COMPILATION
EMSI - 4ÈME IIR
2017/2018

Prof. M. D. RAHMANI

Plan

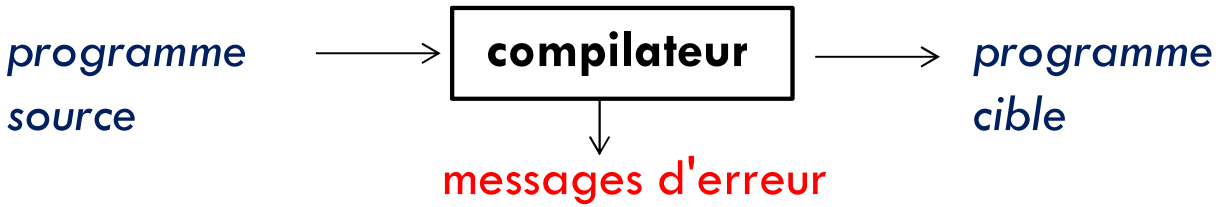
1. L'analyse lexicale : implantation d'un analyseur lexical :
   1. Manuelle avec le langage **C, C++**
   2. Automatique avec le langage ***Flex,***
2. L'analyse syntaxique descendante,
3. Implantation manuelle d'un analyseur avec les langages **C, C++,**
4. L'analyse syntaxique **ascendante,**
5. Implantation automatique d'un analyseur avec le langage ***Bison,***
6. L'analyse sémantique (*portée des variables, graphes acycliques,...*)
7. Conception d'un compilateur complet avec les langages ***Flex*** et ***Bison.***

1- Les compilateurs (1)

Définition:

*Un compilateur est un programme qui **lit** un programme écrit dans un premier langage (le langage source) et le **traduit** en un programme équivalent dans un autre langage (le langage cible).*

Le compilateur doit aussi vérifier que le programme a un certain sens et signaler les erreurs qu'il détecte.



1- Les compilateurs (2)

Il y'a deux parties dans la compilation: l'***analyse*** et la ***synthèse***.

- ***La partie analyse*** partitionne le programme source en ses constituants et en crée une représentation intermédiaire.

- ***La partie synthèse*** construit le programme cible à partir de cette représentation intermédiaire.

2- Les phases d'analyse

L'analyse est constituée de 3 parties:

1. **L'analyse lexicale** (linéaire): le flot de caractères formant le programme source est **lu** de gauche à droite et **groupé** en *lexèmes* (*mots*), qui sont des suites de caractères ayant une signification collective (production d'un diagramme de transition).
2. **L'analyse syntaxique** (grammaticale): les unités lexicales sont regroupés hiérarchiquement dans des collections imbriquées (*phrases*) ayant une signification collective (production d'un arbre syntaxique).
3. **L'analyse sémantique**: contrôle pour s'assurer que l'assemblage des constituants du programme a un sens (arbre syntaxique décoré).

3- Les phases de synthèse

La synthèse est constituée de 3 parties:

1. **Code intermédiaire** : production d'un arbre abstrait et d'un code associé à 3 adresses.
2. **Optimisation du code** : amélioration du code intermédiaire pour que le code final s'exécute plus rapidement et utilise le minimum de mémoire.
3. **Production du code**: production du code en langage de la machine cible.

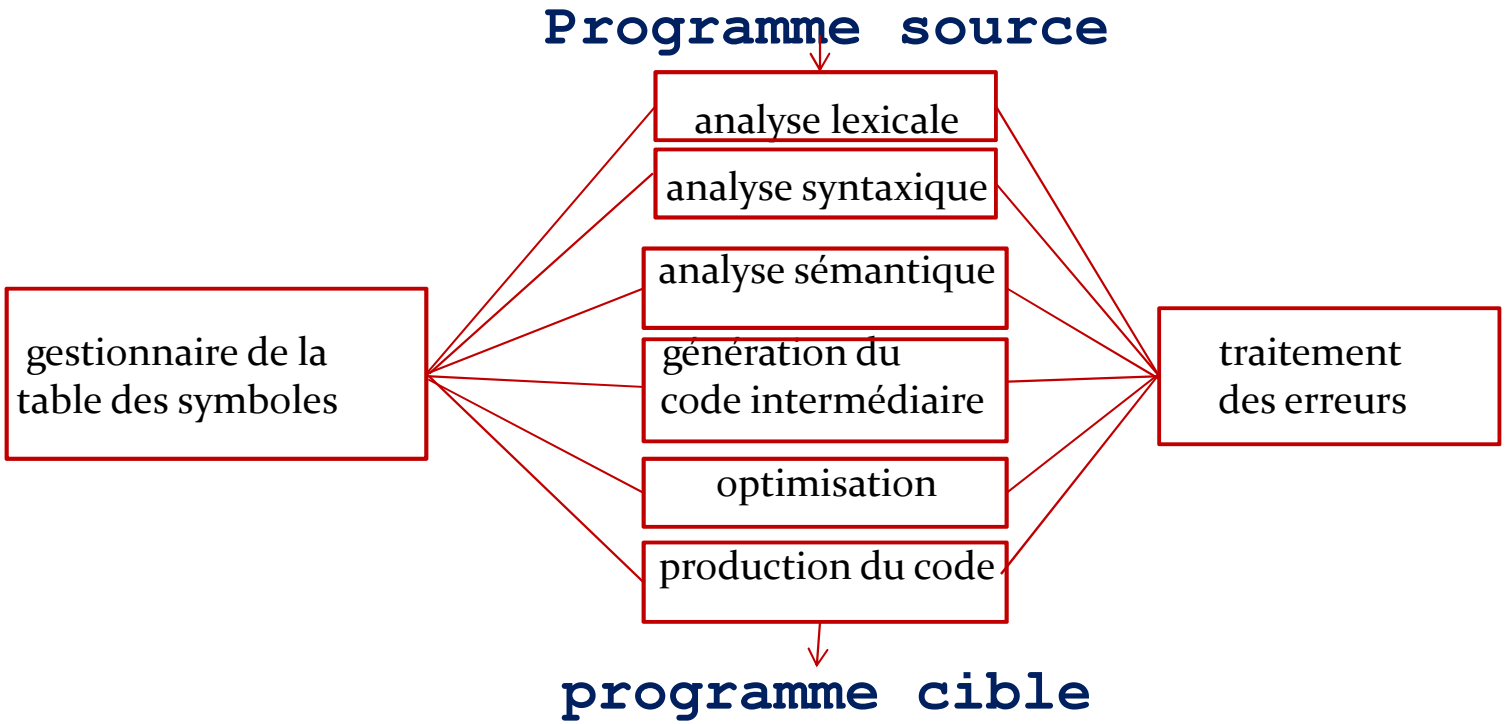
4- Deux phases supplémentaires

1. **La création de la table des symboles:** c'est une structure de données contenant un enregistrement pour chaque identificateur, muni de champs pour ses attributs (*emplacement mémoire, son type, sa portée..*).
   Pour les fonctions, la table des symboles garde le *nombre* et *les types de ses arguments*, *le mode de passage de chacun d'eux et la valeur de retour*.

2. **Le traitement des erreurs** associées à chacune de ces phases.

Après avoir détecté une erreur, une phase doit la traiter de telle façon que la compilation puisse continuer et que d'autres erreurs dans le programme puissent être détectées.

5-Schéma logique d'un compilateur



6-Les phases et les outils utilisés

Les différentes phases

Phases d'analyse :
- analyse lexicale
- analyse syntaxique
- analyse sémantique

Phases de synthèse:
- code intermédiaire
- optimisation
- production du code

Traitements parallèles:
- table des symboles
- traitement des erreurs

Les outils utilisés

- expression régulière
- automate à états finis
- grammaires
- automate à pile
- traduction dirigée par syntaxe
- traduction dirigée par la syntaxe

7- Les interprètes

Au lieu de produire un programme cible, un interprète effectue lui même les opérations spécifiées par le programme source.

Un **interprète** est un outil ayant pour tâche d'**analyser**, de **traduire** et d'**exécuter** les programmes.

On utilise souvent des interprètes pour exécuter les ***langages de commandes***.

Exemples:

- Le *shell* d'Unix
- Les langages de script: PHP, Perl

Certains langages ont deux versions, une **compilée** et une autre **interprétée**.

Exemples: caml, lisp, scala

8- Les 3 grandes catégories de langages

1- Les langages de programmation permettent de créer des programmes, des applications mobiles, des sites Internet, des systèmes d'exploitation, etc...

Les 10 langages de programmation les plus populaires (en 2012):

C, Java, C++, Objective-C, C#, PHP, Basic, Python, Perl, JavaScript.

2- Les langages de description permettent de décrire et de structurer un ensemble de données selon un jeu de règles et contraintes définies.

Exemples : SGML, XML, HTML.

3- Les langages de requêtes permettent d'interroger des structures qui contiennent des données.

Exemples : SQL pour les BD relationnelles, OWL pour les ontologies, XQuery pour les documents XML.

9- Le classement des langages (1)

IEEE : top 10 des meilleurs langages de programmation de l'année 2016

9- Le classement des langages (2)

C est le *meilleur* langage de programmation pour cette année, il était *deuxième en 2014 et 2015*, à chaque fois derrière Java. Java descend donc à *la deuxième place*.

Par rapport au top 5 de l'année dernière, Python (4e en 2015) est monté d'une marche au détriment de C++ qui passe alors de la 3e place en 2015 à la 4e place cette année.

C# sort du top 5 pour occuper la *6e place*, alors qu'on assiste à l'entrée du langage R dans le big five. R était 7e dans le classement de 2015 et 9e dans celui de 2014, soit un bond de 4 places pour le langage de calcul statistique. Ce qui pourrait refléter la croissance du Big data, d'après l'IEEE.

Les autres langages : Swift (#11), Arduino (#12), Assembly (#13), Matlab (#14), Scala (#15), HTML (#16), Perl (#17), Visual Basic (#18), Shell (#19) et Objective C (#20). Rust, le langage de programmation de Mozilla vient à la 26e place devant Delphi en 28e position.

Bibliographie:

Livres de référence:

1- A. Aho, M. Lam, R. Sethi et J. Ullman,

"*Compilateurs: principe, techniques et outils*", Pearson Education.

2- J. Menu, " *Compilateurs avec C++*", Addison - Wesley.

3- R. Wilhelm et D. Maurer, *"Les compilateurs: théorie, construction, génération"*, Masson

4- J.E.F. Friedel, "*Maîtrise des expressions régulières*", O'Reilly

Webographie:

1- Compilateurs avec C++, Jacques Menu

http://cui.unige.ch/~menu/CompilateursAvecC++.pdf

2- Cours de techniques de compilation, J.Bonneville Documentation Lex-Flex

http://users.polytech.unice.fr/~dedale/cours/compilation/Lex-HowTo/

3- Cours de compilation, Luc Maranget

http://www.enseignement.polytechnique.fr/profs/informatique/Luc.Maranget/compil/poly/index.html

4- Cours de compilation, ENS de Lyon, Christophe Alias, cours, TD, TP, examens et partiels

http://perso.ens-lyon.fr/christophe.alias/compilation_ens.html

Webographie:

5- Techniques et outils pour la compilation, H. Garreta
Faculté des Sciences de Luminy - Université de la Méditerranée
http://henri.garreta.perso.luminy.univmed.fr/Polys/PolyCompil.pdf

6- Introduction à la compilation, Yann Régis-Gianas
Université Denis Diderot – Paris 7
http://www.pps.univ-paris-diderot.fr/~yrg/compil/compilation-slides-cours-1.pdf

7- Compilation, théorie des langages, université de Bretagne occidentale
http://www.lisyc.univ-brest.fr/pages_perso/leparc/Etud/Master/Compil/Doc/CoursCompilation.pdf

8- Cours de compilation, J. Ferber
http://www.lirmm.fr/~ferber/Compilation/compil1.htm

9- Cours sur Flex et Bison, John Levine, O'reilly
http://web.iitd.ac.in/~sumeet/flex__bison.pdf

Outils

Flex et *bison*: http://sourceforge.net/projects/winflexbison/

Nous aurons un fichier compressé: **`win_flex_bison-latest`** à décompresser.

ou http://sourceforge.net/projects/gnuwin32/files/bison/2.4.1/bison-2.4.1-setup.exe/download

JFlex: logiciel http://jflex.de/ , manuel: http://jflex.de/manual.html

et *Cup*: http://www2.cs.tum.edu/projects/cup/

Dev-C++: http://www.commentcamarche.net/download/telecharger-59-dev-c

Nous aurons un fichier exécutable: **`Dev-Cpp_5.9.2_TDM-GCC_4.8.1_Setup`**

Geany: http://www.geany.org/ , manuel: http://www.geany.org/Documentation/Manual