Chapitre 7

Environnement de développement

GIT : gérer son code

Visual Studio Code : éditer son code

Visual Studio Code

DÉJÀ INSTALLÉ

GCC : compiler son code C++

```cpp
#include <iostream>

int main()
{
  std::cout << "Hello world" << std::endl;

  return 0;
}
```

```
> gcc .\main.cpp -lstdc++
```

OK, mais...

Comment on fait pour des projets de grande taille ?

Ninja : compiler des projets complexes

Ninja

(Makefile sous caféine)

https://ninja-build.org/

OK, mais...

```
ninja_required_version = 1.3

# The arguments passed to configure.py, for rerunning it.
configure_args =

root = .
builddir = build
cxx = g++
ar = ar
cflags = -g -Wall -Wextra -Wno-deprecated -Wno-missing-field-initializers $
    -Wno-unused-parameter -fno-rtti -fno-exceptions -fvisibility=hidden $
    -pipe '-DNINJA_PYTHON="python"' -O2 -DNDEBUG -DUSE_PPOLL $
    -DNINJA_HAVE_BROWSE -I.
ldflags = -L$builddir

rule cxx
  command = $cxx -MMD -MT $out -MF $out.d $cflags -c $in -o $out
  description = CXX $out
  depfile = $out.d
  deps = gcc

rule ar
  command = rm -f $out && $ar crs $out $in
  description = AR $out

rule link
  command = $cxx $ldflags -o $out $in $libs
  description = LINK $out

# browse_py.h is used to inline browse.py.
rule inline
  command = "$root/src/inline.sh" $varname < $in > $out
  description = INLINE $out
build $builddir/browse_py.h: inline $root/src/browse.py | $root/src/inline.sh
  varname = kBrowsePy
```

Comment écrire un tel fichier ?

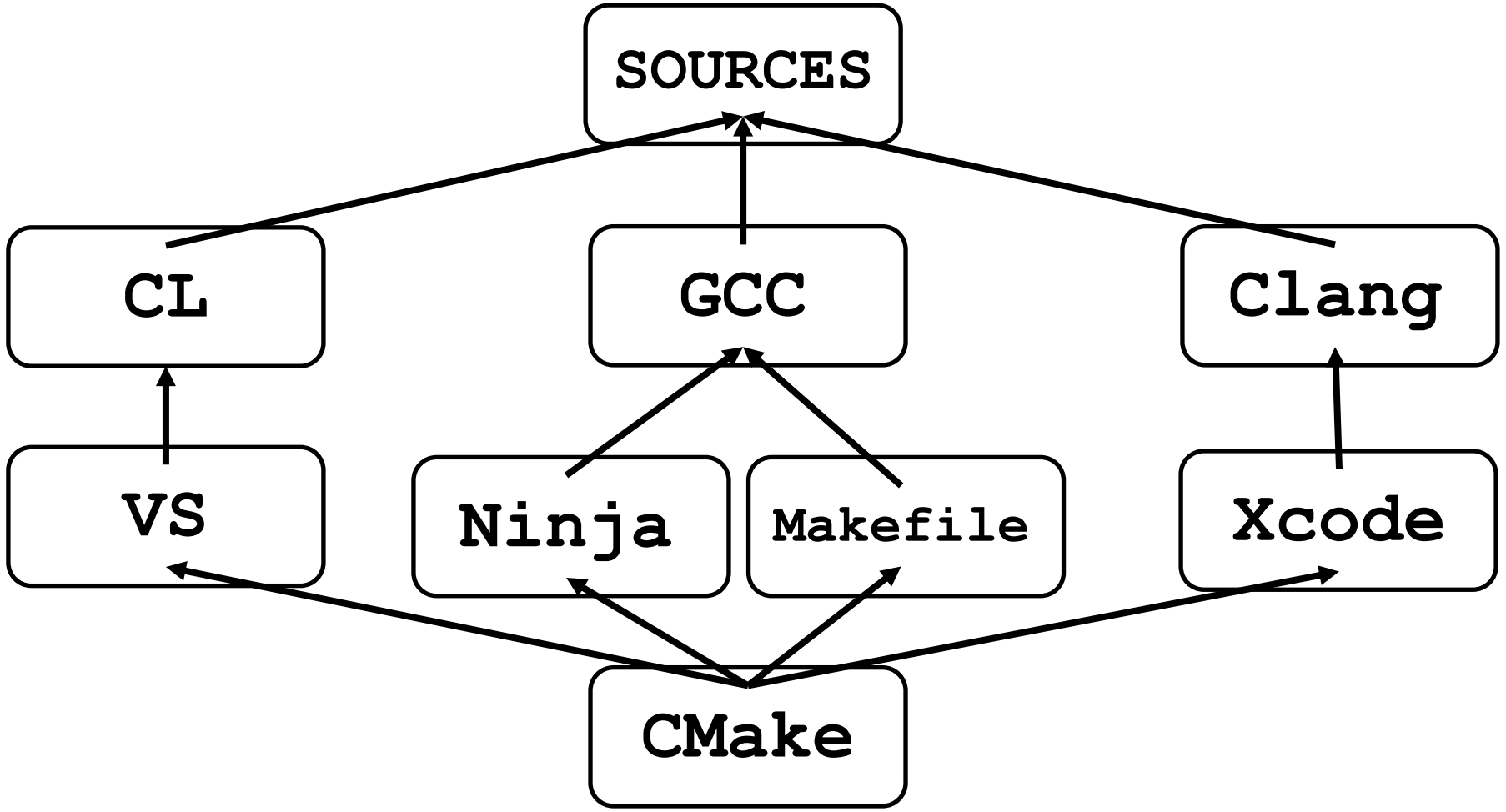
OK, mais...

Comment compiler le même projet, mais sur des plateformes différentes (Visual Studio / cl.exe, XCode / clang, Linux / GCC, etc.) ?

CMake

https://cmake.org/

Vue d'ensemble



Compilation séparée avec CMake

Pour **générer des projets** VS, CodeBlocks, Makefile, Ninja, ...

Stocker la configuration du projet dans un fichier

→ Versioning (avec Git par exemple)

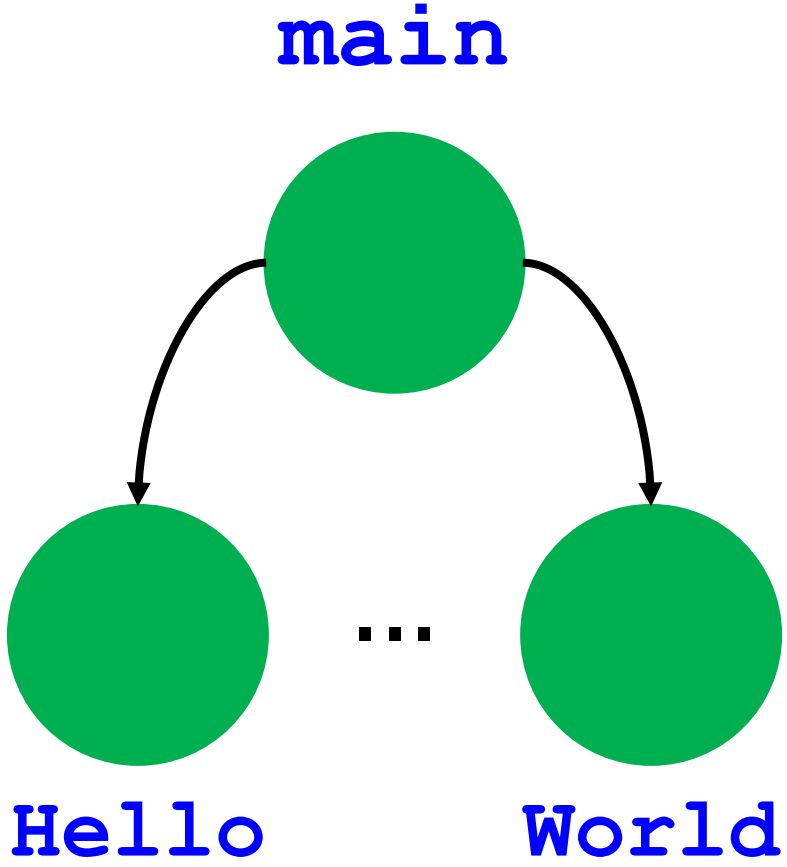
Gère les dépendances, tout comme Makefile

→ Compilation séparée avec modules et bibliothèques.

TUTO Compilation séparée avec CMake

Exemple

1. **Programme principale**
   → Affiche `Hello World`
2. **Module Hello (library)**
   → Affiche `Hello`
3. **Module World (library)**
   → Affiche `World`



TUTO Compilation séparée avec CMake

1. **télécharger le fichier Hello_CMake.zip + dézipper**
2. `mkdir BUILD` **(si non présent)**
3. `cd ./BUILD`
4. `cmake -G "Ninja" ../Sources`
5. `ninja`

TUTO Compilation séparée avec CMake

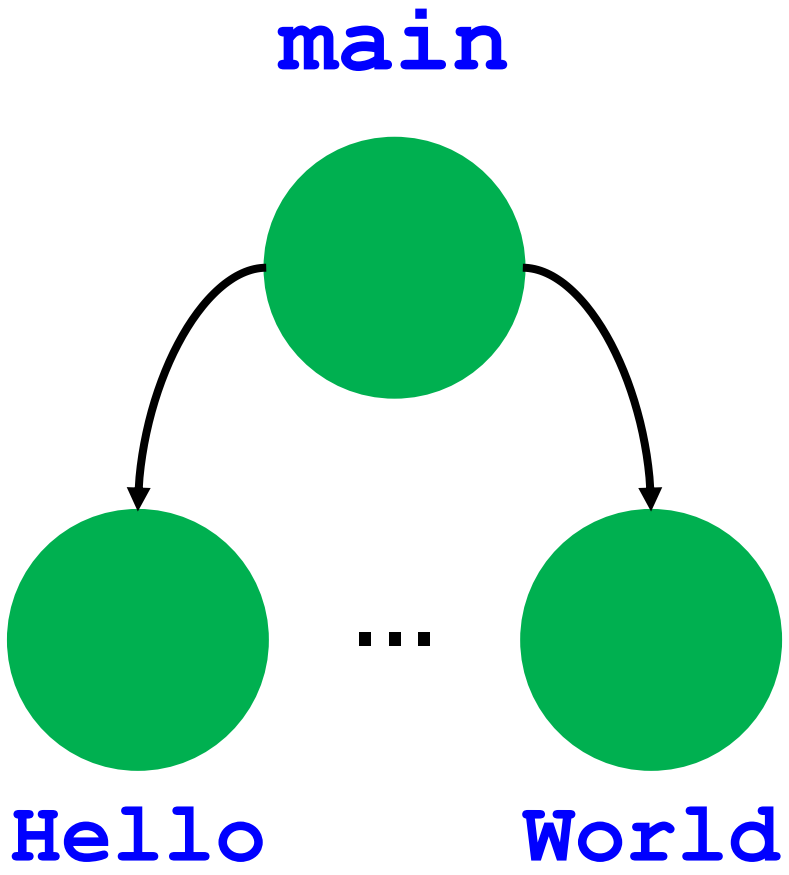
```
project(Hello_CMake)

add_library(2242.1_Hello hello.cpp hello.h)

add_library(2242.1_World world.cpp world.h)

add_executable(2242.1_Main main.cpp)

target_link_libraries(2242.1_Main 2242.1_Hello
2242.1_World)
```

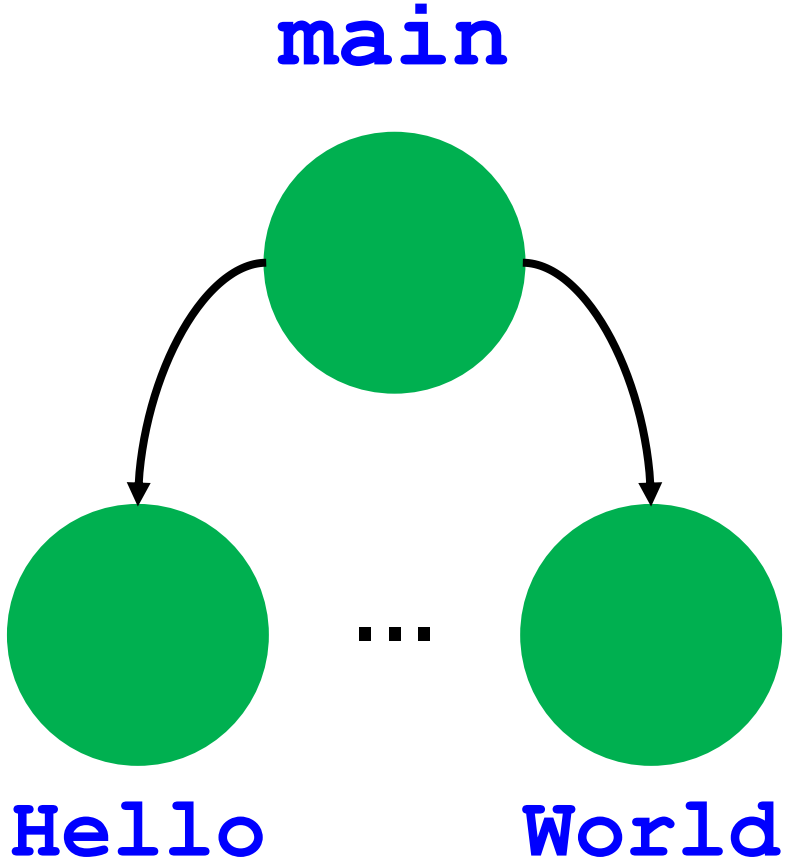


TUTO CMake est récursif

Commande

```
add_subdirectory(Hello)
add_subdirectory(World)
```

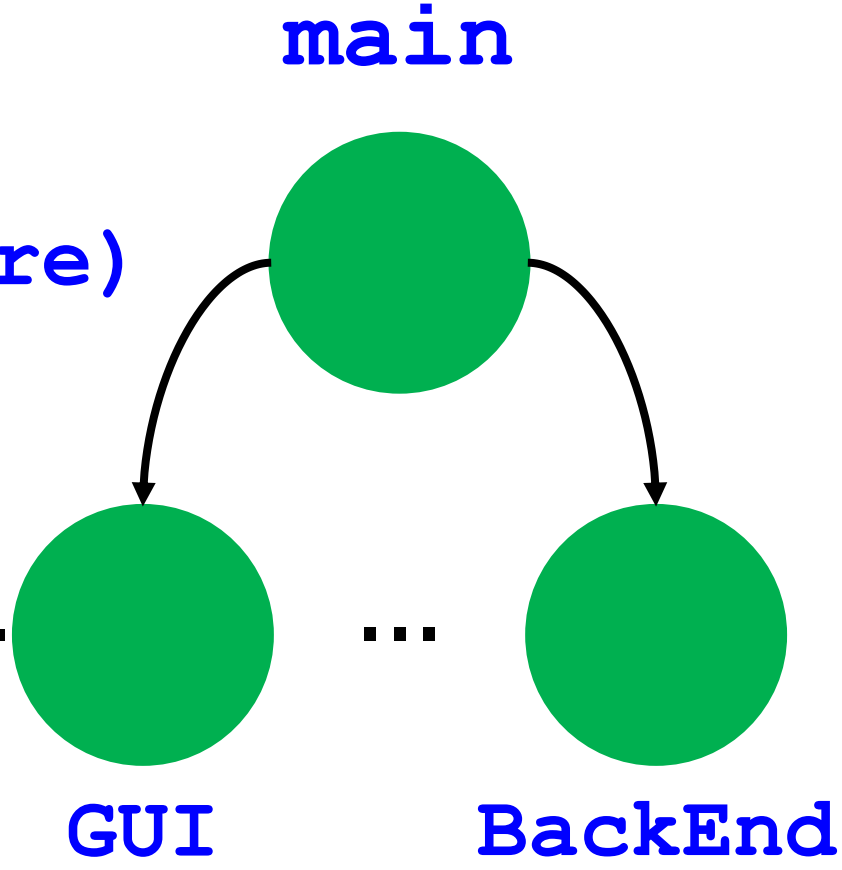
Exécute CMake dans ces sous-répertoires



EXERCICE

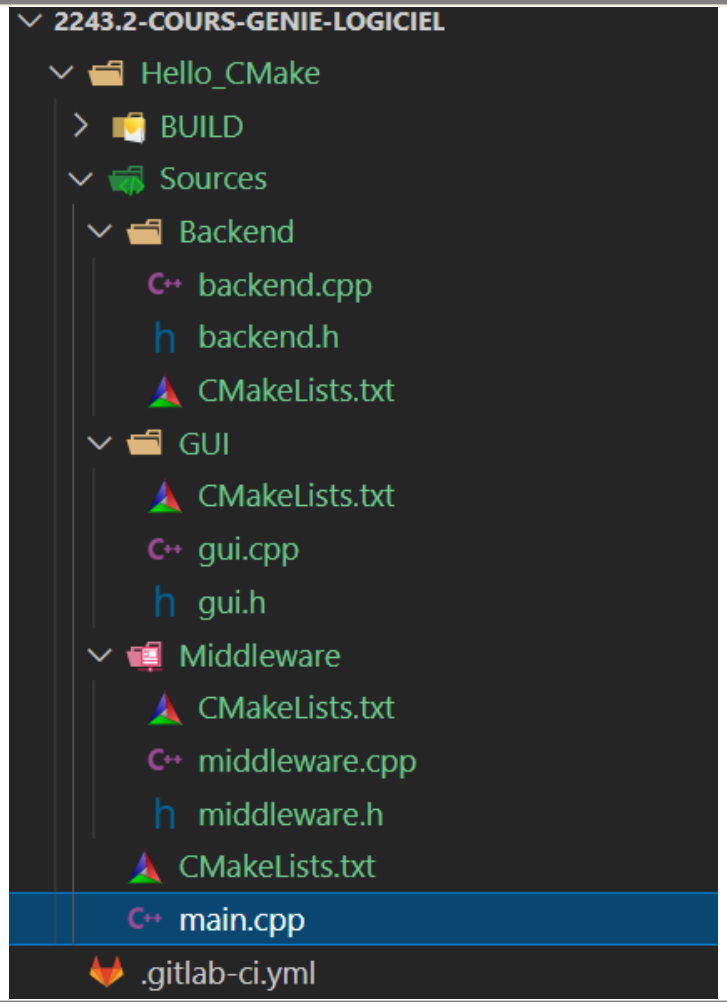
Commande

```
add_subdirectory(GUI)
add_subdirectory(Middleware)
add_subdirectory(BackEnd)
```



Exécute CMake dans ces sous-répertoires

EXERCICE




```cpp
#include "GUI/gui.h"
#include "Backend/Backend.h"
#include "Middleware/middleware.h"

#include <iostream>

int main()
{
    GUI gui;
    Middleware middleware;
    Backend backend;

    gui.Run();
    middleware.Run();
    backend.Run();

    return 0;
}
```

EXERCICE

OBJECTIF #1

Pouvoir exécuter CMake et Ninja en local

EXERCICE

OBJECTIF #2

Comprendre ce qui se passe quand on modifie un seul fichier