

Compilation

Mirabelle Nebut

Bureau 332 - M3
mirabelle.nebut at lifl.fr

2011-2012

Organisation du cours

Organisation : C / TD / TP sur 12 semaines

Évaluation :

- ▶ contrôles courts en amphi au cours du semestre ;
- ▶ TP rendus ;
- ▶ contrôle de 3h en fin de semestre.

Docs et infos là (**mais ne dispensent pas d'assister au cours**) :

<http://www.fil.univ-lille1.fr/portail/>

Organisation première semaine

- ▶ 2 cours :
 - ▶ aujourd'hui = créneau additionnel
 - ▶ jeudi, 8h30 amphi M3 = créneau habituel
- ▶ 1 TD = créneaux particuliers à cette semaine
- ▶ 0 TP

Les intervenants

- ▶ Groupe 1 : Cédric Lhoussaine
- ▶ Groupe 2 : Mirabelle Nebut
- ▶ Groupe 3 : Mirabelle Nebut
- ▶ Groupe 4 : Thomas Pietrzak

Le pourquoi du cours de COMPIL(ation)

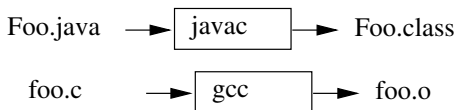
Outils pour la compilation

Contenu du cours

Bibliographie

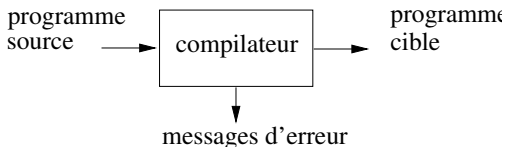
Quels compilateurs connaissez-vous ?

Compilateurs *utilisés* quotidiennement en L3



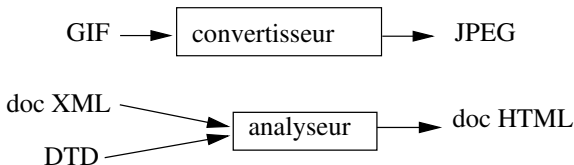
Dans ce cas un **logiciel** qui produit un **exécutable** à partir d'un programme :

- ▶ entrée : programme dans le **langage source** ;
- ▶ sortie : programme (équivalent) dans le **langage cible** ;
- ▶ ou message(s) d'erreur si entrée non correcte.



D'autres exemples ?

Compilateurs *utilisés* couramment



Dans ce cas un **logiciel** qui **transforme** une **entrée textuelle** en sortie équivalente :

- ▶ source et cible pas nécessairement des programmes ;
- ▶ cible pas nécessairement exécutable ;
- ▶ conservation de l'information ;
- ▶ **correction** de l'entrée.

Pensez-vous que vous serez amenés à écrire / développer de tels compilateurs ?

P'êtré ben que non. . .

Parmi vous, peu seront amenés à travailler sur un compilateur pour Java !

Vous ne faites que l'utiliser.

(mais alors, pourquoi ce cours ?)

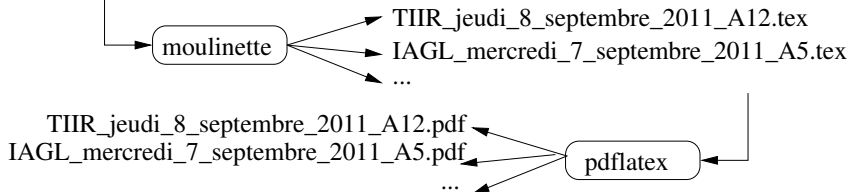
Quel type de compilateur *écrirez-vous* au quotidien ?

Une « **moulinette** » qui prend en entrée un fichier texte **correct** (de données), le **reconnaît** et le **traite**...

- ▶ fichier de configuration d'une application
- ▶ reverse engineering de copybook COBOL vers votre langage préféré
- ▶ extraction automatique de doc (graphe des appels) d'une application
- ▶ et vos moulinettes perso...

Exemple en TP : génération de planning format latex

```
master TIIR
date jeudi 8 septembre 2011
salle A12
Dupont Alex ; Crédit Coop ; J2EE, Struts
Durand Sophie ; La Nef ; modélisation, génération
...
master IAGL
date mercredi 7 septembre 2011
...
```



Exemple en TP : un DSL

DSL = Domain-Specific Language

[Fowler2011] *a DSL is a computer programming language of limited expressiveness focused on a particular domain.*

Autres exemples de DSL de la vraie vie :

- ▶ langage Dot + graphviz : visualisation de graphes ;
- ▶ Mockito : description de mocks pour Java ;
- ▶ CSS ;
- ▶ langage de Makefile pour make

Le livre de Fowler sur les DSL

Utilise le mot « compilateur » au sens « pas un interpréteur ».

Mais les techniques associées aux DSL sont celles de la compilation.

600 pages sur des « *little languages which can help clarify small, but important, areas of a software project* », par un grand monsieur de l'OO et des méthodes agiles.

En bref, quels logiciels étudie-t-on en COMPIL ?

Qu'on l'appelle DSL ou compilateur : c'est un **logiciel** qui

- ▶ prend en **entrée** une **donnée textuelle source** (programme, donnée xml, fichier de configuration, etc) ;
- ▶ la **reconnaît** (l'**analyse**) pour vérifier sa **correction** ;
- ▶ émet éventuellement un message d'erreur ;
- ▶ **calcule** une donnée de sortie (programme, donnée, etc).

Le pourquoi du cours de COMPIL(ation)

Outils pour la compilation

Contenu du cours

Bibliographie

Outils pour la compilation

Génie logiciel

Théorie du langage

À quoi sert le génie logiciel (en compilation)

patterns de **conception objet** standard

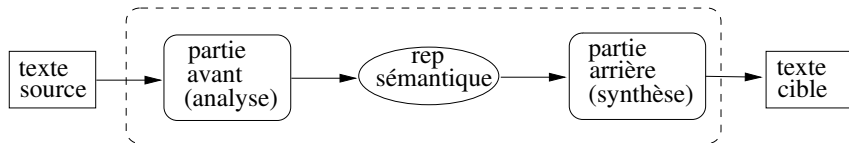
structures de données

structuration classique d'un compilateur en **modules**

Structure globale

En deux parties :

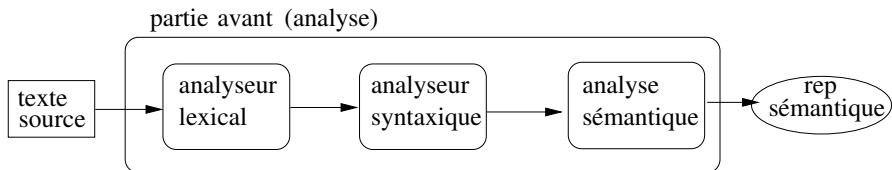
- ▶ analyse/reconnaissance ;
- ▶ synthèse/transformation.



Structure classique d'une application de traitement de données textuelles.

À quoi sert la théorie du langage (en compilation)

Essentielle pour la partie reconnaissance :



Analyse lexicale

- ▶ seul module au contact avec le texte source ;
- ▶ lit le texte source sous la forme d'une **suite de caractères** ;
- ▶ décompose cette suite en une suite d'**unités lexicales** appelées **symboles** ou **tokens** ;

```
Ex : program pgm;      →   PROGRAM IDENT("pgm")  
    int x,y;           →   FININSTR DECL IDENT("x")  
                        →   SEP IDENT("y") FININSTR
```

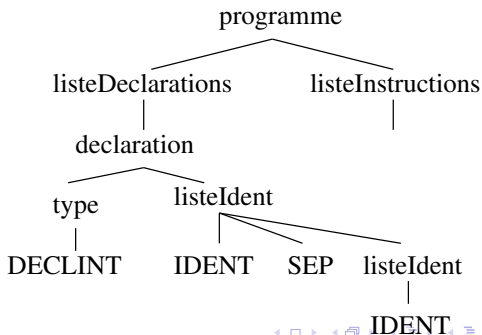
Analyse syntaxique

- ▶ connaît la **syntaxe** des textes corrects :
- ▶ tente de **reconnaître** dans le flot des symboles la **structure** d'un texte correct
- ▶ cette structure peut se décrire par un **arbre syntaxique**

Ex :

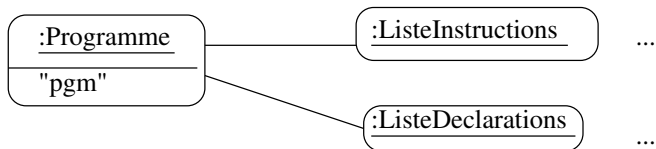
PROGRAM
IDENT("pgm")
FININSTR DECL
IDENT("x")
SEP IDENT("y")
FININSTR

→



Analyse sémantique

Produit une **représentation sémantique interne** du source.



Vérifie certaines **propriétés** dites **statiques** (= à la compilation, par opposition à dynamique = à l'exécution).

- ▶ vérification de typage ;
- ▶ vérification des déclarations ;

Descriptions et formalismes exécutables

Théorie du langage.

module	description	formalisme exécutable
analyse lexicale	expressions régulières	automates à nombre fini d'états
analyse syntaxique	grammaires algébriques	automates à pile
analyse sémantique	grammaires attribuées	automates à pile avec actions

+ arbre syntaxique = structure sous-jacente à l'analyse syntaxique

Le pourquoi du cours de COMPIL(ation)

Outils pour la compilation

Contenu du cours

Bibliographie

En TP

Plusieurs cas d'études, orientés DSL externes.

Illustration des différentes techniques vues en cours

Premier TP : génération de planning Latex **à la main**

TPs suivants : utilisation de générateurs automatiques d'analyseurs lexical et syntaxique

- ▶ comprendre leur expressivité : quand peut-on les utiliser
- ▶ comprendre les messages d'erreur
- ▶ comprendre les contraintes d'utilisation

NB : **impératif d'avoir compris la théorie du langage** vue en cours

En cours et TD

Notions de théorie du langage principalement :

- ▶ analyse lexicale : techniques et outils ;
- ▶ analyse syntaxique :
 - ▶ grammaires algébriques ;
 - ▶ automates à pile, automate des items ;
 - ▶ analyse descendante LL(1) ;
 - ▶ analyse ascendante ;
- ▶ analyse sémantique : grammaires attribuées

Bibliographie

Bibliographie complète sur le portail :

- ▶ "Le dragon" : Aho, Sethi, Ullman,
Compilateurs : principes, techniques et outils ;
- ▶ un ouvrage très pragmatique : Grune, Bal, Jacobs,
Langendoen,
Compilateurs ;
- ▶ un ouvrage plus formel, très rigoureux : Wilhelm, Maurer,
Les compilateurs : théorie, construction, génération ;
- ▶ un ouvrage très pragmatique, orienté Java : Appel,
Modern Compiler Implementation in Java.
- ▶ aucune théorie du langage : Fowler,
Domain-Specific Languages