

# Introduction

Cyril Rabat

`cyril.rabat@univ-reims.fr`

Licence 3 Informatique - Info0602 - Langages et compilation

2020-2021



## Cours n°1

*Qu'est qu'un compilateur ?  
Les phases de la compilation*

Version 11 décembre 2020

# Table des matières

## 1 Introduction à la compilation

- Introduction
- La partie analyse
- La partie synthèse
- Les tâches transversales

# Table des matières

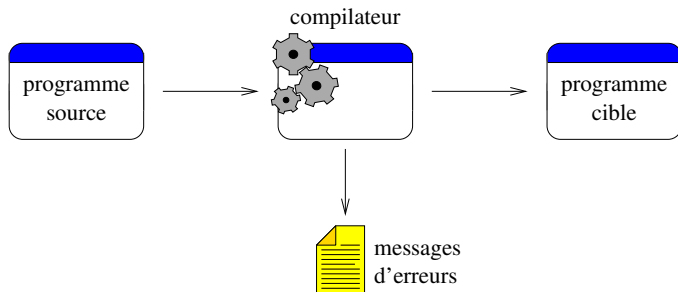
## 1 Introduction à la compilation

- Introduction
- La partie analyse
- La partie synthèse
- Les tâches transversales

# Un compilateur

## Définition : un compilateur

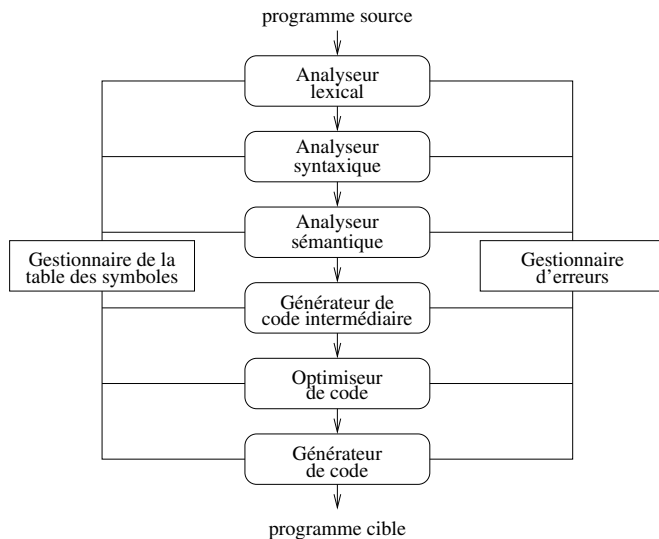
*Programme qui lit un programme écrit dans un premier langage (le **langage source**) et le traduit en un programme équivalent écrit dans un autre langage (le **langage cible**).*



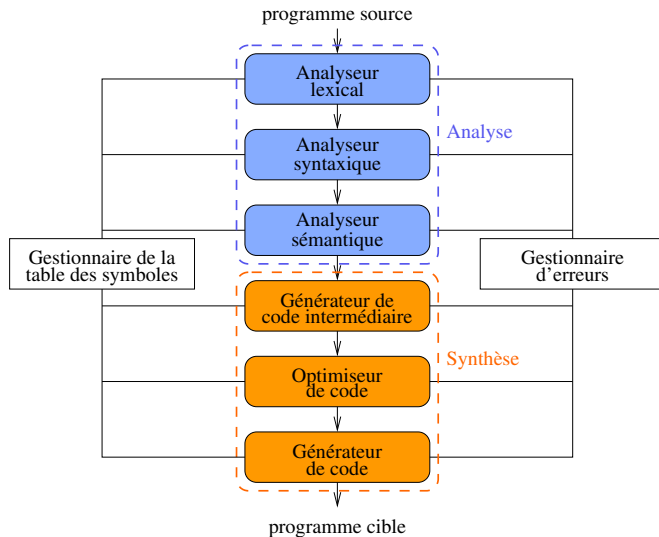
# Les concepts

- La compilation fait appel à :
  - La théorie des langages
  - L'architecture des machines
  - L'algorithmique
  - Le génie logiciel
- Ces concepts sont utilisés dans bien d'autres domaines :
  - Le traitement de texte
  - L'interpréteur de commandes / requêtes

# Les phases d'un compilateur



# Les phases d'un compilateur



# Les deux parties de la compilation

## Définition : analyse

- *Partitionnement du programme source en constituants*
- *Création d'une représentation intermédiaire*

## Définition : synthèse

- *Construction du programme cible à partir de la représentation intermédiaire*

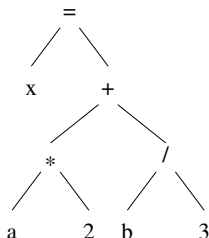


# La représentation intermédiaire

- Pendant l'analyse, les opérations du programme source sont récupérées et conservées
- Pour cela, on utilise une structure hiérarchique : les arbres
- Exemple d'arbre utilisé : l'arbre abstrait

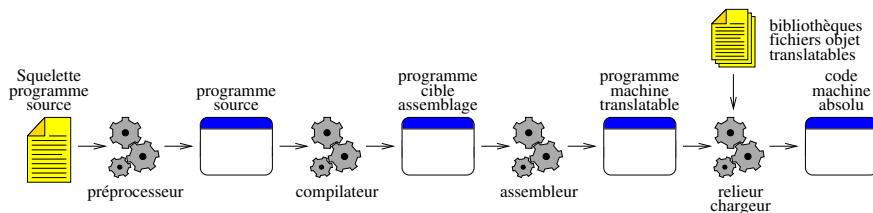
## Exemple

- Instruction :  $x = a * 2 + b / 3$



# Environnement du compilateur

- Pour créer le programme cible :
    - ↪ Compilateur + autres programmes
  - Il est nécessaire d'assembler différentes sources :
    - Plusieurs fichiers sources
    - Utilisation de macros, *etc.*
- ↪ Réalisé par le préprocesseur



# Table des matières

## 1 Introduction à la compilation

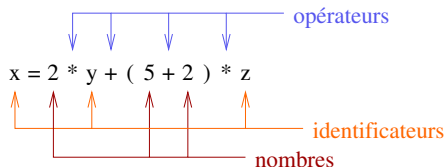
- Introduction
- **La partie analyse**
- La partie synthèse
- Les tâches transversales

# Les différentes phases de l'analyse

- ➊ Analyse linéaire ou **analyse lexicale**
- ➋ Analyse hiérarchique ou **syntaxique** ou **grammaticale**
- ➌ Analyse sémantique

# Analyse lexicale

- Son but est de reconnaître les **unités lexicales** (ou **lexèmes**)
- Exemples :
  - Les identificateurs et mots-clés du langage
  - Les opérateurs
  - Les valeurs

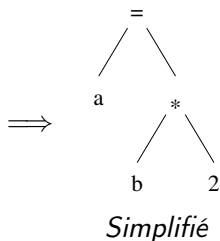
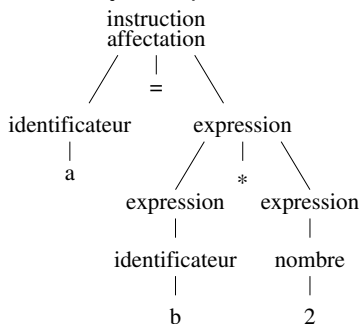


# Analyse syntaxique (1/2)

- L'analyse syntaxique est appelée aussi **analyse grammaticale**
- Son but est de regrouper les unités lexicales en structures grammaticales
  - ↪ Généralement, elle génère des arbres syntaxiques
- Les structures grammaticales sont généralement hiérarchiques
  - ↪ Elles sont exprimées par des règles récursives
  - ↪ Exemple : on utilise des **grammaires**

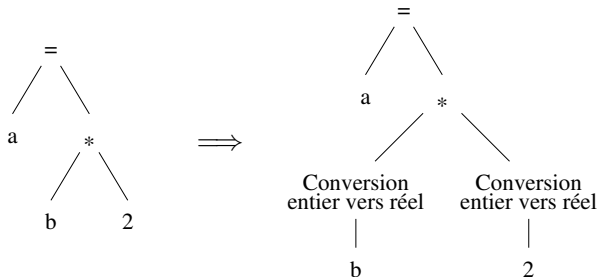
## Analyse syntaxique (2/2)

- Exemple de règles pour une expression :
  - Tout *identificateur* est une expression
  - Tout *nombre* est une expression
  - Si  $exp_1$  et  $exp_2$  sont des expressions, il en est de même pour :
    - $exp_1 + exp_2$
    - $exp_1 * exp_2$
    - $(exp_1)$
- Exemple d'arbre syntaxique de  $a = b * 2$  :



# Analyse sémantique

- Son but est de vérifier si le programme source contient des erreurs sémantiques  
 ⇨ Exemple : la vérification des types
- Elle exploite l'arbre syntaxique
- Exemple avec l'instruction précédente, si  $a$  est un réel et  $b$  un entier :





# Table des matières

## 1 Introduction à la compilation

- Introduction
- La partie analyse
- La partie synthèse
- Les tâches transversales

# Les différentes phases de l'analyse

- La génération de code intermédiaire
- L'optimisation de code
- La génération de code

# La génération de code intermédiaire

- Son but est de construire une représentation intermédiaire du programme source
- Deux propriétés pour cette représentation :
  - Elle doit être facile à produire
  - Elle doit être facile à traduire
- Exemple : le code à 3 adresses  
↪ C'est une séquence d'instructions dont chacune possède au plus 3 opérandes
- Elle nécessite :
  - D'utiliser des variables intermédiaires
  - De faire un choix sur l'ordre de calcul
- Exemple :  $x = y + 2 * z$  (avec  $id1 = x$ ,  $id2 = y$  et  $id3 = z$ )  
 $tmp1 = 2$   
 $tmp2 = id3 * tmp1$   
 $tmp3 = id2 + tmp2$   
 $id1 = tmp3$

# L'optimisation de code

- Son but est d'améliorer le code intermédiaire :
  - Réduction du nombre de variables
  - Réduction du nombre d'instructions
- C'est l'une des phases les plus gourmandes en temps
- Exemple :

```
tmp1 = 2
```

```
tmp2 = id3 * tmp1
```

```
tmp3 = id2 + tmp2
```

```
id1 = tmp3
```

*Avant*

```
tmp1 = id3 * 2
```

```
id1 = id2 + tmp1
```

*Après*

# La génération de code

- Son but est de produire du code machine translatable ou du code en langage d'assemblage
- Des emplacements mémoires sont choisis pour les variables
- La traduction des instructions du code intermédiaire en instructions machine nécessite d'assigner les variables aux registres
- Exemple :

```
MOVF id3, R2  
MULF #2.0, R2  
MOVF id2, R1  
ADDF R2, R1  
MOVF R1, id1
```

# Table des matières

## 1 Introduction à la compilation

- Introduction
- La partie analyse
- La partie synthèse
- Les tâches transversales

# La table des symboles

- Lors de la compilation, il est nécessaire de collecter les informations (attributs) sur les identificateurs
  - ↪ Le type, la valeur, l'emplacement mémoire, la portée
- La **table de symboles** est une structure de données contenant un champ pour chaque identificateur
- Les champs sont créés lors de l'analyse lexicale
- Les attributs sont ajoutés lors des phases suivantes
- La table permet aux différentes phases de trouver les informations nécessaires
  - ↪ Exemple : le type lors de l'analyse sémantique

# La gestion des erreurs

- Lors de chaque phase, des erreurs peuvent être générées
- Objectif de la gestion d'erreurs : traiter l'erreur pour poursuivre autant que possible
  - ↪ Il faut éviter l'arrêt dès la première erreur !
- Exemple d'erreurs :
  - Analyse lexicale : flot de caractères non reconnu
  - Analyse syntaxique : construction incorrecte
  - Analyse sémantique : type incorrect