nous sachions si la grammaire avait la bonne syntaxe et était écrite correctement lexicalement. Notre compilateur est beaucoup moins poussé que Lex et Yacc mais les grands principes sont là.