Utilisation de `make` et `cmake` pour un projet C++ simple

Enseignant en Sciences

6 février 2025

Objectifs de la séance

- Comprendre l'utilisation de `make` pour compiler un projet C++.
- Découvrir l'outil `cmake` pour la génération de Makefile.
- Illustrer ces outils sur un projet simple où les sources et headers se trouvent dans le répertoire `src`.

Organisation du projet

- Structure minimale du projet :
  - `src/main.cpp`
  - `src/fun.cpp`
  - `src/fun.h`
- Un Makefile ou un fichier `CMakeLists.txt` à la racine du projet.

Exemple d'utilisation de `make`

- Un Makefile permet d'automatiser la compilation.
- Il définit des cibles et leurs dépendances.
- Exemple : `main.o` dépend de `main.cpp` et d'un header.

Exemple de Makefile simple

```
# Nom de l'exécutable
EXEC = go

# Liste des sources
SRCS = src/main.cpp src/fun.cpp

# Liste des en-têtes
HDRS = src/fun.h

# Compilation et édition des liens
$(EXEC): main.o fun.o
    $(CXX) -o $(EXEC) main.o fun.o

# Règle pour main.o (dépend de main.cpp et fun.h)
main.o: src/main.cpp src/fun.h
    $(CXX) -c src/main.cpp

# Règle pour fun.o (dépend de fun.cpp et fun.h)
fun.o: src/fun.cpp src/fun.h
    $(CXX) -c src/fun.cpp

# Option de nettoyage
clean:
    rm -f *.o $(EXEC)
```

- **Important :** Utilisez de vraies tabulations (et non des espaces) avant les commandes de compilation.

Astuce : Génération automatique des dépendances

- L'option `-M` de g++ permet de générer la ligne de dépendances d'un fichier source.
- Exemple :

```
g++ -M src/main.cpp
```

- Cette commande affiche les dépendances nécessaires pour `main.cpp` (utile pour automatiser le Makefile).

Introduction à `cmake`

- `cmake` est un outil de configuration de projet multiplateforme.
- Il permet de générer des Makefile (ou des projets pour divers IDE).
- Simplifie la gestion des dépendances et la configuration du projet.

Exemple de CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(go)

# Définition du compilateur
set(CMAKE_CXX_COMPILER g++)

# Définition des fichiers sources
set(SOURCES
    src/main.cpp
    src/fun.cpp
    src/fun.h
)

# Création de l'exécutable
add_executable(${PROJECT_NAME} ${SOURCES})
```

Utilisation de `cmake`

- Créez un répertoire `build` à la racine du projet.
- Dans `build`, lancez les commandes suivantes :

```
cmake ..
make
```

- `cmake ..` configure le projet et génère les Makefile.
- `make` compile le projet.

Conclusion

- `make` est idéal pour des projets simples et des Makefile manuels.
- `cmake` facilite la configuration et la portabilité des projets.
- Veillez toujours à utiliser des tabulations dans vos Makefiles et profitez des options comme `-M` pour automatiser la gestion des dépendances.