Résumé

Le compilateur

STRUCTURE DE FICHIERS

Le groupe compilation a pour objectif de créer un ensemble de fichiers bison et lex. Ces fichiers sont alors tries dans un ensemble de structure de fichiers.

Cette structure de fichiers doit être stables, car les différents fichiers qui vont être rangés ainsi que le makefile vont se reposer sur cette structure de fichiers

Cette structure de fichiers doit donc rendre compte de plusieurs critères :

- Séparer les fichiers lex, les fichiers bisons, les exécutables et les fichiers intermédiaires
- Permettre au makefile de compiler tous nouveaux fichiers rapidement (automatique ou rajout d'une ligne dans le makefile

MAKEFILE

Le makefile permet en une simple ligne de commande de compiler les fichiers, de composer les étapes intermédiaires si nécessaires et de créer l'exécutable à la fin.

Le makefile doit aussi être un des premiers paliers d'interface avec l'utilisateur, et doit alors être relativement exhaustif dans :

- La gestion des erreurs (remontée des erreurs défini par le groupe compilation)
- La gestion de l'aide sur les options (un --help qui soit complet)
- Les différents types de compilation -> compilation de débug, de release, d'optimisation …

LA COMPILATION

Le lancement de la compilation se passe avec le makefile. Pour cela, deux pistes sont possibles :

- Une interface qui simplifie (et augmente) les possibilités du compilateur c++. Il se passe en ligne de commande et permet d'utiliser les différentes possibilités comme la redirection de la sortie, l'ajout de bibliothèque

  Il est alors généré un ensemble de fichier C++ valides. Ces fichiers peuvent alors eux-aussi être compilés (une autre interface ? Lancer le compilateur normalement ?) pour donner un exécutable.

Options en ligne de commande

COMMENT GERER LES OPTIONS

L'objectif des options de compilation est de pouvoir aiguiller le compilateur sur les différentes actions qu'il exécute. Ces options se gèrent normalement directement par le makefile, ou directement ou en passant par un script de traitement externe.

Les seules options qui pourront avoir une communication avec l'intérieur du code sont :

- L'action de certaines options de code (peut-on ajouter du code C++ dans le code)
- La gestion des arguments en ligne de commande
- Les options de verboses, qui activent certaines parties qui affichent alors plus d'informations
- Les bibliothèques incluent dans le code qui doivent être reliées à une option de compilation

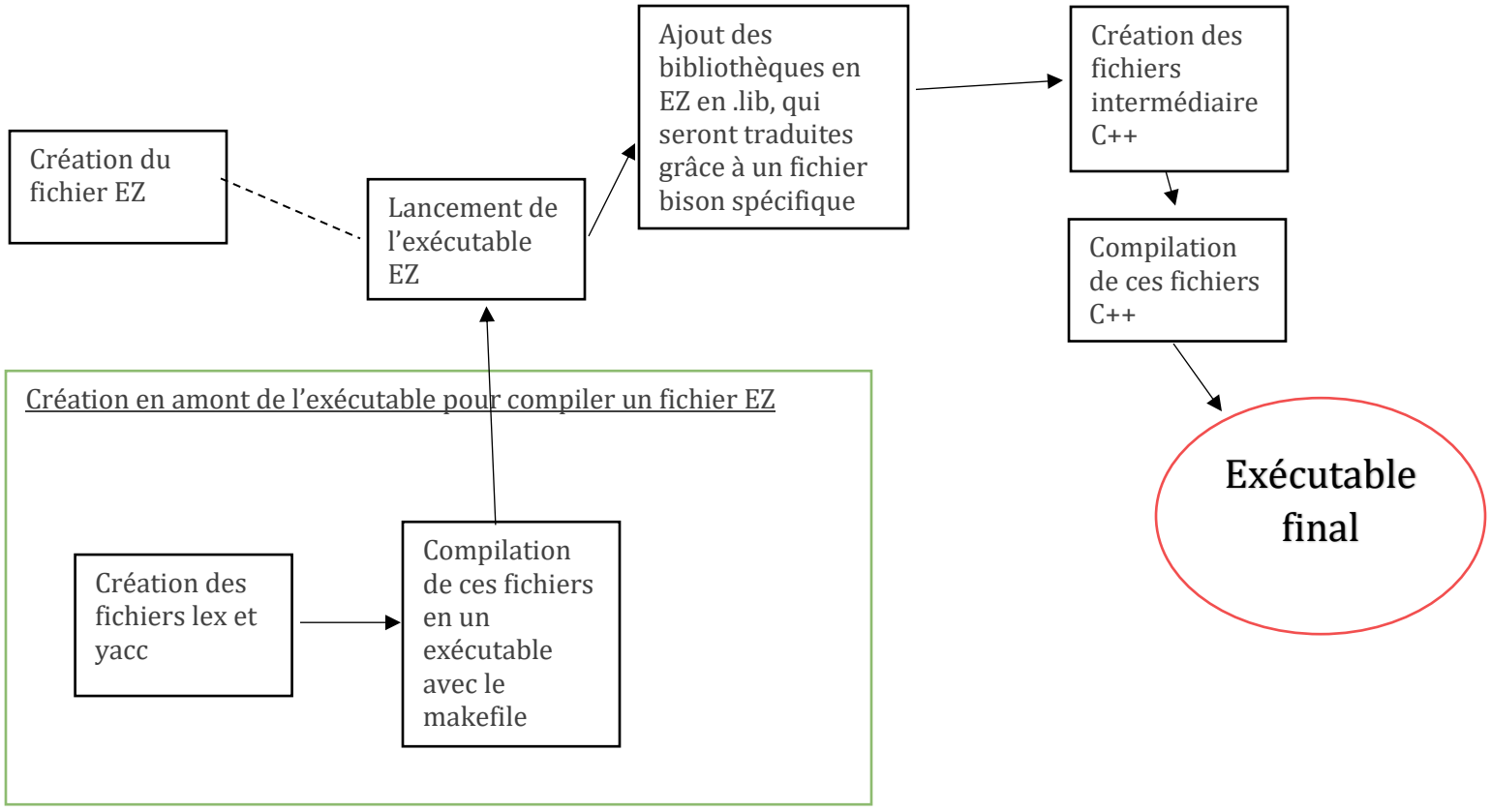
QUELLES SONT LES OPTIONS DE BASES UTILISEES PAR LE COMPILATEUR

(Voir avec Alexandre)

Organisation des bibliothèques du langage

LES BIBLIOTHEQUES

Les différentes classes et fonctions utilisées dans EZ seront rangées dans des fichiers en extension « .lib ». Ces fichiers seront des fichiers avec une syntaxe particulière qui permet de définir les entêtes de fonctions. Seront ensuite définis dans les fichiers bisons les traitements à effectuer pour transformer ces appels de fonctions EZ en appels de fonctions C++.



INCLUSION DES BILBIOTHEQUES

Les bibliothèques doivent être rangées dans un dossier particulier car elles sont différentes des fichiers EZ purs. Pour cela, elles seront rangées dans un dossier « EZ » à l'intérieur de « /usr/lib/ ». Ensuite, ces bibliothèques doivent être implémentées dans le code :

- Intégration complète des bibliothèques :
  Lors de la compilation, c'est lourd mais cela permet d'éviter une gestion complète des bibliothèques. Cela peut-être une solution temporaire avant de passer à une autre méthode

Convention de nommage :

L'extension peut-être :

- « .a » : extension pour les librairies statiques à recompiler à chaque fois en C. Elle parait correspondre à notre utilisation et serait donc cohérente avec le reste des libraires
- « .lib_ez » : extension moins conventionnelle mais plus compréhensible. Elle peut être aussi légèrement modifié, comme par exemple « .lez » ou « .ezb »

Le nom de la bibliothèque doit commencer et correspondre à :

- Commencer par « lib »
- Contenir ensuite un ensemble court décrivant brièvement la librairie (ex : « libmath.a »)
- Finir par l'extension choisie au-dessus

Gestion des versions

Les bibliothèques peuvent être rangées de deux manières :

- Par versions de paquets (ex : g++ 5.4 et g++ 6.0 ont des paquets différents). Les paquets ont donc une version, ce qui implique une plus grande stabilité, mais permet moins de mise à jour
- Par versions de librairies, qui a donc les avantages et inconvénients inverses.

Une version plus intermédiaire pourrait correspondre à nos besoins (gestions par paquets mais avec des versions temporaires, comme si on utilisait un compilateur g++ en version 5.4.0.1 qui n'est pas stable et que d'autres personnes travaillent sur la 5.4.0.2 et qu'on merge en 5.4.1)

Installation et déploiement

Sur les différents schémas, on rend compte de deux compilations différentes :

- Celle d'EZ vers C++ avec notre exécutable
- Celle de C++ vers Exécutable avec G++

Ces deux compilations sont séparées mais s'active en série avec le lancement de l'exécutable EZ. Les deux peuvent donc faire partie du même exécutable.

Dans le déploiement, deux étapes sont à définir :

- Mettre une variable externe remontant à ce dossier
- Le programme peut alors utiliser les différents fichiers présents sur la machine, sans avoir à être dans un lieu particulier.