Utilisation de CMake

Préambule

CMake est un outil qui permet de générer des fichiers de configuration pour compiler un projet sur de multiples plateformes, à partir d'un script qui décrit les différents composants qui le composent. Ce TP va vous permettre de vous familiariser avec cet outil, en créant une application multi plateformes qui implique l'utilisation de bibliothèques différentes en fonction des plateformes.

1/ Compilation d'un hello world avec CMake

- Ecrivez un programme C++ qui permet d'afficher un message 'hello world' sur la ligne de commande à l'aide d'un printf.

- Créer un fichier CMakeLists qui permet de générer le makefile nécessaire à la compilation de ce projet. Pour cela, vous aurez recours aux commandes cmake suivantes : cmake_minimum_required, project, add_executable

- Compiler votre projet à l'aide de cmake (utiliser –G pour spécifier le compilateur que vous souhaitez utiliser)

2/ Remplacement du printf par une boite de dialogue sous windows

2.1/ Ajout de définitions préprocesseur en fonction de la plateforme

- Affichez le système d'exploitation sur lequel vous exécutez cmake.

  Pour cela, modifiez le fichier CMakeLists.txt afin de rajouter un message à l'aide de la commande 'message', et affichez le contenu de la variable 'CMAKE_SYSTEM_NAME'.

  commande cmake : message

- Rajoutez une définition préprocesseur en fonction du système d'exploitation courant.

  Vous utiliserez à l'intérieur du fichier CMakeLists.txt un bloc de type 'if / endif' ainsi que les fonctions STREQUAL / MATCHES pour tester la valeur de 'CMAKE_SYSTEM_NAME', et appelez la commande cmake 'add_definitions' afin de définir '_SYSTEM_LINUX' sur les systèmes Linux, et '_SYSTEM_WINDOWS' sur les systèmes Windows.

2.2/ Modification du programme en fonction de la plateforme

- Créez une fonction DisplayMessage(const char * szMessage). Dans un premier temps, cette fonction contient juste un printf.

- Remplacez le contenu de cette fonction afin qu'elle utilise la fonction 'printf' (contenue dans stdio.h) sur Linux, et la fonction MessageBox (contenue dans windows.h) sur Windows.

  Vous veillerez à utiliser #ifdef / #endif pour conditionner le code spécifique à chaque plateforme aux définitions de _SYSTEM_LINUX ou _SYSTEM_WINDOWS.

2.3/ Création d'une librairie statique contenant la fonction DisplayMessage

- Déplacer l'implémentation de la fonction 'DisplayMessage' dans le fichier DisplayMessage.cpp et le prototype correspondant dans le fichier DisplayMessage.h. Faire attention à positionner ces fichiers dans un répertoire différent du répertoire courant : 'lib/', afin de pouvoir y placer le fichier CMakeLists.txt nécessaire à la compilation de la librairie.

- Créer dans le répertoire 'lib/' un nouveau fichier CMakeLists.txt afin de créer une librairie statique 'DisplayMessage.a' à partir du code contenu dans lib/.

  Pour cela, vous utiliserez la commande cmake add_library, configurée en mode STATIC.

  Attention : CMake ne gère que des chemins absolus. Il faut donc bien veiller à référencer votre librairie à l'aide de son chemin absolu dans le CMakeLists.txt

2.4/ Utilisation de la librairie statique dans le programme principal

- Modifier le programme principal afin d'inclure DisplayMessage.h dans le répertoire 'lib/'.

- Modifier le CMakeLists.txt associé au programme principal afin de lier l'exécutable avec la librairie 'DisplayMessage.a'

- Vous utiliserez pour cela la commande cmake 'target_link_libraries'.

2.5/ Création d'une librairie dynamique contenant la fonction DisplayMessage et utilisation dans le programme principal

- Sur le modèle de 2.3 et 2.4, créer une librairie dynamique (DisplayMessage.dll sous windows / DisplayMessage.so sous Linux) dans le répertoire dll/, en utilisant la commande cmake_add_library configurée en mode 'SHARED'.

3/ Création d'une configuration 'Debug' et d'une configuration 'Release' pour chacune des librairies, ainsi que pour le programme principal

- Vous utiliserez la possibilité de fournir à cmake une variable sur la ligne de commande à l'aide de '-D' afin de spécifier quelles options de compilation activer.

- La configuration 'Debug' contiendra les informations de debuggage, et aucune optimisation.

- La configuration 'Release' ne contiendra pas d'information de debuggage, et sera compilée avec le niveau d'optimisations maximal.