Cours de compilation

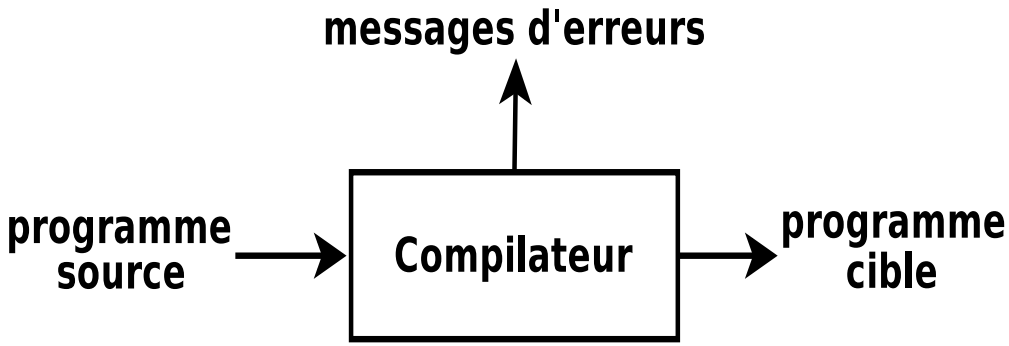
Lakhdar Sais

Compilation

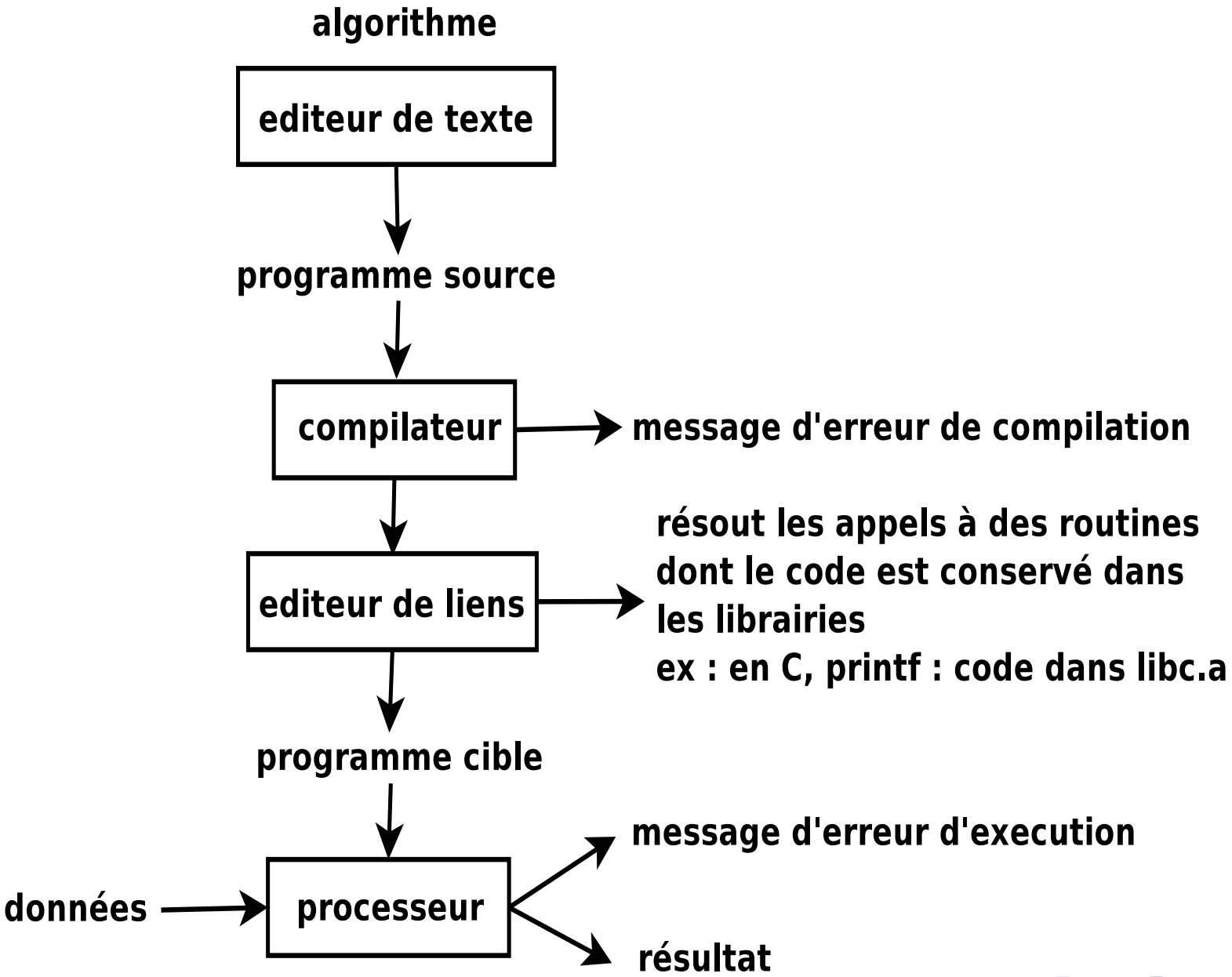
5 septembre 2016

Un compilateur ?

Un compilateur est un logiciel qui traduit un programme écrit dans un langage de haut niveau (par le programmeur) en instructions exécutables (par un ordinateur) .



Chaîne de développement d'un programme



Compilateur $\neq$ interpréteur

- Compilateur : Un compilateur traduit un programme source P en un programme écrit dans un langage cible P'.
  (Le fichier résultat est directement exécutable par la machine)

Exemple

Pascal, C, C++, ADA, Fortran, Cobol, ...

Compilateur $\neq$ interpréteur

- Interpréteur :
  - Au lieu de produire un programme cible comme dans le cas d'un compilateur, un interpréteur exécute lui-même au fur et à mesure les opérations spécifiées par le programme source. Il analyse une instruction après l'autre puis l'exécute immédiatement.
  - Un interpréteur doit être présent dans le système à chaque fois que le programme est exécuté.
  - Il analyse une instruction après l'autre et l'éxécute.
    (on ne peut pas cacher le code ou le garder secret)
  - Il travaille simultanément sur le programme et les données.
  - Il est plus petit en taille mais plus lent et tolère plus d'erreurs.

Exemple

BASIC, Perl, Prolog, Scheme, CaML, Tcl, Lisp, ...

Langage P-Code

Il existe des langages qui sont à mi-chemin de l'interprétation et de la compilation : langage P-code

- Pseudo-code (P-Code ou Langage intermédiaire)
  Code source qui est traduit (compilé) dans un langage cible qui n'est pas encore un code machine (du pseudo code ou P-code). Pas encore exécutable par la machine.

  Lorsque l'on exécute le programme, ce P-code est ensuite interprété.

Exemple

- Java (traduction en "byte code" (.class) interprété par une machine virtuelle)
- Autre langage P-code : Python
- Les interpréteurs de P-code peuvent être relativement petits et rapides (peut s'exécuter aussi rapidement que du binaire compilé).

Pourquoi ce cours ?

1. Avec les connaissances de ce cours, on peut mieux comprendre les contraintes d'un programme.
2. On peut se servir des outils présentés pour faire :
   - la traduction d'un langage de bas niveau a un langage de haut niveau (de Pascal à C, de Java à C++)
   - la traduction du code machine en programme de haut niveau (exemple : pour récupérer le programme d'un vieux logiciel, piratage de vieux logiciels, ...)
   - la traduction d'un programme quelconque (.doc .html, .ps .pdf,...)
3. Les idées et techniques sont générales et fondamentales. Un informaticien les utilisera très souvent (traitement des données, moteur de recherche, ...)

Contenu de ce cours

- Les principes de base inhérents à la réalisation de compilateurs : analyse lexicale, analyse syntaxique, analyse sémantique et génération de code.
- Les outils fondamentaux utilisés pour effectuer ces analyses : fondements de base de la théorie des langages (grammaires, automates, ...), méthodes algorithmiques d'analyse, ...

Qu'est ce qu'on attend d'un compilateur ?

- Détecter et identifier les erreurs de compilation : erreurs statiques (ne nécessitent pas l'execution du programme) et les prévenir d'une manière efficace à l'utilisateur.
- Efficace en temps : produire un programme qui s'execute rapidement
- Correction : le calcul décrit dans le langage source doit être effectivement celui executé par la machine.
- Modularité : compilation séparée d'un gros développement (chaque module peut-être compilé sans connaitre l'implantation de toutes les fonctions qu'il utilise)

Bibliographie

- A. Aho, R. Sethi, J. Ullman, Compilateurs : principes, techniques et outils, InterEditions 1991.
- N. Silverio, Réaliser un compilateur, Eyrolles 1995
- J. Levine, T. Masson, D. Brown, Lex & Yacc, Editions O'Reilly International Thomson 1995
- ...