Compilateur Lua - Projet IFT3150

Philippe Caron

28 août 2017

1 Introduction

Le but du préprocesseur est de standardiser tous les fichiers de code de manière à ce que le compilateur puisse éventuellement traiter des fichiers suivant un certain patron plus strict de formattage. C'est important puisque rien ne garanti que tous les programmeurs ont la même éthique d'écriture. Dans ce cas-ci particulièrement, le préprocesseur est très utile car le langage Lua a une syntaxe plus permissive que la plupart des autres langages impératifs. Alors que beaucoup exigent le fameux « ; » à la fin de chaque ligne, en Lua il n'est nécessaire que pour séparter deux instructions consécutives ambiguës. Il existe également plusieurs forme équivalentes pour exprimer la même chose, ce qui pourrait devenir compliquer pour un compilateur. Finalement, les expressions ne sont pas forcément correctement parenthésées au moment de la compilation, le préprocesseur permet de compléter les parenthèse manquante.

2 Fonctions du préprocesseur

Afin de faciliter au maximum le travail du compilateur, le préprocesseur rempli de nombreuses fonctions simultanément.

2.1 Détection d'erreur

Il est facile de faire des erreurs en programant, mais pas toujours de les trouver. Le simple fait de chercher une parenthèse mal fermée peut représenter une perte de temps importante. Par exemple dans le segment de code suivant, on voit qu'une parenthèse fermante manque.

```
1  local x = print(test(i)
```

Exemple 1 – Mauvais parenthèsage

Un des rôles du préprocesseur va être d'avertir l'usager de son erreur. Si on tente d'exécuter luna sur le code suivant, on obtient ceci :

```
FILE: tmp.lua
Error at line 1: Parenthesis mismatch.
local x = print(test(i)
                ^
```