المدرسة الحسنية للأشغال العمومية
ECOLE HASSANIA DES TRAVAUX PUBLICS

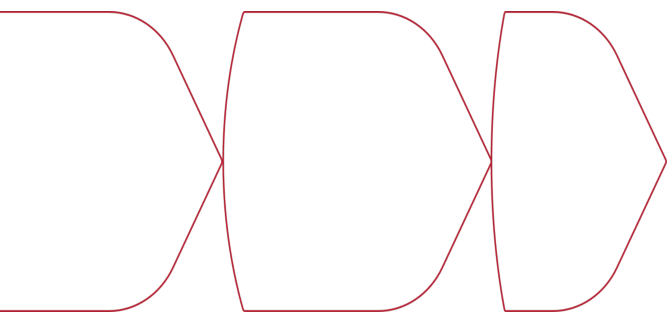
Introduction à la Théorie de la Compilation

Comprendre les bases de la création de compilateurs

Slimani Mohamed Amine

EHTP

February 7, 2025



Sommaire

Qu'est-ce que la compilation ?

- **Définition** : La compilation est le processus de traduction d'un programme écrit dans un langage de haut niveau en un langage de bas niveau (ex : assembleur, code machine).
- **Objectif** : Produire un programme exécutable à partir du code source.
- **Phases de la compilation** : Analyse lexicale, analyse syntaxique, analyse sémantique, génération de code, optimisation.

Pourquoi la théorie de la compilation est-elle importante ?

- **Compréhension des langages** : Comprendre comment les langages de programmation fonctionnent.
- **Optimisation** : Améliorer les performances des programmes.
- **Création de compilateurs** : Développer des compilateurs pour de nouveaux langages.

Phases de la compilation

- **Analyse lexicale** : Convertir le code source en une séquence de tokens.
- **Analyse syntaxique** : Vérifier la structure syntaxique du programme.
- **Analyse sémantique** : Vérifier la signification du programme.
- **Génération de code** : Produire du code de bas niveau.
- **Optimisation** : Améliorer l'efficacité du code généré.

Outils et technologies

- **Lex** : Générateur d'analyseurs lexicaux.
- **Yacc** : Générateur d'analyseurs syntaxiques.
- **LLVM** : Infrastructure pour la génération de code et l'optimisation.

Exemple d'analyse lexicale

Code source et tokens

```
int main() {
        return 0;
}
```

```
KEYWORD(int) IDENTIFIER(main) SYMBOL(() SYMBOL()) SYMBOL({)
↪  KEYWORD(return) NUMBER(0) SYMBOL(;) SYMBOL(})
```

Exemple d'analyse syntaxique

Arbre syntaxique

```
FunctionDeclaration
|---Type: int
|---Name: main
|--- Parameters: ()
|--- Body:
     |---ReturnStatement
         |--- Expression: 0
```

Bonnes pratiques

- **Modularité** : Diviser le compilateur en phases distinctes.
- **Gestion des erreurs** : Fournir des messages d'erreur clairs et informatifs.
- **Optimisation** : Utiliser des techniques d'optimisation pour améliorer les performances.

Défis de la compilation

- **Complexité** : Gérer la complexité des langages modernes.
- **Performance** : Produire du code optimisé.
- **Portabilité** : Générer du code pour différentes architectures.

Pourquoi c'est important ?

- La théorie de la compilation est essentielle pour comprendre comment les langages de programmation sont transformés en code exécutable.
- Elle permet d'optimiser les performances des programmes et de créer de nouveaux compilateurs.
- Comprendre la théorie de la compilation est crucial pour les développeurs de compilateurs et les concepteurs de langages.

Résumé

La théorie de la compilation est une discipline clé pour la création de compilateurs et l'optimisation des programmes. Explorez, apprenez, et innovez avec la compilation !