

Introduction à CMake et CTest

Anne Cadiou

Laboratoire de Mécanique des Fluides et d'Acoustique

Informatique scientifique pour le calcul
École doctorale
2015-2016



ÉCOLE
CENTRALE LYON



Table des matières

- 1 Contexte
- 2 Exemple simple
- 3 Exemple avec lien
- 4 Paramètres
- 5 CTest
- 6 Conclusion

Table des matières

- 1 Contexte
- 2 Exemple simple
- 3 Exemple avec lien
- 4 Paramètres
- 5 CTest
- 6 Conclusion

Qu'est ce que c'est ?

CMake, CTest, CPack, CDash sont des outils opensource

- développés par Kitware (VTK - Visualisation ToolKit) depuis 2001
- outils adoptés par les développeurs de KDE

- **CMake** est un outil de **gestion du processus de compilation**
- **CTest** permet de faire des **tests unitaires**
- **CPack** permet de construire des **packages**
- **CDash** est un outil d'**intégration continue**

Adopter une méthodologie d'**intégration continue**

Visé à l'automatisation des tâches (compilation, tests unitaires et fonctionnels, tests de performances, validation, documentation ...)

Pour faire quoi ?

CMake

- aide à générer le Makefile pour différentes plateformes
 - pour compiler des sources
- possède de nombreuses commandes permettant de localiser les dépendances
 - recherche de bibliothèques
 - facilite le portage ou la gestion de différents compilateurs
- permet l'interface avec des tests unitaires

CTest/Boost

- permet de faire des tests unitaires
 - tests différents du code, permettant d'en valider une partie déterminée ou une fonctionnalité particulière
 - à écrire avant le code dans la méthodologie Extreme Programming
- s'utilise avec ou sans CMake

Références

- Kitware

<http://www.cdash.org/>

<https://cmake.org/Wiki/CMake>

Professional Training Courses, Kitware à Lyon

- Livre

K. Martin, B. Hoffman, "Mastering CMake" book, Kitware ed.

- Liste de discussion

<https://cmake.org/mailling-lists/>

- nombreuses ressources en ligne, par exemple

<https://github.com/TheErk/CMake-tutorial>

<http://sirien.metz.supelec.fr/depot/SIR/TutorielCMake/index.html>

Autres outils

Outils généraux de construction (et packaging)

Autotools, Ant, Scons, ...

suivent une démarche connue :

```
./configure; make; make install
```

pour le déploiement d'application (une tâche complexe)

Outils de développement, suivant une démarche d'intégration continue

Eclipse, Visual Studio, KDevelop ...

Ici on se focalise sur la configuration, la compilation, l'exécution de tests unitaires multi-plateformes dans le cadre d'un développement pour le calcul scientifique

Table des matières

1 Contexte

2 Exemple simple

3 Exemple avec lien

4 Paramètres

5 CTest

6 Conclusion

Projet simple

Programme seul, sans dépendances.

```
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "'Hello, world!'\n";
6 }
```

Structure du projet

```
project/
├── src/
│   └── prog.cpp
```

Objectif : compiler le code, l'installer, le tester.

Compilation manuelle

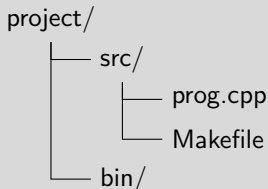
en utilisant les compilateurs de GNU

```
cadiou@moon:~/project/src$ g++ prog.cpp
```

Génération de l'exécutable

```
cadiou@moon:~/project/src$ ls
cadiou@moon:~/project/src$ a.out prog.cpp
cadiou@moon:~/project/src$ ./a.out
'Hello, world!'
```

Création manuelle d'un Makefile



```
cadiou@moon:~/project/src$ make -n
g++ -c prog.cpp
g++ -o a.out prog.o
cadiou@moon:~/project/src$ make
cadiou@moon:~/project/src$ make install
mv a.out ../bin/.
cadiou@moon:~/project/src$ cd ../bin
cadiou@moon:~/project/bin$ ./a.out
```

Exemple de Makefile

```
CC          = g++
CLINKER     = g++

OBJS        = prog.o

.SUFFIXES: .o .cpp
.cpp.o:
    $(CC) -c $(CCLAGS) $<

EXE = a.out

all : $(EXE)

$(EXE): $(OBJS)
    $(CLINKER) $(LDFLAGS) -o $@ $(OBJS)

clean:
    rm -f $(OBJS) $(EXE)

install:
    mv $(EXE) ../bin/.
```

Construction par CMake

```
project/  
├── src/  
│   ├── prog.cpp  
│   └── CMakeLists.txt  
├── bin/  
└── CMakeLists.txt
```

```
cadiou@moon:~/project$ mkdir build; cd build  
cadiou@moon:~/project/build$ cmake ..  
cadiou@moon:~/project/build$ make  
Scanning dependencies of target a.out  
[100%] Building CXX object src/CMakeFiles/a.out.dir/prog.o  
Linking CXX executable a.out  
[100%] Built target a.out  
cadiou@moon:~/project/build$ make install  
Install the project...  
-- Install configuration: "Release"  
-- Installing: ~/project/build/../../bin/a.out
```

CMakeLists.txt à la racine du projet

```
# cmake version
cmake_minimum_required(VERSION 2.8.10 FATAL_ERROR)

# define project name and langage
project(MyProject CXX)

# define executable names
set(EXE MyProgram)

# define directories
set(SRC      ${CMAKE_SOURCE_DIR}/src)

add_subdirectory(${SRC})
```

CMakeLists.txt associé aux sources

```
# set up the source files compilation

# add the source files
set(SRC
    prog.cpp
)

# define the executable in terms of the source files
add_executable(${EXE} ${SRC})

# how to install this executable
install(TARGETS ${EXE} RUNTIME DESTINATION bin)
```

Makefile vs. CMake

Make

```
main .o: main .c main .h
<TAB > cc -c main .c
MyProgram : main .o
<TAB > cc -o MyProgram main .o -lm -lz
```

CMake

```
PROJECT (MyProject C)
ADD_EXECUTABLE (MyProgram main .c)
TARGET_LINK_LIBRARIES (MyProgram z m)
```


Pourquoi ne pas se contenter d'appeler le compilateur ?

Dans les codes de calculs, la construction de l'exécutable est une tâche parfois délicate. Elle dépend

- d'un **grand nombre de fichiers**
- avec de nombreuses **dépendances**
- du cas à traiter (compilation **conditionnelle**, ...)
- de nombreux **éléments externes** (bibliothèques, ...)
- et de la manière dont ils sont installés
- de la **plateforme**
- de l'**environnement** sur le plateforme

La plupart du temps, on ne contrôle pas ces dépendances.

Pourquoi ne pas se contenter d'écrire un makefile ?

Le makefile ne fait pas tout :

- il **dépend de la plateforme**
- il est essentiellement orienté sur les OS de type **unix**

C'est un outils puissant qui peut être paramétré, mais dans la grande majorité des cas :

- les dépendances doivent être **explicitées manuellement**
- ainsi que la **recherche des éléments externes**

Sur une plateforme donnée, les **performances** peuvent varier suivant

- l'environnement,
- le compilateur, ses options (tester, c'est refaire les dépendances...)

Dans un makefile, se trouvent explicitées à la fois les **commandes de construction** du code et la **manière** de le faire sur une plateforme donnée, pour un environnement donné. Le portage est facilité si ces deux étapes sont **séparées**.

Table des matières

- 1 Contexte
- 2 Exemple simple
- 3 Exemple avec lien**
- 4 Paramètres
- 5 CTest
- 6 Conclusion

Projet parallèle

Programme appelant la bibliothèque MPI

```
1 #include <mpi.h>
2 #include <iostream>
3
4 int main(int argc, char* argv[])
5 {
6     MPI::Init(argc, argv);
7
8     int rank = MPI::COMM_WORLD.Get_rank();
9     int size = MPI::COMM_WORLD.Get_size();
10
11     std::cout << "Process " << rank << "/" << size << " says '
        Hello, world!'\n";
12
13     MPI::Finalize();
14     return 0;
15 }
```

Même structure du projet que précédemment

Compilation manuelle

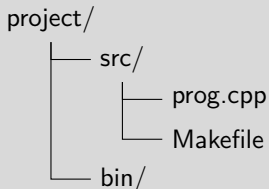
en utilisant les compilateurs de GNU

```
cadiou@moon:~/project/src$ mpicxx prog.cpp
```

Génération de l'exécutable

```
cadiou@moon:~/project/src$ ls
cadiou@moon:~/project/src$ a.out prog.cpp
cadiou@moon:~/project/src$ ./a.out
cadiou@moon:~/project/src$ mpirun -np 4 a.out
Process 1/4 says 'Hello, world!'
Process 2/4 says 'Hello, world!'
Process 3/4 says 'Hello, world!'
Process 0/4 says 'Hello, world!'
```

Création manuelle d'un Makefile



```
cadiou@moon:~/project/src$ make -n
mpicxx -c prog.cpp
mpicxx -o a.out prog.o -lmpi
cadiou@moon:~/project/src$ make
cadiou@moon:~/project/src$ make install
mv a.out ../bin/.
cadiou@moon:~/project/src$ cd ../bin
cadiou@moon:~/project/bin$ ./a.out
```

Exemple de Makefile

```
.SUFFIXES:
.SUFFIXES: .cpp .o

CXX    = mpicxx
CXXFLAGS =
LDFLAGS = -lmpi

EXE    = a.out

SRC= \
    prog.cpp
OBJ=    $(SRC:.cpp=.o)

.cpp.o:
    $(CXX) $(CXXFLAGS) -c $<

all:    $(EXE)
$(EXE): $(OBJ)
    $(CXX) -o $@ $^ $(LDFLAGS)
clean:
    rm -f $(OBJ) $(EXE) core
install:
    mv $(EXE) ../bin/.
```

Construction par CMake

```
project/  
├── src/  
│   ├── prog.cpp  
│   └── CMakeLists.txt  
├── bin/  
└── CMakeLists.txt
```

```
cadiou@moon:~/project$ mkdir build; cd build  
cadiou@moon:~/project/build$ cmake ..  
cadiou@moon:~/project/build$ make  
Scanning dependencies of target a.out  
[100%] Building CXX object src/CMakeFiles/a.out.dir/prog.o  
Linking CXX executable a.out  
[100%] Built target a.out  
cadiou@moon:~/project/build$ make install  
Install the project...  
-- Install configuration: "Release"  
-- Installing: ~/project/build/../../bin/a.out
```


CMakeLists.txt à la racine du projet

```
# cmake version
cmake_minimum_required(VERSION 2.8.10 FATAL_ERROR)

# define project name and langage
project(MyProject CXX)

# MPI
find_package(MPI REQUIRED)
include_directories(${MPI_INCLUDE_PATH})
set(INCLUDE ${MPI_INCLUDE_PATH})

# define executable names
set(EXE MyProgram)

# define directories
set(SRC      ${CMAKE_SOURCE_DIR}/src)

add_subdirectory(${SRC})
```

CMakeLists.txt associé aux sources

```
# set up the source files compilation

# add the source files
set(SRC
    prog.cpp
)

# define the executable in terms of the source files
add_executable(${EXE} ${SRC})

# add the needed libraries and special compiler flags
set_target_properties(${EXE} PROPERTIES
    COMPILE_FLAGS "${MPI_CXX_COMPILE_FLAGS}"
    LINK_FLAGS "${MPI_CXX_LINK_FLAGS}")
include_directories(${MPI_CXX_INCLUDE_PATH})
target_link_libraries(${EXE} ${MPI_CXX_LIBRARIES})

# how to install this executable
install(TARGETS ${EXE_MPI} RUNTIME DESTINATION bin)
```

Syntaxe

- Configuration dynamique

```
configure_file(config.h.in config.h)
```

```
#cmakedefine FOO_VERSION ${FOO_VERSION}  
#cmakedefine BUILD_SHARED_LIBS
```

- Détection de bibliothèque

```
find_package(BZip2)  
if (BZIP2_FOUND)  
    include_directories(${BZIP2_INCLUDE_DIRS})  
    target_link_libraries(${PROJET} ${BZIP2_LIBRARIES})  
endif (BZIP2_FOUND)
```

```
find_package(Doxyxygen)  
find_program(DOXYPY doxypy)  
if (DOXYGEN_FOUND)  
    add_subdirectory(doc)  
endif (DOXYGEN_FOUND)
```

Table des matières

1 Contexte

2 Exemple simple

3 Exemple avec lien

4 Paramètres

5 CTest

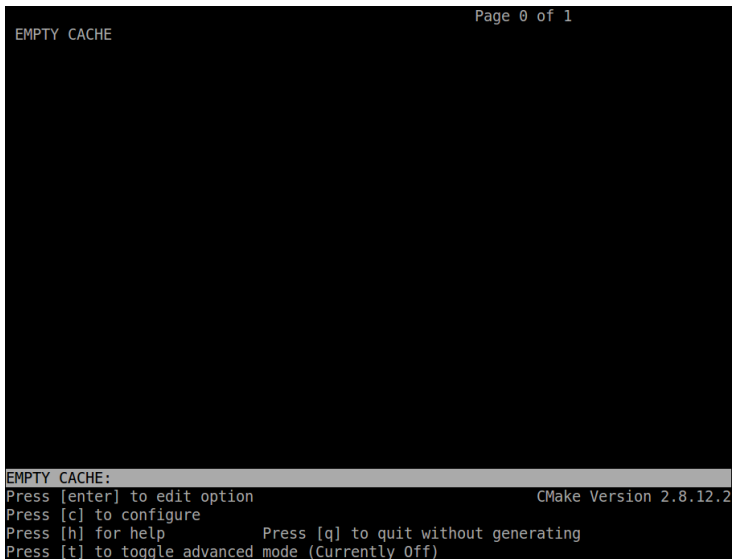
6 Conclusion

Lancement de CMake

```
cadiou@moon:~/project/build$ cmake ..
-- The CXX compiler identification is GNU 4.8.4
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: ~/project/build
```

Lancement en mode interactif (quasi graphique !)

```
cadiou@moon:~/project/build$ cmake ..
```

A screenshot of the CMake interactive configuration interface. The window has a black background. In the top right corner, it says "Page 0 of 1". On the left side, the text "EMPTY CACHE" is displayed. At the bottom, there is a light gray bar containing the text "EMPTY CACHE:". Below this bar, the following instructions are listed: "Press [enter] to edit option", "Press [c] to configure", "Press [h] for help", and "Press [t] to toggle advanced mode (Currently Off)". On the right side of the bottom section, the text "CMake Version 2.8.12.2" is visible.

```
Page 0 of 1
```

EMPTY CACHE

EMPTY CACHE:

Press [enter] to edit option

Press [c] to configure

Press [h] for help

Press [t] to toggle advanced mode (Currently Off)

CMake Version 2.8.12.2

[c]

```
Page 1 of 1
CMAKE_BUILD_TYPE          *
CMAKE_INSTALL_PREFIX      */usr/local

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)
```

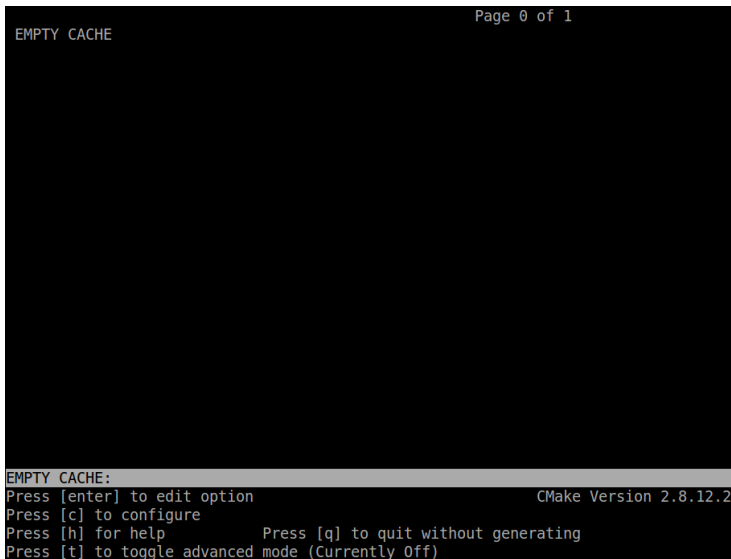
```
[c] [g] make
```

```
Page 1 of 1
CMAKE_BUILD_TYPE
CMAKE_INSTALL_PREFIX /usr/local

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option CMake Version 2.8.12.2
Press [c] to configure Press [g] to generate and exit
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```


Modifications via l'interface

```
cadiou@moon:~/project/build$ cmake ..
```

A screenshot of the CMake GUI application. The window title is "Page 0 of 1". The main area is black with the text "EMPTY CACHE" in white. At the bottom, there is a grey bar with the text "EMPTY CACHE:". Below this bar, the following text is displayed: "Press [enter] to edit option", "Press [c] to configure", "Press [h] for help", and "Press [t] to toggle advanced mode (Currently Off)". The text "CMake Version 2.8.12.2" is visible in the bottom right corner of the window.

```
Page 0 of 1  
EMPTY CACHE  
  
EMPTY CACHE:  
Press [enter] to edit option  
Press [c] to configure  
Press [h] for help  
Press [t] to toggle advanced mode (Currently Off)  
CMake Version 2.8.12.2
```

[c]

```

Page 1 of 1
CMAKE_BUILD_TYPE          *
CMAKE_INSTALL_PREFIX      */usr/local

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)
Press [q] to quit without generating
CMake Version 2.8.12.2

```

```
[enter] Release [enter]; [enter] .. [enter];
```

```
Page 1 of 1
CMAKE_BUILD_TYPE      *Release
CMAKE_INSTALL_PREFIX  *.

CMAKE INSTALL PREFIX: Install path prefix, prepended onto install directories.
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)
CMake Version 2.8.12.2
Press [q] to quit without generating
```

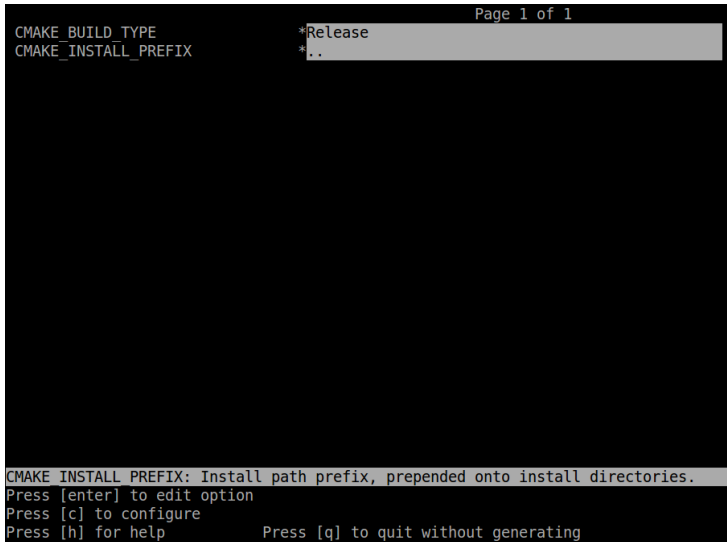
[c] [g] make

```
Page 1 of 1
CMAKE_BUILD_TYPE          Release
CMAKE_INSTALL_PREFIX      ..

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option          CMake Version 2.8.12.2
Press [c] to configure                Press [g] to generate and exit
Press [h] for help                    Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Modifications en ligne de commande

```
cadiou@moon:~/project/build$ cmake -DCMAKE_BUILD_TYPE=
Release -DCMAKE_INSTALL_PREFIX=.. ..
```



The screenshot shows a CMake configuration window titled "Page 1 of 1". It displays two configuration options: "CMAKE_BUILD_TYPE" with a value of "*Release" and "CMAKE_INSTALL_PREFIX" with a value of "*..". The window has a dark background with light text. At the bottom, there is a message: "CMAKE INSTALL PREFIX: Install path prefix, prepended onto install directories." followed by instructions: "Press [enter] to edit option", "Press [c] to configure", "Press [h] for help", and "Press [q] to quit without generating".

```
Page 1 of 1
CMAKE_BUILD_TYPE      *Release
CMAKE_INSTALL_PREFIX  *..

CMAKE INSTALL PREFIX: Install path prefix, prepended onto install directories.
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [q] to quit without generating
```

[c] [g] make

```

Page 1 of 1
CMAKE_BUILD_TYPE          Release
CMAKE_INSTALL_PREFIX      ..

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option      CMake Version 2.8.12.2
Press [c] to configure            Press [g] to generate and exit
Press [h] for help                Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

```

Contenu du répertoire build

```
project/build/
├── CMakeCache.txt
├── CMakeFiles/
├── cmake_install.cmake
├── Makefile
└── src/
    ├── CMakeFiles/
    ├── cmake_install.cmake
    └── Makefile
```

Fichier **CMakeCache.txt** généré lors de la construction

Contient l'ensemble des variables par défaut, et les dépendances internes

Aperçu du CMakeCache.txt

```
#####
# EXTERNAL cache entries
#####

//CXX compiler.
CMAKE_CXX_COMPILER:FILEPATH=/usr/bin/c++

//Flags used by the compiler during all build types.
CMAKE_CXX_FLAGS:STRING=

//Flags used by the compiler during debug builds.
CMAKE_CXX_FLAGS_DEBUG:STRING=-g

//Flags used by the compiler during release minsize builds.
CMAKE_CXX_FLAGS_MINSIZEREL:STRING=-Os -DNDEBUG

//Flags used by the compiler during release builds (/MD /Ob1
/Oi
// /Ot /Oy /Gs will produce slightly less optimized but
smaller
// files).
CMAKE_CXX_FLAGS_RELEASE:STRING=-O3 -DNDEBUG

//Flags used by the compiler during Release with Debug Info
```


Configuration : variables principales du cache

La configuration met à jour les variables du cache, listées dans le fichier CMakeCache.txt

- Type de l'exécutable

```
CMAKE_BUILD_TYPE=[Debug , Release]
```

- Répertoire d'installation

```
CMAKE_INSTALL_PREFIX=[/usr/local , C:\Program Files]
```

- Production de bibliothèque statique ou dynamique

```
CMAKE_SHARED_LIBS=[OFF , ON]
```

- Visualisation du Makefile

```
CMAKE_VERBOSE_MAKEFILE=ON
```

Paramètres pré-définis dans le CMakeLists.txt

```
# cmake version
cmake_minimum_required(VERSION 2.8.10 FATAL_ERROR)

if (DEFINED CMAKE_BUILD_TYPE)
    set(CMAKE_BUILD_TYPE ${CMAKE_BUILD_TYPE} CACHE STRING "
        Choose the type of build, options are: None Debug
        Release RelWithDebInfo MinSizeRel.")
else()
    set(CMAKE_BUILD_TYPE Debug CACHE STRING "Choose the type
        of build, options are: None Debug Release RelWithDebInfo
        MinSizeRel.")
endif()

# define project name and langage
project(MyProject CXX)

# define executable names
set(EXE MyProgram)

# define directories
set(SRC      ${CMAKE_SOURCE_DIR}/src)
add_subdirectory(${SRC})
```

Action...

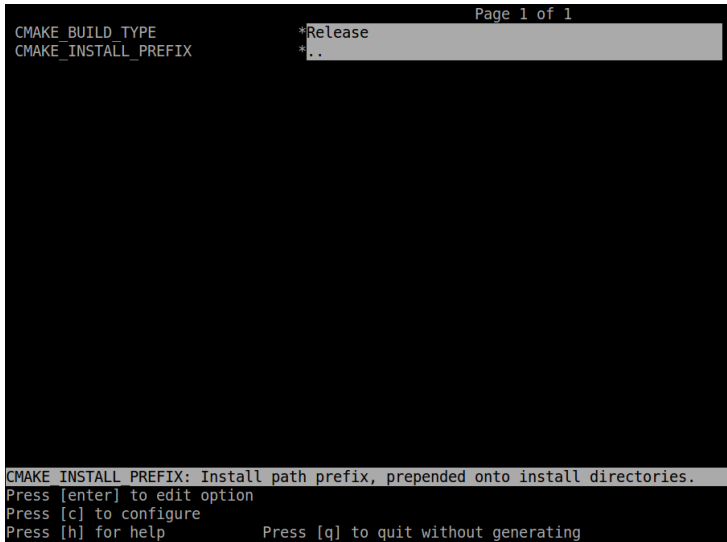
```
cadiou@moon:~/project/build$ cmake ..
```

```
Page 1 of 1
CMAKE_BUILD_TYPE          *Debug
CMAKE_INSTALL_PREFIX      */usr/local

CMAKE BUILD TYPE: Choose the type of build, options are: None Debug Release RelWithDebInfo
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)
Press [q] to quit without generating
CMake Version 2.8.12.2
```

... modifications en ligne de commande

```
cadiou@moon:~/project/build$ cmake -DCMAKE_BUILD_TYPE=
Release -DCMAKE_INSTALL_PREFIX=.. ..
```



The screenshot shows a terminal window with a black background. At the top right, it says "Page 1 of 1". On the left, the labels "CMAKE_BUILD_TYPE" and "CMAKE_INSTALL_PREFIX" are visible. To their right, the values "*Release" and "*.." are shown, each preceded by an asterisk. At the bottom of the terminal, there is a light gray banner with the text "CMAKE INSTALL PREFIX: Install path prefix, prepended onto install directories." Below this banner, the following instructions are displayed: "Press [enter] to edit option", "Press [c] to configure", "Press [h] for help", and "Press [q] to quit without generating".

```
Page 1 of 1
CMAKE_BUILD_TYPE      *Release
CMAKE_INSTALL_PREFIX  *..

CMAKE INSTALL PREFIX: Install path prefix, prepended onto install directories.
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [q] to quit without generating
```

Modification de la variable d'installation

Par défaut

```
CMAKE_INSTALL_PREFIX=/usr/local
```

défini en chemin absolu

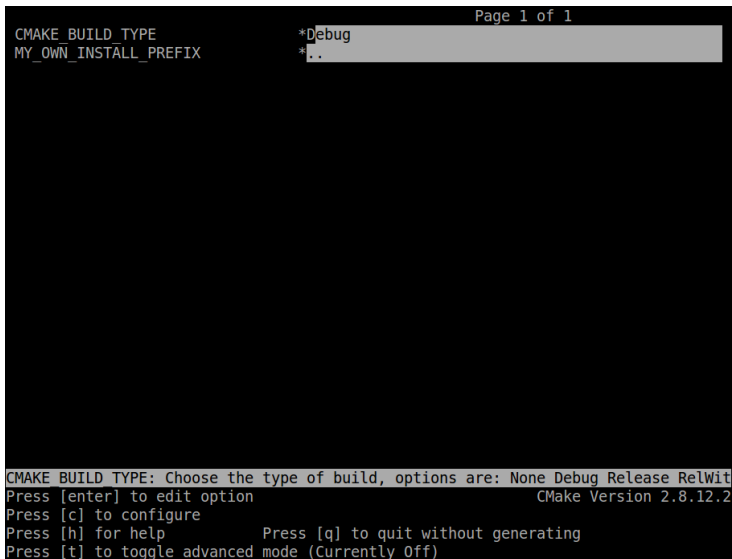
Définir une variable spécifique, pré-définie, modifiable en ligne de commande, par exemple en ajoutant

```
set(MY_OWN_INSTALL_PREFIX ".." CACHE PATH "Prefix prepended  
to install directories")  
SET(CMAKE_INSTALL_PREFIX "${MY_OWN_INSTALL_PREFIX}" CACHE  
INTERNAL "Prefix prepended to install directories" FORCE  
)
```

dans le fichier **CMakeLists.txt** du projet.

Action...

```
cadiou@moon:~/project/build$ cmake ..
```



```
Page 1 of 1
CMAKE_BUILD_TYPE          *Debug
MY_OWN_INSTALL_PREFIX     *..

CMAKE BUILD TYPE: Choose the type of build, options are: None Debug Release RelWith
Press [enter] to edit option                                     CMake Version 2.8.12.2
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

... en ligne de commande

```
cadiou@moon:~/project/build$ cmake -DMY_OWN_INSTALL_PREFIX=
other ..
```

The screenshot shows a terminal window with a black background. At the top right, it says "Page 1 of 1". On the left, there are two labels: "MY_OWN_INSTALL_PREFIX" and "CMAKE_BUILD_TYPE". To the right of these labels, there is a list of options: "*other" (which is highlighted with a light gray background) and "Debug". At the bottom of the terminal, there is a light gray bar containing the text "MY OWN INSTALL PREFIX: Prefix prepended to install directories". Below this bar, there are three lines of text: "Press [enter] to edit option", "Press [c] to configure", and "Press [h] for help". To the right of these lines, there is another line of text: "Press [q] to quit without generating".

```

Page 1 of 1
MY_OWN_INSTALL_PREFIX      *other
CMAKE_BUILD_TYPE           Debug

MY OWN INSTALL PREFIX: Prefix prepended to install directories
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [q] to quit without generating
  
```

Liste des paramètres

```
cadiou@moon:~/project/build$ cmake ..  
[t]
```

```
Page 1 of 2  
CMAKE_AR                */usr/bin/ar  
CMAKE_BUILD_TYPE        *  
CMAKE_COLOR_MAKEFILE    *ON  
CMAKE_CXX_COMPILER       */usr/bin/c++  
CMAKE_CXX_FLAGS          *  
CMAKE_CXX_FLAGS_DEBUG    *-g  
CMAKE_CXX_FLAGS_MINSIZEREL *-Os -DNDEBUG  
CMAKE_CXX_FLAGS_RELEASE  *-O3 -DNDEBUG  
CMAKE_CXX_FLAGS_RELWITHDEBINFO *-O2 -g -DNDEBUG  
CMAKE_EXE_LINKER_FLAGS   *  
CMAKE_EXE_LINKER_FLAGS_DEBUG *  
CMAKE_EXE_LINKER_FLAGS_MINSIZE *  
CMAKE_EXE_LINKER_FLAGS_RELEASE *  
CMAKE_EXE_LINKER_FLAGS_RELWITH *  
CMAKE_EXPORT_COMPILE_COMMANDS *OFF  
CMAKE_INSTALL_PREFIX     */usr/local  
CMAKE_LINKER             */usr/bin/ld  
CMAKE_MAKE_PROGRAM       */usr/bin/make  
CMAKE_MODULE_LINKER_FLAGS *  
CMAKE_MODULE_LINKER_FLAGS_DEBU *  
CMAKE_MODULE_LINKER_FLAGS_MINS *  
CMAKE_MODULE_LINKER_FLAGS_RELE *  
CMAKE_MODULE_LINKER_FLAGS_RELW *  
CMAKE_AR: Path to a program.  
Press [enter] to edit option  
Press [c] to configure  
Press [h] for help  
CMake Version 2.8.12.2  
Press [q] to quit without generating
```



```

CMAKE_NM                */usr/bin/nm
CMAKE_OBJCOPY            */usr/bin/objcopy
CMAKE_OBJDUMP            */usr/bin/objdump
CMAKE_RANLIB              */usr/bin/ranlib
CMAKE_SHARED_LINKER_FLAGS *
CMAKE_SHARED_LINKER_FLAGS_DEBU *
CMAKE_SHARED_LINKER_FLAGS_MINS *
CMAKE_SHARED_LINKER_FLAGS_RELE *
CMAKE_SHARED_LINKER_FLAGS_RELW *
CMAKE_SKIP_INSTALL_RPATH */OFF
CMAKE_SKIP_RPATH         */OFF
CMAKE_STATIC_LINKER_FLAGS *
CMAKE_STATIC_LINKER_FLAGS_DEBU *
CMAKE_STATIC_LINKER_FLAGS_MINS *
CMAKE_STATIC_LINKER_FLAGS_RELE *
CMAKE_STATIC_LINKER_FLAGS_RELW *
CMAKE_STRIP              */usr/bin/strip
CMAKE_USE_RELATIVE_PATHS */OFF
CMAKE_VERBOSE_MAKEFILE   */OFF

```

Page 2 of 2

```

CMAKE NM: Path to a program.
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently On)

```

CMake Version 2.8.12.2

Compilation conditionnelle

```
1 #include <iostream>
2
3 int main()
4 {
5     #ifdef NL
6         std::cout << "'Hoi, wereld!'\n";
7     #elif TK
8         std::cout << "'Selam Dünya!'\n";
9     #else
10         std::cout << "'Hello, world!'\n";
11 #endif
12 }
```

Compilation manuelle

```
cadiou@moon:~/project/src$ g++ prog.cpp
cadiou@moon:~/project/src$ ./a.out
'Hello, world!'
```

```
cadiou@moon:~/project/src$ g++ -DTK prog.cpp
cadiou@moon:~/project/src$ ./a.out
'Selam Dünya!'
```

avec CMake

Ajouter un test

```
# parameters
include(CMakeFunction.txt)
option(MULTI_LANG "Multiple language" OFF)
set(LanguageValues "anglais")
if (MULTI_LANG)
    set(LanguageValues "anglais;turc;neerlandais" CACHE STRING
        "List of possible languages")
    choice(Language "${LanguageValues}" "Language chosen by
        the user at
CMake configure time")
endif ()
message(STATUS "Language='${Language}')"

if("${Language}" STREQUAL "turc")
    add_definitions(-DTK)
endif()
if("${Language}" STREQUAL "neerlandais")
    add_definitions(-DNL)
endif()
```

Fonction pour sélectionner un choix

```
function(choice varname choices documentation)
  if("${varname}" STREQUAL "" OR "${choices}" STREQUAL "")
    message(FATAL_ERROR "choice function requires varname
      and choices values")
  endif()

  set(default_index ${ARGV3})
  if(NOT default_index)
    set(default_index 0)
  endif()

  list(GET choices ${default_index} default_value)

  set(${varname} ${default_value} CACHE STRING ${
    documentation})
  set_property(CACHE ${varname} PROPERTY STRINGS ${choices})
endfunction()
```

Exemple

```
cadiou@moon:~/project/build$ cmake ..
```

```
Page 1 of 1
CMAKE_BUILD_TYPE          *
CMAKE_INSTALL_PREFIX      */usr/local
MULTI_LANG                 *OFF

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)
```

```
Page 1 of 1
CMAKE_BUILD_TYPE      Release
CMAKE_INSTALL_PREFIX  ..
MULTI_LANG             OFF

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option                                CMake Version 2.8.12.2
Press [c] to configure      Press [g] to generate and exit
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

```
Page 1 of 1
CMAKE_BUILD_TYPE      Release
CMAKE_INSTALL_PREFIX  ..
MULTI_LANG             ON

MULTI LANG: Multiple language
Press [enter] to edit option
Press [c] to configure  it
Press [h] for help      Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

CMake Version 2.8.12.2
```



```
Page 1 of 1
Language          *anglais
LanguageValues    *anglais;turc;neerlandais
CMAKE_BUILD_TYPE  Release
CMAKE_INSTALL_PREFIX ..
MULTI_LANG        ON

Language: Language chosen by the user at
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)

CMake Version 2.8.12.2
```

```
Page 1 of 1
Language          *turc
LanguageValues    *anglais;turc;neerlandais
CMAKE_BUILD_TYPE  Release
CMAKE_INSTALL_PREFIX ..
MULTI_LANG        ON

Language: Language chosen by the user at
Press [enter] to edit option
Press [c] to configure
Press [h] for help
Press [t] to toggle advanced mode (Currently Off)

CMake Version 2.8.12.2
```

```
Page 1 of 1
CMAKE_BUILD_TYPE          Release
CMAKE_INSTALL_PREFIX      ..
Language                  turc
LanguageValues             anglais;turc;neerlandais
MULTI_LANG                 ON

CMAKE BUILD TYPE: Choose the type of build, options are: None(CMAKE CXX FLAGS or C
Press [enter] to edit option                                     CMake Version 2.8.12.2
Press [c] to configure      Press [g] to generate and exit
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Remarques

- Construire un exécutable via CMake s'avère utile dans le cas de projet avec de **nombreuses dépendances**, de nombreuses **options**, éventuellement en compilation conditionnelle.
- CMake facilite le portage **multi-plateformes** en découplant les sources, les instructions de construction et la réalisation de l'exécutable.
- Sur une même plateforme, cela facilite la construction de différents exécutables, suivant les compilateurs disponibles ou les options applicatives souhaitées par l'utilisateur.
- CTest permet l'automatisation des tests unitaires au moment de la construction.

Table des matières

1 Contexte

2 Exemple simple

3 Exemple avec lien

4 Paramètres

5 CTest

6 Conclusion

CTest

CTest utilise le même fichier **CMakeLists.txt**
(comme les tests définis dans le Makefile)

CTest organise la vérification des tests unitaires. Il permet de retourner la valeur **0** lorsque le test est effectué avec succès et renvoie une autre information dans le cas contraire. Le programme testé peut être

- un binaire
- un script
- ...

Réaliser des **tests unitaires** participe d'une démarche d'intégration continue et permet de **valider la non-régression du code** au cours des développements. Ils doivent être conçus avant l'intégration dans le code.

Syntaxe

Se rajoute aisément au fichier **CMakeLists.txt**

```
ENABLE_TESTING()  
  
ADD_TEST( testname testexecutable args )
```

```
project/  
├── src/  
│   └── prog.cpp  
└── test/  
    ├── prog_syntaxe.cpp  
    ├── ref_test_uk.log  
    └── test_uk.sh
```

Exemple

Teste la conformité des résultats sur un cas simple.

Modification du code :

```
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "Hello, word!\n";
6 }
```

Le résultat est-il modifié par cette évolution du code ?

CMakeLists.txt à la racine du projet

```
# cmake version
cmake_minimum_required(VERSION 2.8.10 FATAL_ERROR)

# define project name and langage
project(MyProject CXX)
include(CTest)
enable_testing()

# define executable names
set(EXE MyProgram)

# define directories
set(SRC      ${CMAKE_SOURCE_DIR}/src)
set(TESTS    ${CMAKE_SOURCE_DIR}/test)
add_subdirectory(${SRC})
add_subdirectory(${TESTS})
```

CMakeLists.txt associé aux sources

```
# set up the source files compilation

# add the source files
set(SRC
    prog.cpp
)

# define the executable in terms of the source files
add_executable(${EXE} ${SRC})

# how to install this executable
install(TARGETS ${EXE} RUNTIME DESTINATION bin)
```

CMakeLists.txt associé aux tests (1/2)

```
# set up the source files compilation

# add the source files
set(TEST_FILES
    test_uk.sh
    ref_test_uk.log
)

# copy into build directory
foreach(file ${TEST_FILES})
    add_custom_command(#
        OUTPUT "${CMAKE_CURRENT_BINARY_DIR}/${file}"
        COMMAND cmake -E copy "${CMAKE_CURRENT_SOURCE_DIR}/${file}"
            "${CMAKE_CURRENT_BINARY_DIR}/${file}"
        DEPENDS "${CMAKE_CURRENT_SOURCE_DIR}/${file}"
    )
    list(APPEND file_dest "${CMAKE_CURRENT_BINARY_DIR}/${file}"
        )
endforeach(file)
```

CMakeLists.txt associé aux tests (2/2)

```
# define the executable in terms of the source files
add_executable(test_syntaxe prog_syntaxe.cpp)
set_target_properties(test_syntaxe PROPERTIES
    LINKER_LANGUAGE CXX)

# add test
add_custom_target(test_uk.sh ALL DEPENDS ${file_dest})

add_test(NAME test_syntaxe COMMAND ./test_syntaxe)

add_test(NAME test_uk COMMAND ./test_uk.sh)
```

CMake/CTest

```
cadiou@moon:~/project$ mkdir build; cd build
cadiou@moon:~/project/build$ cmake ..
cadiou@moon:~/project/build$ make
canning dependencies of target MyProgram
[ 33%] Building CXX object src/CMakeFiles/MyProgram.dir/prog
.cpp.o
Linking CXX executable MyProgram
[ 33%] Built target MyProgram
Scanning dependencies of target test_syntaxe
[ 66%] Building CXX object test/CMakeFiles/test_syntaxe.dir/
prog_syntaxe.cpp.o
Linking CXX executable test_syntaxe
[ 66%] Built target test_syntaxe
Scanning dependencies of target test_uk.sh
[100%] Generating test_uk.sh
[100%] Built target test_uk.sh
cadiou@moon:~/project/build$ make test
```

Tests validés (programme inchangé)

```
cadiou@moon:~/project/build$ make test
Running tests...
Test project ~/project/build
  Start 1: test_syntaxe
1/2 Test #1: test_syntaxe ..... Passed
    0.00 sec
  Start 2: test_uk
2/2 Test #2: test_uk ..... Passed
    0.01 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =  0.01 sec
```

... avec les modifications du programme

```

cadiou@moon:~/project/build$ make test
Running tests...
Test project ~/project/build
  Start 1: test_syntaxe
1/2 Test #1: test_syntaxe ..... Passed
    0.00 sec
  Start 2: test_uk
2/2 Test #2: test_uk .....***Failed
    0.01 sec

50% tests passed, 1 tests failed out of 2

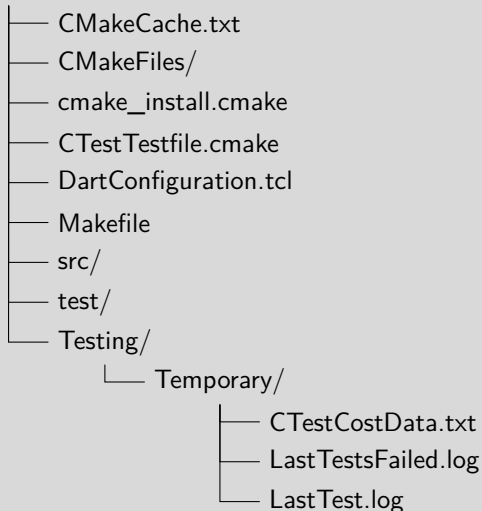
Total Test time (real) =  0.01 sec

The following tests FAILED:
  2 - test_uk (Failed)
Errors while running CTest

```

Contenu du répertoire build

project/build/



Contenu du répertoire build

```
command: "~/project/build/test/test_uk.sh"
```

```
Directory: ~/project/build/test
```

```
"test_uk" start time: Mar 09 10:30 CET
```

```
Output:
```

```
-----  
run test uk ...
```

```
1c1
```

```
< Hello, world!
```

```
---
```

```
> Hello, word!
```

```
... done
```

```
<end of output>
```

```
Test time = 0.01 sec
```

```
-----  
Test Failed.
```

```
"test_uk" end time: Mar 09 10:30 CET
```

```
"test_uk" time elapsed: 00:00:00  
-----
```

```
End testing: Mar 09 10:30 CET
```

Remarques

- Les programmes de test sont différents du code.
- Ils peuvent être écrits dans un autre langage.
- Ils servent à vérifier et à valider le code.
- Il est préférable de les écrire (ou tout au moins les concevoir) avant le développement prévu.
- Ils peuvent tester une application du code sur un cas simple, pour vérifier la conformité du résultat, la vitesse d'exécution, etc.
- Les tests archivent les résultats de référence.

Table des matières

- 1 Contexte
- 2 Exemple simple
- 3 Exemple avec lien
- 4 Paramètres
- 5 CTest
- 6 Conclusion**

Conclusion

CMake/CTest sont des alternatives intéressantes aux Makefiles

La construction de l'exécutable n'**interfère pas avec les sources** (il est facile d'exclure le répertoire build d'un suivi sous git par exemple)

Dans le cadre de développements pour une application de calcul scientifique, il est souvent intéressant d'adopter une démarche **agile**, impliquant au minimum une **automatisation** des étapes suivantes :

- un **suivi des versions**
- une méthode de construction de l'exécutable facilitant le portage par la **recherche automatique des dépendances**
- une méthode d'**intégration continue** validant la non-régression du code, orienté sur les applications