4. Utilisation de cmake

4.1 Exercice 1 – exemple simple d'utilisation de cmake

Créez un répertoire de travail EDLTP2/exercice1. Copiez les fichiers suivants dans ce répertoire :

```
TP1Exercice1/TP1EX1SourcesCompletes/src/ddtools.[ch]
TP1Exercice1/TP1EX1SourcesCompletes/gpl-3.0.txt
TP1Exercice1/TP1EX1PourEtudiants/pingpong/tstpingpong.c
```

Vous devez obtenir la liste suivante à l'issue d'une commande ls -1 dans ce répertoire :

```
ddtools.c
ddtools.h
gpl-3.0.txt
tstpingpong.c
```

Question : créez un fichier CMakeLists.txt afin de générer un Makefile pour compiler tstpingpong sans passer par la création d'une archive « libddtools.a ».

Indices :

- Pour inclure le répertoire « . » dans la liste des répertoires contenant des fichiers « .h » il faut ajouter la commande « INCLUDE_DIRECTORIES(.) » dans le fichier CMakeLists.txt.
- Pour générer le fichier Makefile à partir de votre fichier CMakeLists.txt, il faut utiliser la commande « cmake . », sans oublier le « . » !
- Pour forcer cmake à tout regénérer, il suffit de supprimer le fichier CMakeCache.txt ; vous devriez d'ailleurs éditer ce fichier car son contenu peut être utile dans certains cas.
- Pour vérifier les options disponibles, une fois le Makefile généré, il suffit de taper « make help » ; pour compiler le projet, il suffit de taper « make » puis « ./tstpingpong » pour lancer l'exécutable.

Compléments (inutiles pour cette question): voir les commandes LINK_DIRECTORIES et LINK_LIBRARIES dans le fichier CMakeLists.txt via le fichier /usr/share/doc/cmake/cmake.html

4.2 Exercice 2 – bibliothèque statique libddtools.a (1/2)

Créez un répertoire de travail EDLTP2/exercice2. Copiez les fichiers suivants dans ce répertoire :

```
TP1Exercice1/TP1EX1SourcesCompletes/src/ddtools.[ch]
TP1Exercice1/TP1EX1SourcesCompletes/gpl-3.0.txt
TP1Exercice1/TP1EX1PourEtudiants/pingpong/tstpingpong.c
```

Question : créez un fichier CMakeLists.txt afin de générer un Makefile pour compiler tstpingpong en passant par la création d'une archive « libddtools.a ».

Compléments : Pour créer le fichier CMakeLists.txt, utilisez les commandes suivantes en les adaptant :

`ADD_LIBRARY(malib fichier1danslibrairie.c fichier2danslibrairie.c)`
pour créer la librairie « libmalib » à partir d'un ensemble de fichiers (effectue l'équivalent de l'appel à `ar cr libmalib.a fichier1danslibrairie.o fichier2danslibrairie.o`)

`TARGET_LINK_LIBRARIES(tstpingpong malib)`
à ajouter en fin de fichier CMakeLists.txt pour effectuer l'édition des liens en incluant la librairie « libmalib » (provoque l'ajout de « -lmalib » lors de l'édition des liens (via cc ... -o ...))

Compléments : Si tout s'est bien passé, testez les commandes suivantes (*cf.* pages de man avant) :

```
nm libddtools.a
strip libddtools.a
```

4.3 Exercice 3 – bibliothèque statique libddtools.a (2/2)

En vous inspirant de la section 2 du TP1, compilez « tstpingpong » en mettant en place les deux niveaux d'accès exposés lors du TP1, grâce à la bibliothèque statique « libddtools.a » :

- un niveau pour l'enseignant ayant accès aux sources complètes de la librairie libddtools.a ;
- un niveau étudiants, ayant accès aux sources des exemples et à la librairie libddtools.a.

Pour réaliser ceci à l'aide de make, il fallait ajouter un fichier Makefile par répertoire. Avec cmake, il faut ajouter des fichiers **CMakeLists.txt** dans chaque répertoire pour obtenir l'arborescence suivante :

```
.
|-- TP2EX3PourEtudiants
|   |-- CMakeLists.txt
|   |-- pingpong
|   |   |-- CMakeLists.txt
|   |   `-- tstpingpong.c
|   `-- src
|       |-- ddtools.h
|       `-- libddtools.a -> ../../TP2EX3SourcesCompletes/src/libddtools.a
`-- TP2EX3SourcesCompletes
    |-- CMakeLists.txt
    `-- src
        |-- CMakeLists.txt
        |-- ddtools.c
        `-- ddtools.h
```

Cette arborescence – obtenue grâce à la commande "tree" – est extraite d'une arborescence plus complète, ce qui explique que les fichiers **CMakeLists.txt** situés dans les sous-répertoires « TP2EX3PourEtudiants » et « TP2EX3SourcesCompletes » se réduisent à des simples « relais » destinés à accéder aux fichiers **CMakeLists.txt** des sous-répertoires « TP2EX3PourEtudiants/pingpong » et « TP2EX3SourcesCompletes/src ». Réalisez les fichiers **CMakeLists.txt**.

4.4 Exercice 4 – bibliothèque partagée libddtools.so

En vous inspirant de la section précédente, compilez « tstpingpong » en mettant en place les deux niveaux d'accès grâce à la bibliothèque partagée « libddtools.so » :

- un niveau pour l'enseignant ayant accès aux sources complètes de la librairie libddtools.so ;
- un niveau étudiants, ayant accès aux sources des exemples et à la librairie libddtools.so.

Créez les fichiers **CMakeLists.txt** dans chaque répertoire pour obtenir l'arborescence suivante :

```
.
|-- TP2EX4PourEtudiants
|   |-- CMakeLists.txt
|   |-- pingpong
|   |   |-- CMakeLists.txt
|   |   `-- tstpingpong.c
|   `-- src
|       |-- ddtools.h
|       `-- libddtools.so -> ../../TP2EX4SourcesCompletes/src/libddtools.so
`-- TP2EX4SourcesCompletes
    |-- CMakeLists.txt
    `-- src
        |-- CMakeLists.txt
        |-- ddtools.c
        `-- ddtools.h
```

Compléments : Pour créer le fichier CMakeLists.txt, utilisez les commandes suivantes en les adaptant :

`ADD_LIBRARY(malib SHARED fichier1danslibrairie.c fichier2danslibrairie.c)`
pour créer la librairie partagée « libmalib.so » à partir d'un ensemble de fichiers

`TARGET_LINK_LIBRARIES(tstpingpong malib)`
fonctionne également pour les librairies/bibliothèques partagées, donc rien à changer à ce niveau en fin de fichier CMakeLists.txt pour effectuer l'édition des liens en incluant la librairie « libmalib.so »

Compléments : Si tout s'est bien passé, testez les commandes suivantes (cf. pages de man avant) :

```
nm libddtools.so
strip libddtools.so
```

5. Utilisation de PkgConfig

Cet outil est capable de lire des informations stockées dans un fichier portant l'extension « .pc » (cf cours et man). Par exemple (/home/duvivier/usr/lib/pkgconfig/vle-0.6.pc) :

```
prefix=/home/duvivier/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include

Name: vle
Description: VLE multimodelling and Simulation tools
Requires: libxml++-2.6 glibmm-2.4
Version: 0.6.0
Libs: -L${libdir} -lvledata \
      -lvledevs -lvleextension -lvlegeometry -lvlegraph -lvlemanager \
      -lvleutils -lvleoov -lvlevalue -lvlevpz -lvletranslator
Cflags: -I${includedir}/vle-0.6.0 -I/usr/include
```

Exemple d'utilisation (via une commande en ligne) :

aptitude install libglibmm-2.4-dev libglibmm-2.4-doc

-> Installe « glibmm-2.4 » utilisée pour la suite des tests

pkg-config --libs glibmm-2.4

-> Retourne les librairies à inclure pour compiler en incluant glibmm-2.4

pkg-config --cflags glibmm-2.4

-> Retourne les « cflags » à include pour compiler en incluant glibmm-2.4

Exemple d'utilisation dans un fichier Makefile (cf. man) :

```
program: program.c
      cc program.c `pkg-config --cflags --libs gnomeui`
```

Pour utiliser certaines fonctionnalités, un fichier .cmake doit être installé à la racine de votre projet.
Par exemple (/home/duvivier/VLE/vle/cmake/FindPkgConfig.cmake copié depuis /usr/share/cmake-2.6/Modules/FindPkgConfig.cmake) :

```
# - a pkg-config module for CMake
#
# Usage:
#   pkg_check_modules(<PREFIX> [REQUIRED] <MODULE> [<MODULE>]*)
#     checks for all the given modules
#
#   pkg_search_module(<PREFIX> [REQUIRED] <MODULE> [<MODULE>]*)
#     checks for given modules and uses the first working one
# ...
```

Exemple de recherche de paquets, librairies et « cflags » via le fichier « CmakeLists.txt » :

Recherche des paquets :

```
FIND_PACKAGE(PkgConfig)
PKG_CHECK_MODULES(GLIBMM glibmm-2.4)
```

Utilisation des informations (--libs et --cflags) :

```
INCLUDE_DIRECTORIES(${GLIBMM_INCLUDE_DIRS})
LINK_DIRECTORIES(${GLIBMM_LIBRARY_DIRS})
ADD_EXECUTABLE(example example1.cpp example2.cpp)
TARGET_LINK_LIBRARIES(example ${GLIBMM_LIBRARIES})
```

6. Et la suite ?

Je passe en mode « client » --> de plus en plus, je vais vous laisser vous débrouiller seul pour trouver et diffuser la doc (via groupe google ou wiki(s) afin que je puisse suivre où vous en êtes par exemple)...

Il y a encore quelques points à voir en cours en détail ... mais ... pour la suite je donnerai des pistes, je fournirai des/certaines documentations ET SURTOUT ... un projet !