CMake : Travaux pratiques

30 novembre 2016

Ce TP va permettre de découvrir le système de production de programme CMake au travers d'un exemple de projet en C.

Introduction : le projet *dinner*

Ce projet est une implémentation un peu spéciale du problème du dîner des philosophes de Dijkstra ¹.

Au lieu de résoudre le problème d'accès aux couverts par l'ordonnancement des actions, les philosophes se combattent comme dans un jeu de rôle pour s'approprier le couvert détenu par leur voisin. Lorsqu'ils gagnent le combat, ils peuvent manger des spaghettis.

Un philosophe qui n'arrive pas à manger meurt de famine au bout d'un certain temps.

Le projet est composé de plusieurs éléments :

- un protocole, défini dans le fichier `dinnerd/DiningHackers.txt`,
- la bibliothèque partagée `libdinner.so` qui implémente le protocole de communication,
- le serveur `dinnerd` qui gère les interactions entre les philosophes,
- le programme `hungryhacker` qui implémente le comportement d'un philosophe affamé,
- le programme `vdinner` qui permet de visualiser la scène via une interface OpenGL.

Création d'une infrastructure de compilation minimale

1. Récupérer les sources du projet :

```
git clone http://192.168.1.1/git/dinner.git
```

2. Créer un fichier `CMakeLists.txt` à la racine du projet pour descendre dans `libdinner`
3. Créer un fichier `CMakeLists.txt` dans le répertoire `libdinner` pour compiler la bibliothèque dynamique `libdinner.so` à partir des fichiers sources.

1. https://fr.wikipedia.org/wiki/Dîner_des_philosophes

4. Créer un répertoire de construction `build`, générer le système de compilation dans ce répertoire et compiler la bibliothèque. Vérifier qu'elle est construite correctement.
5. Compléter en écrivant les fichiers `CMakeLists.txt` pour les répertoires `dinnerd`, `hungryhacker` et `vdinner`.
6. Construire le projet complet

Exécution

1. Lancer `dinnerd` et `vdinner` dans un terminal
2. Lancer plusieurs processus `hungryhacker`, par exemple à l'aide du script `sample.sh` fourni dans le répertoire.
3. Manger du pop-corn en observant le résultat...

Améliorer le projet

Une fois l'infrastructure de base en place, il y a plusieurs choses qui peuvent être faites pour améliorer et compléter le projet.

1. Prévoir l'installation des programmes et de la bibliothèque dans le système pour pouvoir fournir un paquet binaire.
2. Utiliser CPack pour automatiser la production d'un paquet binaire au format Debian
3. Ajouter le traitement des pages de manuel Unix dans le projet (mise en pages, installation, empaquetage).
4. `vdinner` a besoin des bibliothèques OpenGL et de `freeglut` pour se compiler et s'exécuter. Ajouter un module pour tester la présence de Freeglut et de libGL au projet. Ajouter la dépendance à l'exécution dans le paquet binaire.

Utilisation de ninja

Ninja est un outil de gestion de compilation de bas niveau, similaire à make. Ninja a été conçu pour être plus rapide que make. CMake sait utiliser ninja via le *générateur* Ninja.

1. Créer un nouveau répertoire de compilation `build-ninja`.
2. Générer l'infrastructure de compilation avec le générateur Ninja :

```
cmake -G Ninja ..
```
3. Compiler et empaqueter le projet avec ninja.
4. Comparer les temps d'exécution avec le système basé sur make