Bienvenue

Ingénieur en systèmes embarqués

Travail dans le domaine de l’embarqué pour des caméras de mésure

Enseignant sur Udemy

CMake Clarifié : Maîtrisez vos Projets C/C++

Luc Badinga
Ingénieur Systèmes embarqués

Sommaire

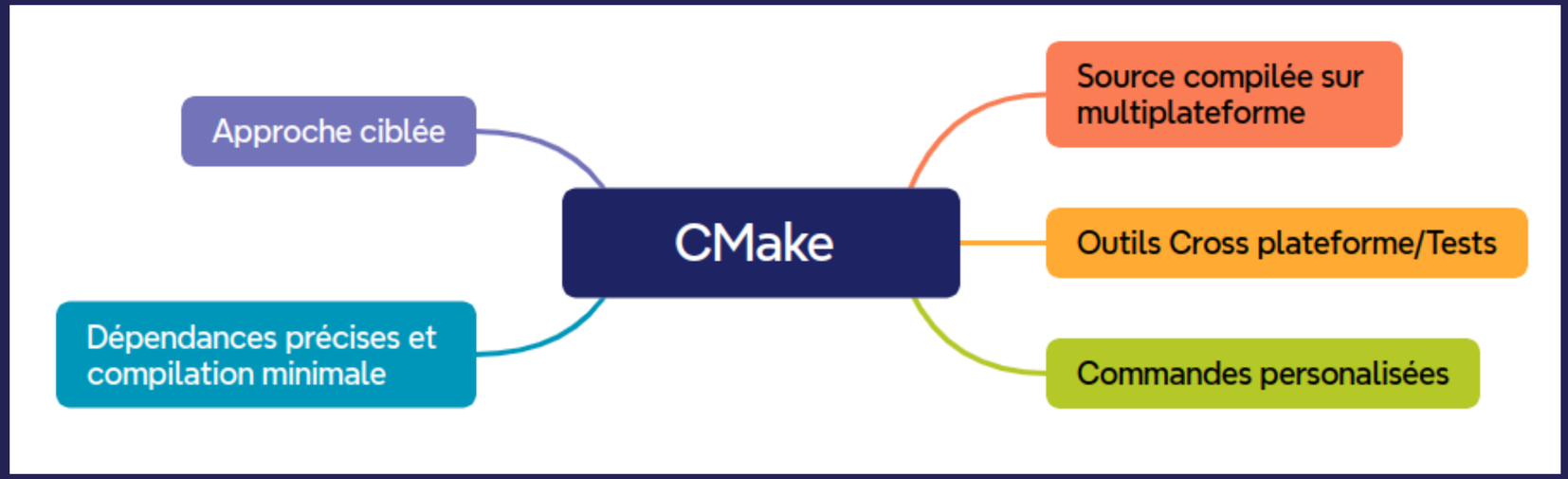
Introduction à CMake

1. **Installation**
2. **Cmake, Test et Outils**
    - **Module 1 - Les Fondamentaux CMake**
    - **Module 2 - Gestion de Projets simples et complexes avec CMake**
    - **Module 3 - Intégration Bibliothèque/librairie externe**
    - **Module 4 - Tests unitaires et et option d’optimisation**
    - **Module 5 - Projet template**
    - **Module 6 - Outils**

Conclusion et Ressources

Introduction à CMake

- Défintion CMake



- Pourquoi utiliser CMake et exemples des projets

1. Installation des outils

1.1 Installation sur Windows
1.2 Installation sur Linux
1.3 Installation sur Mac

1.1 Installation sur Windows

Les outils à installer:

Recommandés :

1. **Visual Studio Code**

https://code.visualstudio.com/Download

- **Installation kit MSVC**

***Attention** : Bien choisir le kit de développement*

https://learn.microsoft.com/en-us/cpp/build/vscpp-step-0-installation?view=msvc-170#step-4---choose-workloads

- **Installation WSL**

https://learn.microsoft.com/fr-fr/windows/wsl/install

installation identique sur Linux

2. **CMake**

https://cmake.org/download/

Optionnels:

- **Git ou GitDesktop**

https://git-scm.com/downloads

https://desktop.github.com/download/

- **Doxygen**

https://www.doxygen.nl/download.html

- **Python : Version 3.8 ou la plus reçente**

1.2 Installation sur Linux

Avant installation des outils:
sudo apt-get update
sudo apt-get upgrade

Recommandés:

- **sudo apt-get install gcc g++ gdb**
- **sudo apt-get install make cmake**
- **sudo apt-get install git**
- **sudo apt-get install doxygen**
- **sudo apt-get install python3 python3-pip**

Optionnels:

- **sudo apt-get install lcov gcovr**
- **sudo apt-get install ccache**
- **sudo apt-get install cppcheck**
- **sudo apt-get install llvm clang-format clang-tidy**
- **sudo apt-get install curl zip unzip tar**
- **sudo apt-get install graphviz**

1.3 Installation sur Mac

Les outils à installer sur Mac:

- **Xcode**

https://ics.uci.edu/~pattis/common/handouts/macclion/clang.html

  - **lldb --version**
  - **clang --version**
  - **clang++ --version**

- **Brew**
  - **Installation : https://www.igeeksblog.com/how-to-install-homebrew-on-mac/**
  - **Guide complet https://www.igeeksblog.com/how-to-install-homebrew-on-mac/**

Installation :

  - **brew install git**
  - **brew install make**
  - **brew install cmake**
  - **brew install doxygen**
  - **brew install lcov**
  - **brew install gcovr**
  - **brew install ccache**

1. Setup VScode avec WSL

Les outils à installer sur Windows:

- **Installer VS Code**

https://code.visualstudio.com/download

- **Installer WSL**

https://learn.microsoft.com/fr-fr/windows/wsl/install

- **Set up pour WSL dans VScode**

Ressources du cours

- **Dépôt Github**

https://github.com/mbadinga1/CoursCMake

- **slide du cours disponible**

Organisation du cours

- **VScode avec WSL**
- **Ressources sur Git**
- **slide du cours disponible**

Module 1 - Les Fondamentaux CMake

Mots clés :

CMake, CMakeLists.txt, cmake_minimum_required, add_executable, add_library, build, make

Objectifs:

- Etre capable de créer un projet basique avec CMake
- Connaître les commandes basiques pour lancer build et l'exécutable
- Apprentissage par projet - Conversion température

Module 2 : Gestion de Projets C/C++ - CMake

Mots clés :

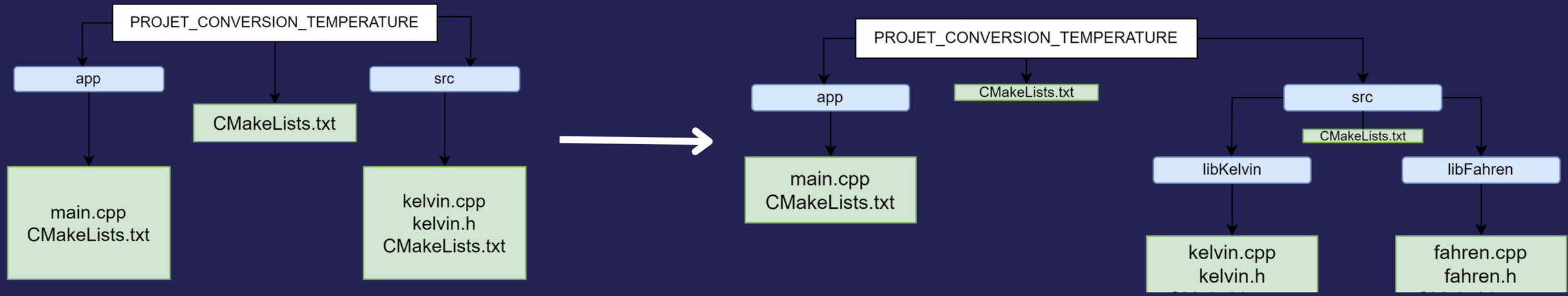
target_link_libraries, add_subdirectory,find_package, $, set

Objectifs:

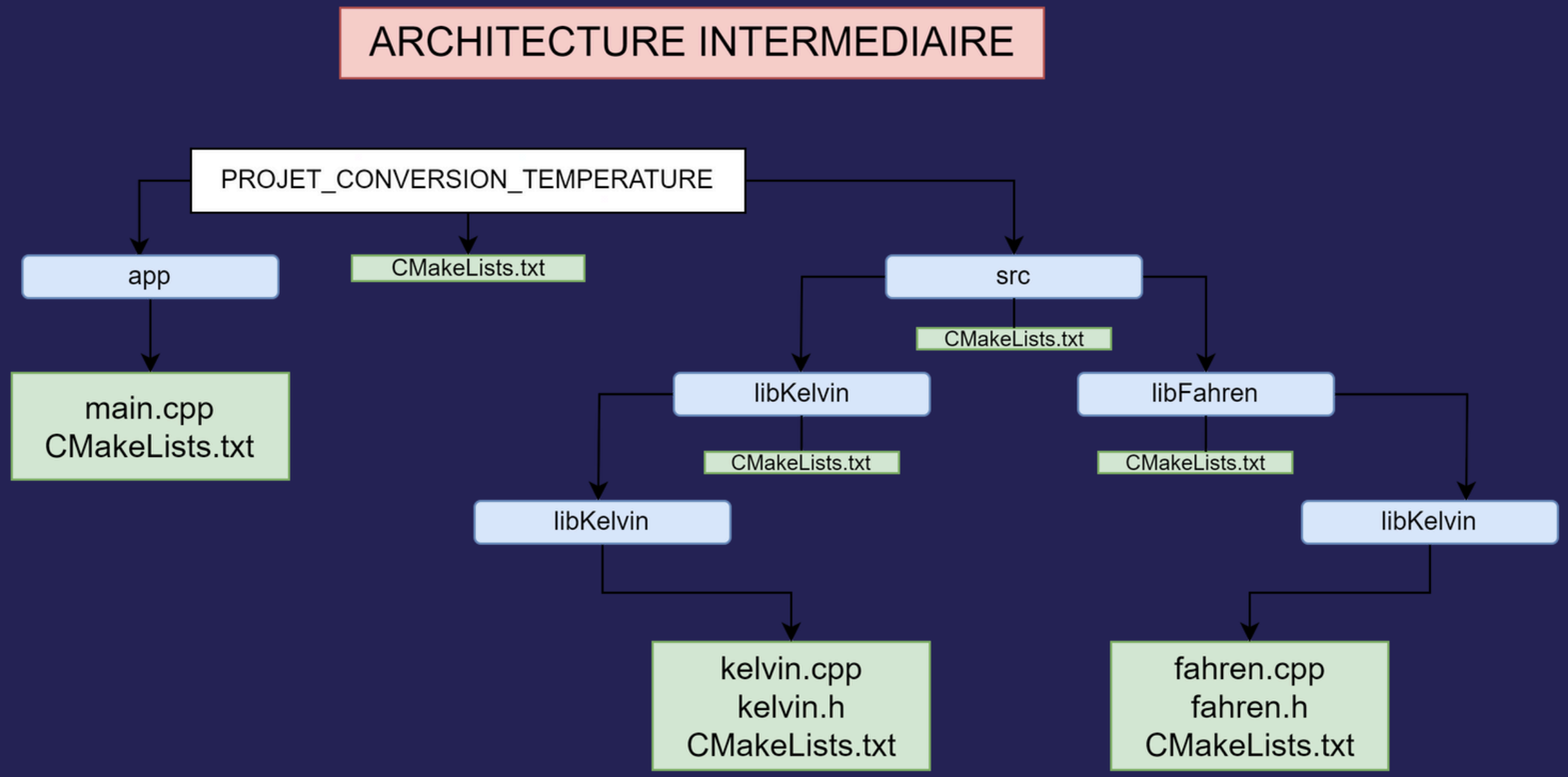
- Etre capable de créer un projet simple avec plusieurs dossiers
- Autre type d'architecture de projet
- Comprendre les variables et options de compilations
- Structurer avec les variables SOURCE et HEADER

Module 2 : Gestion de Projets C/C++ - CMake

ARCHITECTURE



Module 2 : Gestion de Projets C/C++ - CMake



Module 3 : Intégration Bibliothèque/librairie externe

Mots clés :

find_package, FetchContent,ExternalProject_Add, git submodule, Conan, Cpm, vcpkg

Objectifs:

- Etre capable d'ajouter une librairie externe à votre projet
- Comparaison et comment choisir

Module 3 : Intégration Bibliothèque/librairie externe

Conclusion

Module 3 : Comparaison - Libraires externes

GIT SUBMODULE	FETCHCONTENT	CPM (PACKAGE MANGER)	CONAN (PACKAGE MANGER)	VCPKG
git submodule add “repo” git submodule update --init --recursive Utile pour les projets non-cmake	include(FetchContent) FetchContent_Declare **Cas d’usage:** Projet CMake **Github/ Gitlab**	Release sur Github include(cmake/CPM.cmake) CPMAddPackage C**as d’usage:** Projet CMake **Github/ Gitlab**	pip install conan Cloner le repo Build le package conanfile.py **Inconvénients:** • Installation package, python	cloner le repo lancer .bat etc. **Inconvénients:** • gestion vcpkg.cmake • Récupération builtbaseline

Module 4 - Tests unitaires et d'autres options

- Tests Unitaires (Catch et GoogleTest) avec CTest
- Rappel variables CMake
- Ajouter les Warnings du compilateur
- Option de Compilation et Débogage

Ajouter les Warnings du compilateur

MSCV (Microsoft Visual C++)

Options	Description
/w /W0	Supprime tous les avertissements du compilateur
/W1 /W2 /W3 /W4	affiche les avertissements de niveau précédent et correspondant. Le niveau 1 étant un niveau grave
/Wall	Affiche tous les avertissements affichés par /W4 et tous les autres avertissements que /W4 exclut
/WX	Considère tous les avertissements du compilateur comme des erreurs
/permissive	le mode de conformité des normes au compilateur

GCC ou Clang

Options	Description
-Wall	Variable non initialisée, variable non utilisé, conversion suspecte, incohérence du nombre de paramètre dans les fonctions, etc.
-Wextra	Conversion entre unsigned et signed int, manque d'initialisation d'un des champs d'une variable struct, etc.
-Werror	Considère tous les avertissements du compilateur comme des erreurs
Wconversion, Wuninitialized	-

Variables importantes CMAKE

Variables	Description
CMAKE_BUILD_TYPE	Spécifier le type de build : Debug ou Release
CMAKE_MODULE_PATH	Variable permet de spécifier à cmake les chemins à inclure. Permet d'utiliser après include () ou find_package () pour inclure les modules
CMAKE_CURRENT_DIR	Chemin où se trouve les sources au top level
CMAKE_BINARY_DIR	Chemin où se trouve les binaires au top level
CMAKE_CURRENT_SOURCE_DIR	Chemin où se trouve les sources dans le dossier actuel
CMAKE_CURRENT_BINARY_DIR	Chemin où se trouve les binaires dans le dossier actuel
CMAKE_CXX_FLAGS	Flag pour différent langages utilisés pour le build à chaque configuration

Mots clés

Mots clés : PRIVATE, PUBLIC, INTERFACE

- Utiliser avec les commandes target_link_directories et target_include_directories
- Contrôler la visibilité et la propagation des options et des dépendances sur la cible actuelle et les autres

Mot clés	Cible actuelle	Propagation autres cibles
PRIVATE	OUI	NON
PUBLIC	OUI	OUI
INTERFACE	NON	OUI

Mots clés : STATIC, SHARED

- Utiliser avec la commande add_library
- Création bibliothèque soit statique, soit partagée(dynamique)
- Création des fichiers .so, .dll, .dylib

Type	SHARED	STATIC
Utilisation	Bibliothèque dynamique	Bibliothèque statique
Extension	.so (Linux), .dll (Windows), .dylib (MacOs)	.a (Linux/MacOs), .lib (Windows)
Chargement	Chargé à l'exécution (dynamique)	Incluse à la compilation(statique)

- Alternatif, utilisé la variable BUILD_SHARED_LIBS

Mots clés

PRIVATE, PUBLIC, INTERFACE

- Utiliser avec les commandes target_link_directories et target_include_directories
- Contrôler la visibilité et la propagation des options et des dépendances sur la cible actuelle et les autres

Mot clés	Cible actuelle	Propagation autres cibles
PRIVATE	OUI	NON
PUBLIC	OUI	OUI
INTERFACE	NON	OUI

Module 5 - CMake Avancé pour Projets C/C++

- Gestion des dépendances Externes
- Création de Modules et Scripts CMake
- Bonne pratiques et techniques avancées

Luc Badinga

Module 5 - Outils

Luc Badinga

Conclusion