

Théorie des langages (suite)

Avertissements

Ce document contient des prises de notes de cours, mais sont très incomplètes ! LA qualité de ce cours est très inférieure à ce qui était attendu, préférant passer le temps du cours à jouer à Chrono Trigger, au lieu de suivre sérieusement. Pour plus d'informations, il vaudrait mieux se référer au polycopié, disponible à la reprographie. Je tacherais de compléter si j'ai l'occasion de retrouver le cours.

1 Langages intermédiaires (Pseudo-Code)

Il existe une sorte de pseudo-code interprété, par exemple **Java** ou encore **Python**. Les avantages sont :

- la rapidité d'exécution,
- la flexibilité,
- la puissance

2 Bootstrap

Le concept de Bootstrap cherche à résoudre le problème de comment écrire un compilateur dans le langage de programmation cible qu'il doit compiler, par exemple, comment écrire un compilateur java, lui-même écrit en java ?

Terminologie :

- **Build** : ordinateur qui compile
- **Host** : ordinateur qui exécutera
- **Target** : ordinateur pour lequel le logiciel produit des données

3 Compilation native

Principe de compilation "au vol"

- Bytecode : code natif au chargement ou à la première exécution
- Peut produire plusieurs alternatives natives d'un même code
- Gain de temps

JVM est de type "token threaded code"

Arbres syntaxiques pour représenter les structures de contrôle des langages.

GCC utilise un langage intermédiaire

Intel 32 bits : architecture CISC

exemple de code

```
a = a + b * c;
```

donnera en assembleur x86 :

```
movl b, %eax  
imull c, %eax  
addl %eax, a
```