

Systèmes de construction logicielle

Introduction à cmake

F. Langrognnet



(Lm^B)

laboratoire de mathématiques de besançon
UNIVERSITÉ DE FRANCHE-COMTÉ • CNRS • UMR 6623



PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

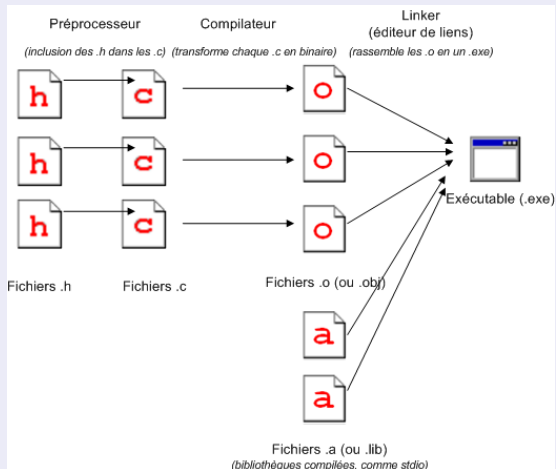
Objectifs des systèmes de construction logicielle

Définition

- Les **systèmes de construction logicielle** (build systems ou encore moteur de production) sont des logiciels dont l'objectif principal est d'**automatiser**, directement ou indirectement, **le processus de compilation** d'un code source voire la distribution des produits logiciels.
- Wikipedia :
Un moteur de production est un logiciel dont la fonction principale consiste à automatiser l'ensemble des actions (préprocessing, compilation, éditions des liens, etc.) contribuant, à partir de données sources, à la production d'un ensemble logiciel opérationnel.

Compilation, éditions de liens

Processus



Pourquoi utiliser un système de construction logicielle ?

Quelques motivations...

- Pouvoir compiler, construire l'exécutable dans des contextes différents (OS, compilateurs, ...)
- Ne recompiler que le strict minimum
- Disposer d'outils complémentaires : création de packages (linux ou Mac), ou d'installateurs pour windows

Liens avec les IDE

Les principaux IDE intègrent certains de ces outils :

- KDevelop
- Eclipse CDT
- Visual Studio
- Code : :Blocks

PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

PLAN

1 Objectifs des systèmes de construction logicielle

2 **cmake**

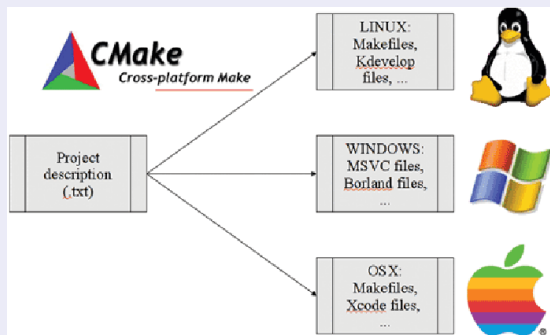
- **Introduction**
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

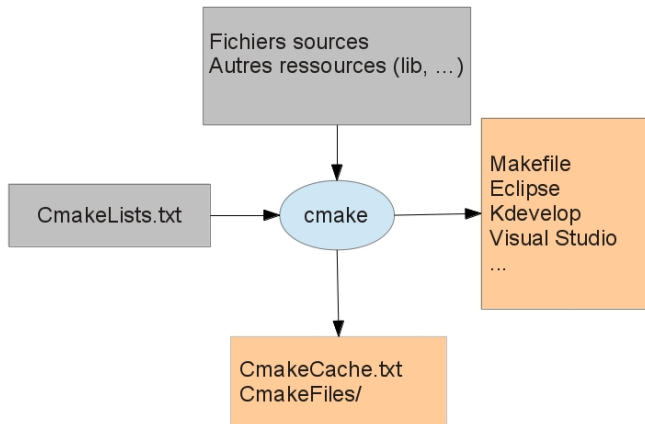
- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

Logiciel de construction logicielle

- multilingage
- multiplate-forme
- licence libre



cmake - Principe général



PLAN

1 Objectifs des systèmes de construction logicielle

2 **cmake**

- Introduction
- **Langage cmake et utilisation**
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

Langage de cmake

Langage cmake

cmake possède son propre langage (fichiers **CMakeLists.txt**) avec :

- variables
- listes
- conditions : IF
- boucles : FOREACH

Syntaxe

Ceci est un commentaire

command(argument1 argument2 ... argumentN)

COMMAND(argument1 argument2 ... argumentN)

Utilisation de cmake

cmake par défaut

Une fois les fichiers CMakeLists.txt créés, il suffit de lancer cmake :
\$ cmake .

Par défaut, un **Makefile** sera créé

Il suffit ensuite de taper les commandes suivantes pour construire et installer le projet :

\$ make

\$make install

cmake : générateur de configuration pour IDE

Syntaxe :

\$ cmake . -G <generator>

Exemples :

\$ cmake . -G "KDevelop3"

\$ cmake . -G "CodeBlocks"

PLAN

1 Objectifs des systèmes de construction logicielle

2 `cmake`

- Introduction
- Langage `cmake` et utilisation
- **makefile, make, `cmake`**

3 Exemples avec `cmake`

- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

Le couple makefile, make

Les **Makefiles** sont des fichiers utilisés par le programme **make** pour exécuter un ensemble d'actions, comme la compilation d'un projet, l'archivage de document, la mise à jour de site, ...

Exemple de makefile

```
Makefile
CC=gcc
CFLAGS=-W -Wall -ansi -pedantic
LDFLAGS=
EXEC=hello
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)

all: $(EXEC)

hello: $(OBJ)
      $(CC) -o $@ $^ $(LDFLAGS)

main.o: hello.h

%.o: %.c
      $(CC) -o $@ -c $< $(CFLAGS)

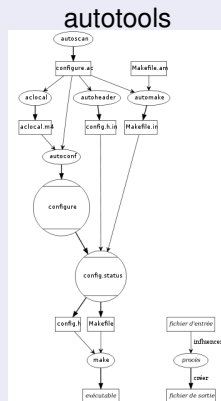
.PHONY: clean mrproper

clean:
      rm -rf *.o

mrproper: clean
      rm -rf $(EXEC)
```

Comment créer des makefiles ?

- A la main
- Avec les autotools (1990)
- Avec des "nouveaux" générateurs
 - ▶ imake
 - ▶ qmake
 - ▶ nmake
 - ▶ **cmake** (1999)



PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- Construction d'une bibliothèque
- Création de packages

PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

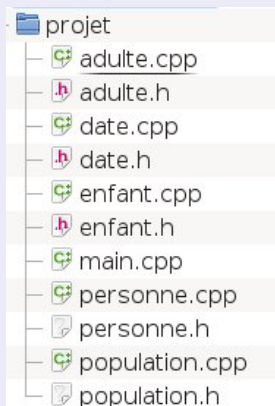
- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- **Construction d'un exécutable**
- Construction d'une bibliothèque
- Création de packages

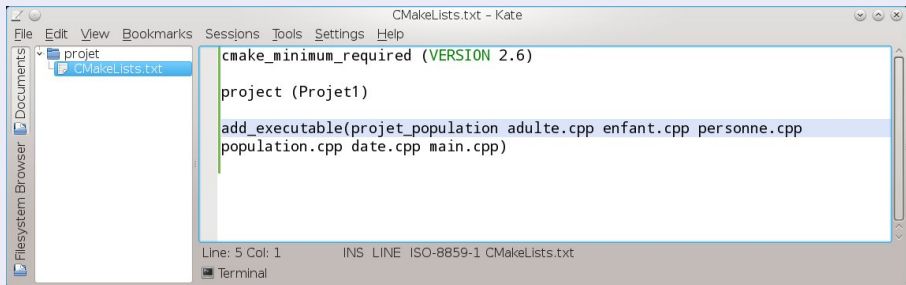
Exemple 1 : Construction d'un simple exécutable

Arborescence du projet



Exemple 1

CMakeLists.txt



```
cmake_minimum_required (VERSION 2.6)

project (Projet1)

add_executable(projet_population adulte.cpp enfant.cpp personne.cpp
population.cpp date.cpp main.cpp)
```

Line: 5 Col: 1 INS LINE ISO-8859-1 CMakeLists.txt

Terminal

Construction

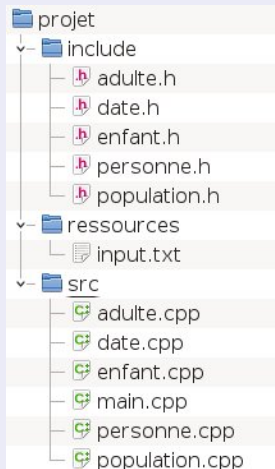
`$ cmake .`

`$ make`

Création de l'exécutable projet_population


Exemple 2 : Construction d'un simple exécutable avec une arborescence

Arborescence du projet



Exemple 2

CMakeLists.txt



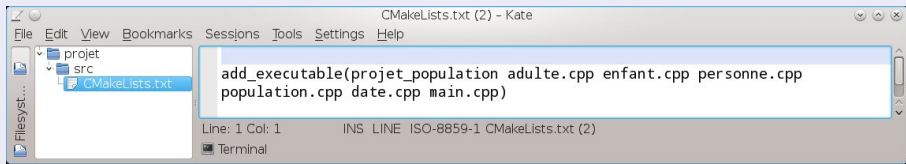
A screenshot of a text editor window titled "CMakeLists.txt - Kate". The window shows a file explorer on the left with a folder named "projet" containing a file "CMakeLists.txt". The main editor area displays the following CMake code:

```
cmake_minimum_required (VERSION 2.6)

project (Projet1)

add_subdirectory(src)
```

The status bar at the bottom indicates "Line: 5 Col: 7" and "INS LINE ISO-8859-1 CMakeLists.txt". A "Terminal" icon is visible in the bottom left corner.



A screenshot of a text editor window titled "CMakeLists.txt (2) - Kate". The window shows a file explorer on the left with a folder named "projet" containing a subfolder "src" and a file "CMakeLists.txt". The main editor area displays the following CMake code:

```
add_executable(projet_population adulte.cpp enfant.cpp personne.cpp
population.cpp date.cpp main.cpp)
```

The status bar at the bottom indicates "Line: 1 Col: 1" and "INS LINE ISO-8859-1 CMakeLists.txt (2)". A "Terminal" icon is visible in the bottom left corner.

Construction

\$ cmake .

\$ make

Exemple 3 : automatiser l'installation

CMakeLists.txt



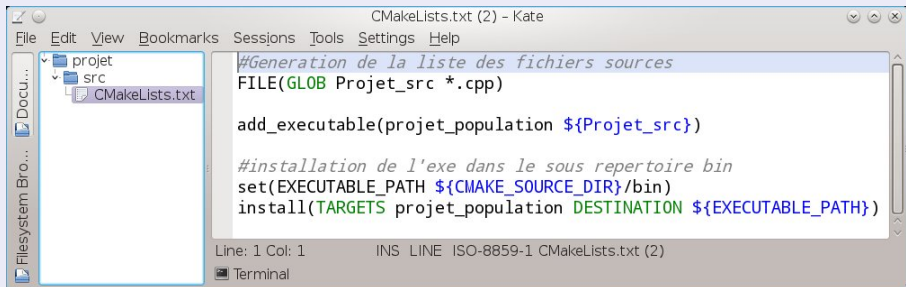
The screenshot shows the Kate editor with the file 'CMakeLists.txt' open. The file content is as follows:

```
cmake_minimum_required (VERSION 2.6)

project (Projet1)

add_subdirectory(src)
```

The status bar at the bottom indicates 'Line: 5 Col: 7' and 'INS LINE ISO-8859-1 CMakeLists.txt'. A 'Terminal' icon is visible in the bottom left corner.



The screenshot shows the Kate editor with the file 'CMakeLists.txt (2)' open. The file content is as follows:

```
#Generation de la liste des fichiers sources
FILE(GLOB Projet_src *.cpp)

add_executable(projet_population ${Projet_src})

#installation de l'exe dans le sous repertoire bin
set(EXECUTABLE_PATH ${CMAKE_SOURCE_DIR}/bin)
install(TARGETS projet_population DESTINATION ${EXECUTABLE_PATH})
```

The status bar at the bottom indicates 'Line: 1 Col: 1' and 'INS LINE ISO-8859-1 CMakeLists.txt (2)'. A 'Terminal' icon is visible in the bottom left corner.

Exemple 3 (suite)

Construction - Installation

Construction et installation de l'exécutable projet_population dans le sous-répertoire bin

```
$ cmake .  
$ make  
$make install
```


PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

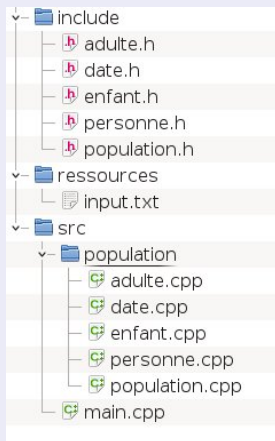
- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- **Construction d'une bibliothèque**
- Création de packages

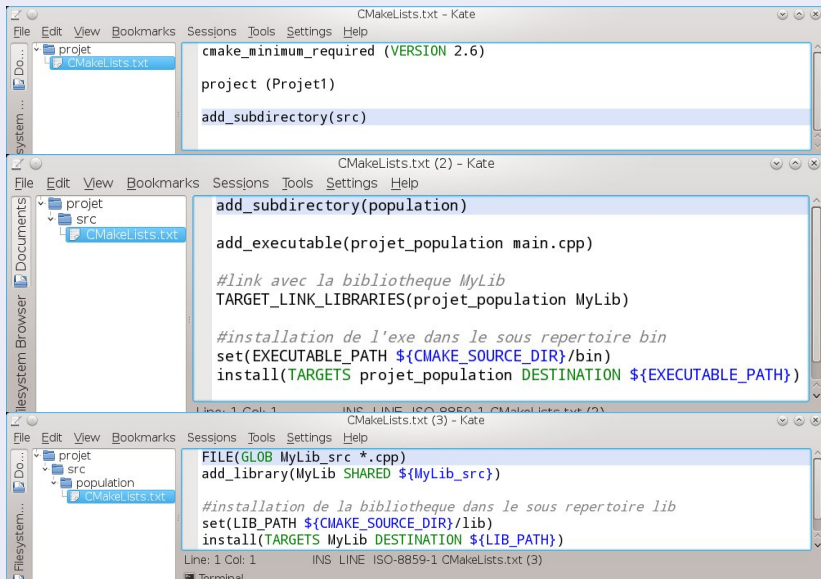
Construction d'une bibliothèque et d'un exécutable

Arborescence du projet



Bibliothèque et exécutable

CMakeLists.txt



The image shows three overlapping screenshots of a CMakeLists.txt file in a text editor. The top window shows the initial configuration, the middle window shows the executable and library linking, and the bottom window shows the library source files and installation paths.

```
cmake_minimum_required (VERSION 2.6)

project (Projet1)

add_subdirectory(src)
```

```
add_subdirectory(population)

add_executable(projet_population main.cpp)

#link avec la bibliotheque MyLib
TARGET_LINK_LIBRARIES(projet_population MyLib)

#installation de l'exe dans le sous repertoire bin
set(EXECUTABLE_PATH ${CMAKE_SOURCE_DIR}/bin)
install(TARGETS projet_population DESTINATION ${EXECUTABLE_PATH})
```

```
FILE(GLOB MyLib_src *.cpp)
add_library(MyLib SHARED ${MyLib_src})

#installation de la bibliotheque dans le sous repertoire lib
set(LIB_PATH ${CMAKE_SOURCE_DIR}/lib)
install(TARGETS MyLib DESTINATION ${LIB_PATH})
```

Bibliothèque et exécutable (suite)

Construction - Installation

- Construction et installation de la bibliothèque libMyLib.so dans le sous-répertoire lib
- Construction et installation de l'exécutable projet_population dans le sous-répertoire bin

```
$ cmake .  
$ make  
$make install
```

PLAN

1 Objectifs des systèmes de construction logicielle

2 cmake

- Introduction
- Langage cmake et utilisation
- makefile, make, cmake

3 Exemples avec cmake

- Construction d'un exécutable
- Construction d'une bibliothèque
- **Création de packages**

Construction de packages avec cpack

cpack

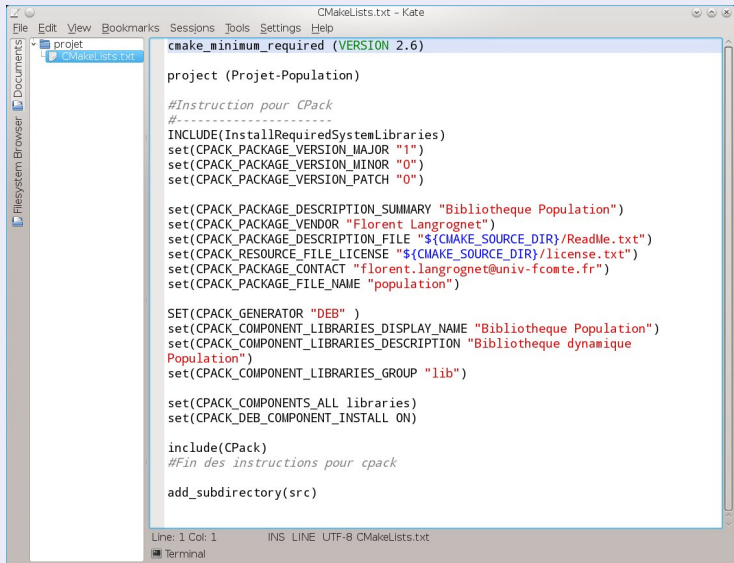
Moteur de création de paquets RPM, DEB, installateur Windows, ...

Peut s'utiliser

- de façon autonome (fichier CPackConfig.cmake)
- intégré à cmake (dans un CMakeLists.txt)

Création de package deb ave cpack

CMakeLists.txt



The image shows a screenshot of a text editor window titled "CMakeLists.txt - Kate". The window displays the content of a CMakeLists.txt file. The file starts with a CMake minimum required version of 2.6. It defines a project named "Projet-Population". There are several comments in French, including "#Instruction pour CPack" and "#Fin des instructions pour cpack". The file sets various CPack variables for versioning, description, vendor, and contact information. It also sets the generator to "DEB" and defines component libraries for "Bibliothèque Population" with a group of "lib". Finally, it includes CPack and adds a subdirectory named "src".

```
cmake_minimum_required (VERSION 2.6)

project (Projet-Population)

#Instruction pour CPack
#-----
INCLUDE(InstallRequiredSystemLibraries)
set(CPACK_PACKAGE_VERSION_MAJOR "1")
set(CPACK_PACKAGE_VERSION_MINOR "0")
set(CPACK_PACKAGE_VERSION_PATCH "0")

set(CPACK_PACKAGE_DESCRIPTION_SUMMARY "Bibliothèque Population")
set(CPACK_PACKAGE_VENDOR "Florent Langrognet")
set(CPACK_PACKAGE_DESCRIPTION_FILE "${CMAKE_SOURCE_DIR}/ReadMe.txt")
set(CPACK_RESOURCE_FILE_LICENSE "${CMAKE_SOURCE_DIR}/license.txt")
set(CPACK_PACKAGE_CONTACT "florent.langrognet@univ-fcomte.fr")
set(CPACK_PACKAGE_FILE_NAME "population")

SET(CPACK_GENERATOR "DEB" )
set(CPACK_COMPONENT_LIBRARIES_DISPLAY_NAME "Bibliothèque Population")
set(CPACK_COMPONENT_LIBRARIES_DESCRIPTION "Bibliothèque dynamique
Population")
set(CPACK_COMPONENT_LIBRARIES_GROUP "lib")

set(CPACK_COMPONENTS_ALL libraries)
set(CPACK_DEB_COMPONENT_INSTALL ON)

include(CPack)
#Fin des instructions pour cpack

add_subdirectory(src)
```

Line: 1 Col: 1 INS LINE UTF-8 CMakeLists.txt
Terminal

Création de package deb ave cpack

cpack

Création du fichier population-lib.deb (package pour debian, ubuntu)

```
$ cmake .
```

```
$ cpack
```

Ce package peut ensuite être installé avec

```
$ sudo dpkg -i population-lib.deb
```


FIN

Merci de votre attention