Systèmes de construction logicielle
Introduction à cmake

F. Langrognet

laboratoiredemathématiquesdebesançon
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

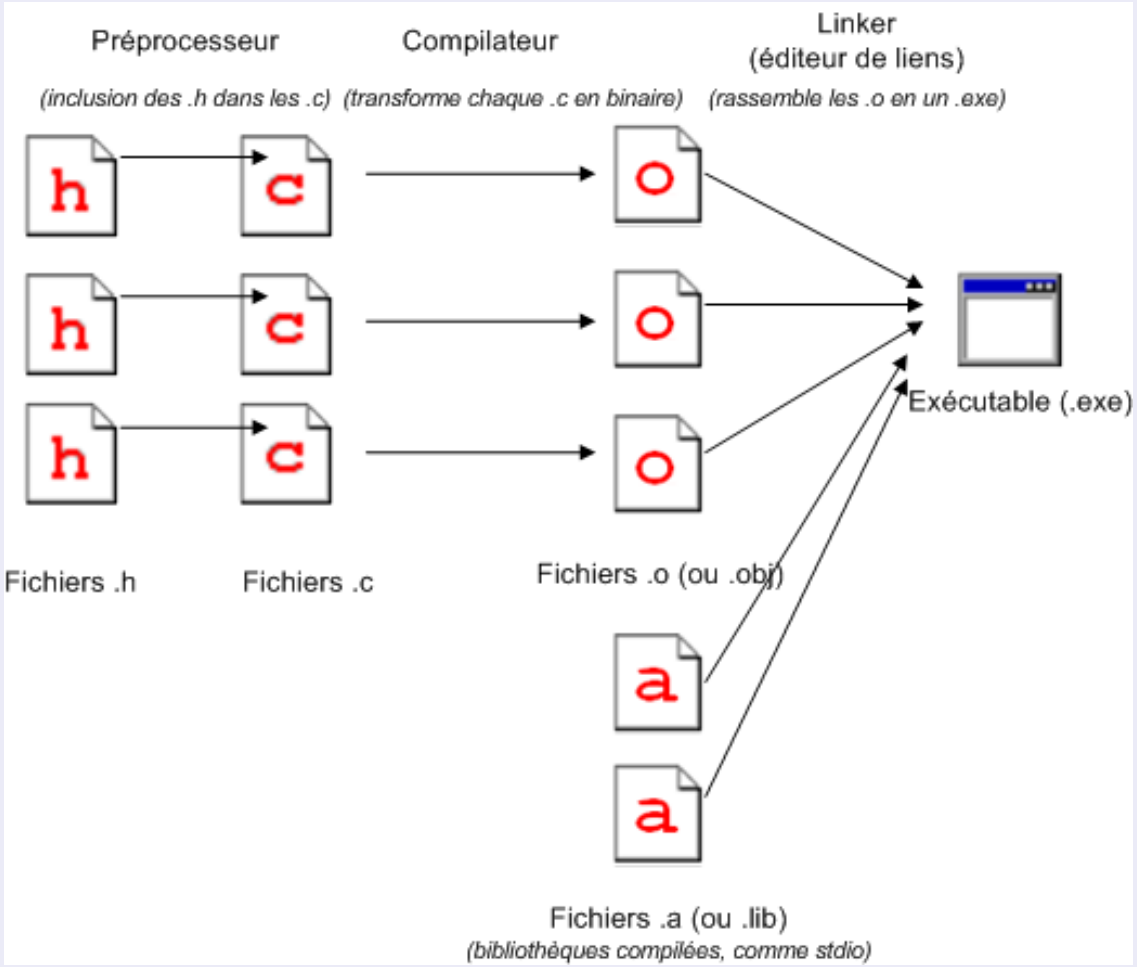
Objectifs des systèmes de construction logicielle

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 systeme 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

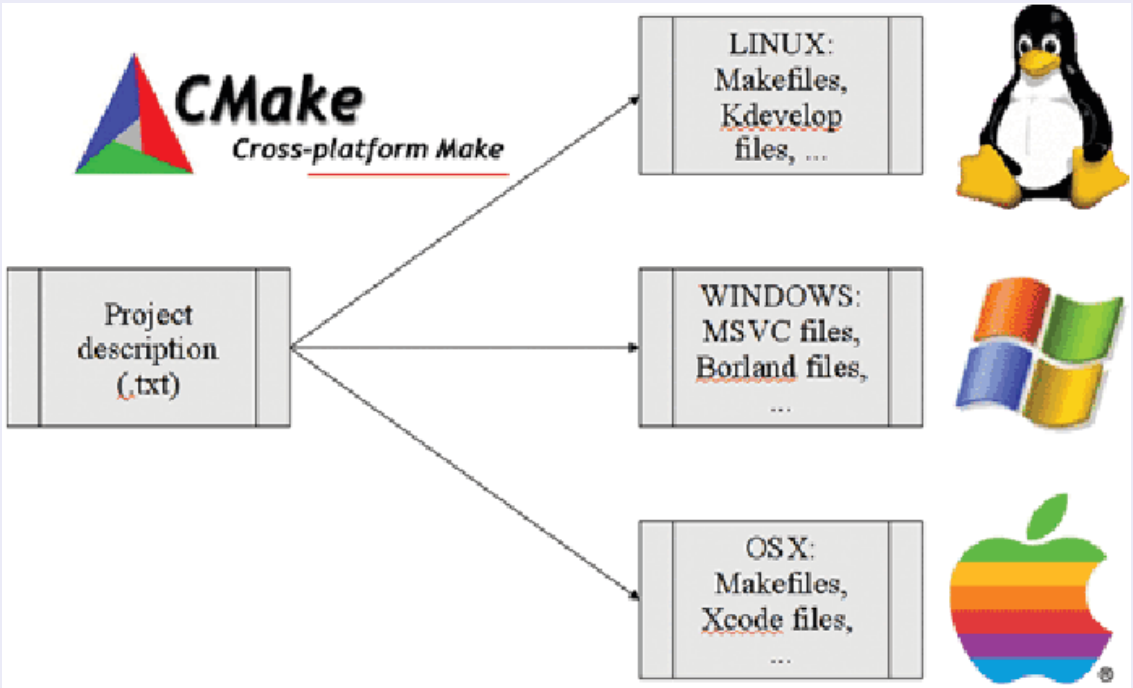
PLAN

PLAN

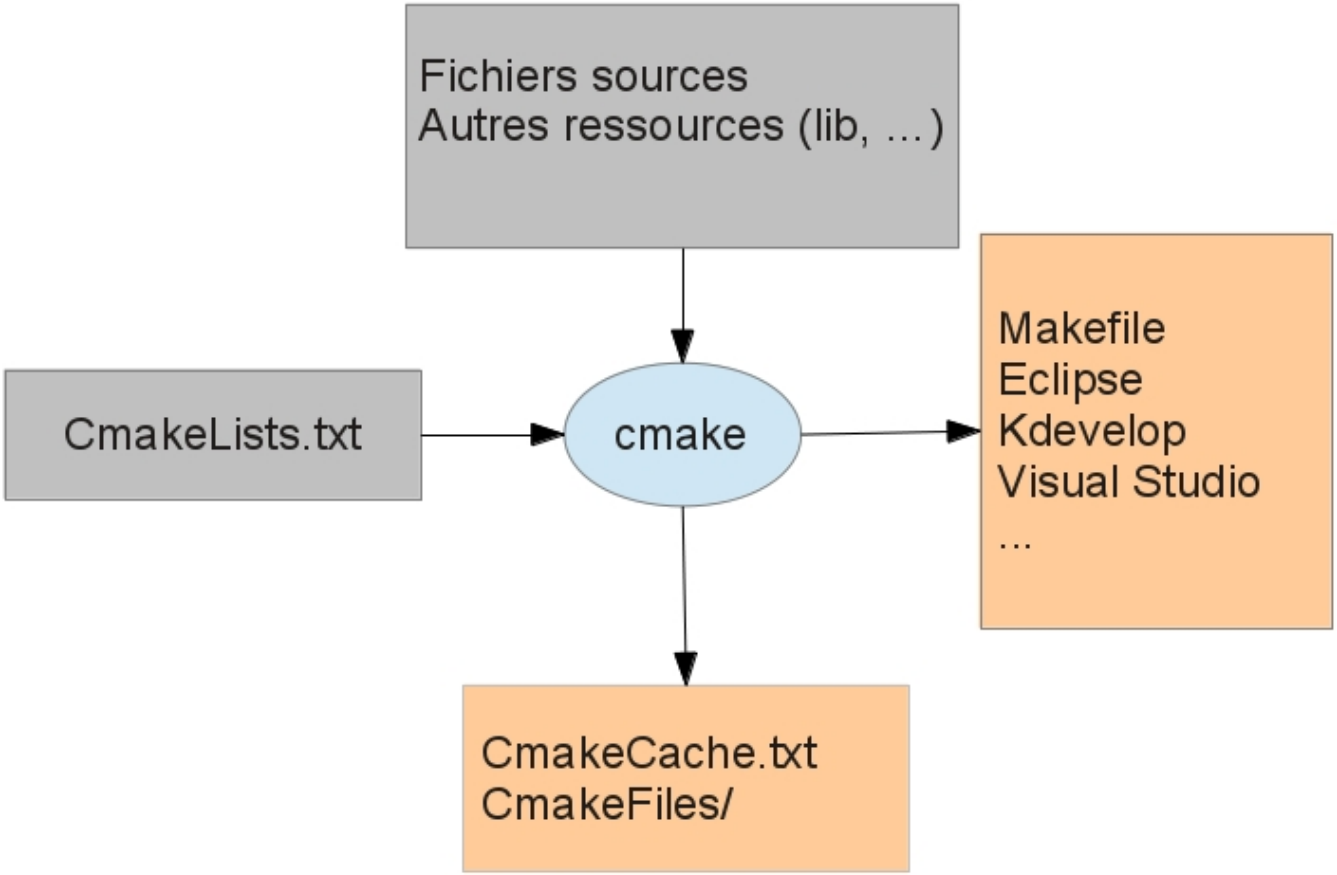
cmake

Logiciel de construction logicielle

- multilangage
- multiplate-forme
- licence libre



cmake - Principe général



PLAN

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

makefile - make

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

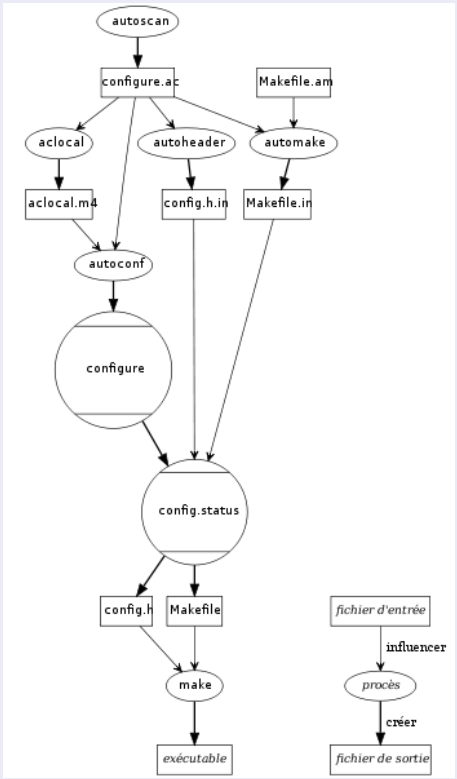
Makefile

```
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)

autotools

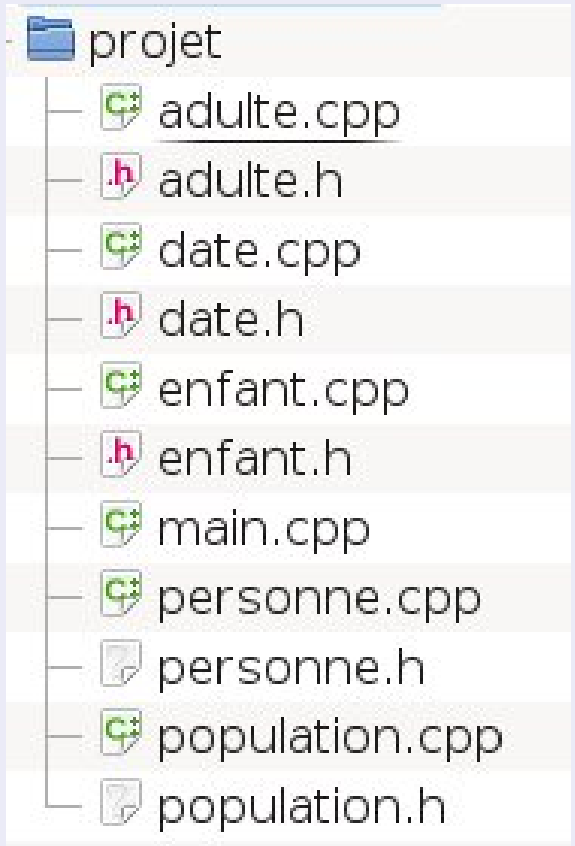


PLAN

PLAN

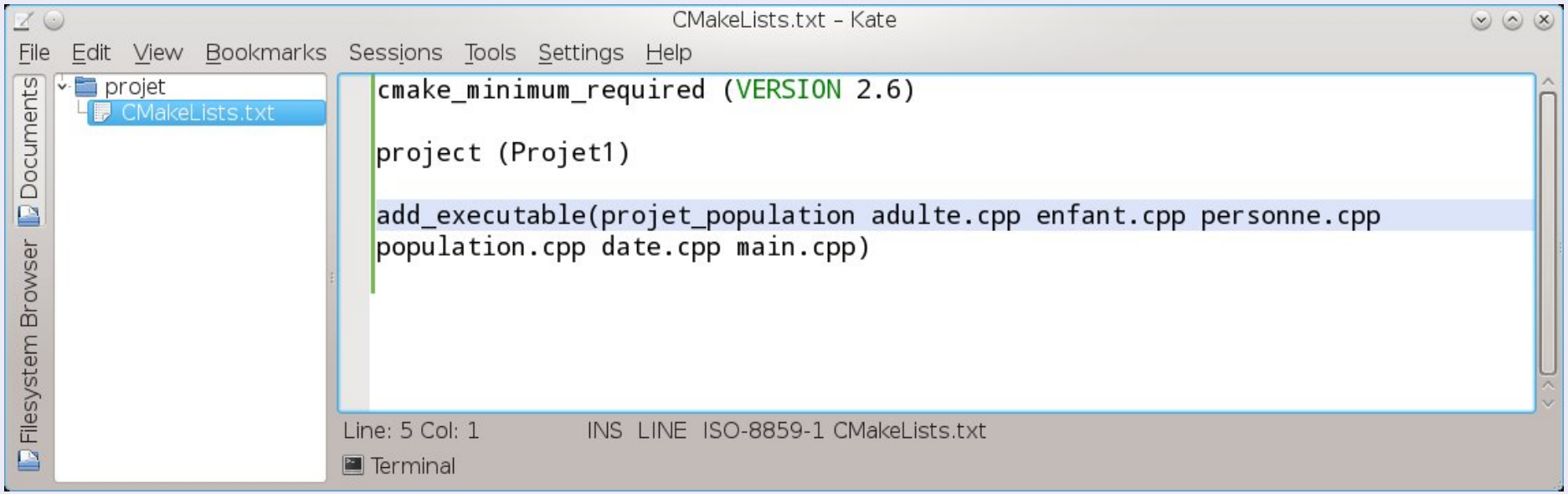
Exemple 1 : Construction d'un simple exécutable

Arborescence du projet



Exemple 1

CMakeLists.txt

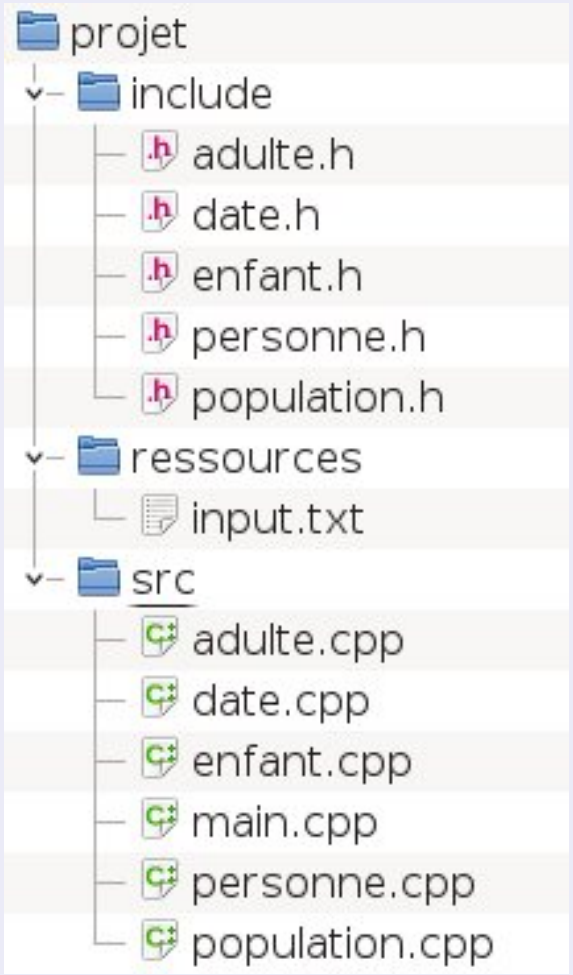


Construction

$ cmake .
$ make
Création de l'exécutable projet_population

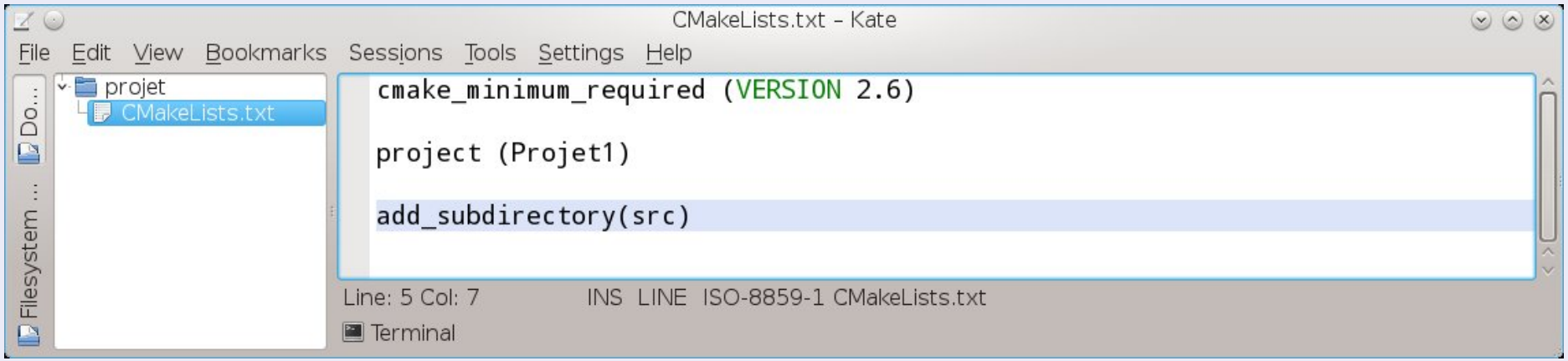
Exemple 2 : Construction d'un simple exécutable avec une arborescence

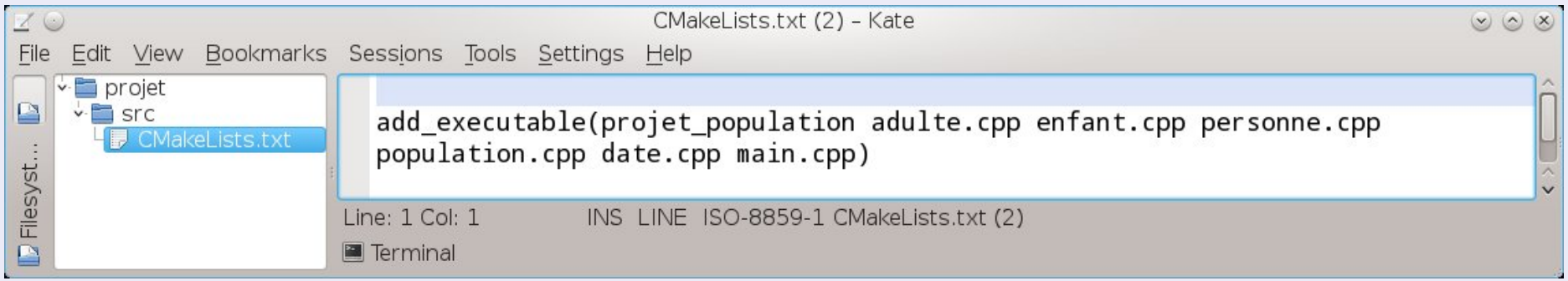
Arborescence du projet



Exemple 2

CMakeLists.txt



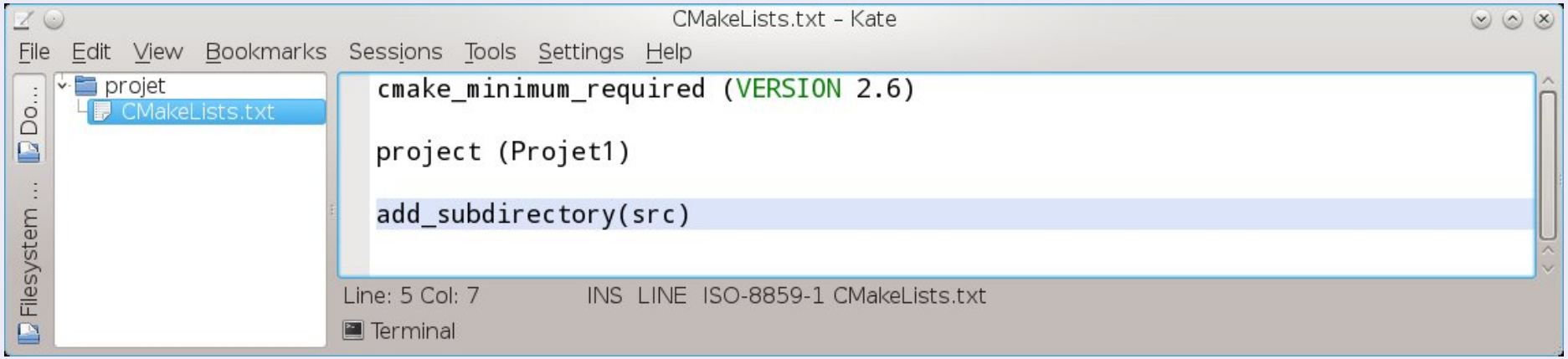


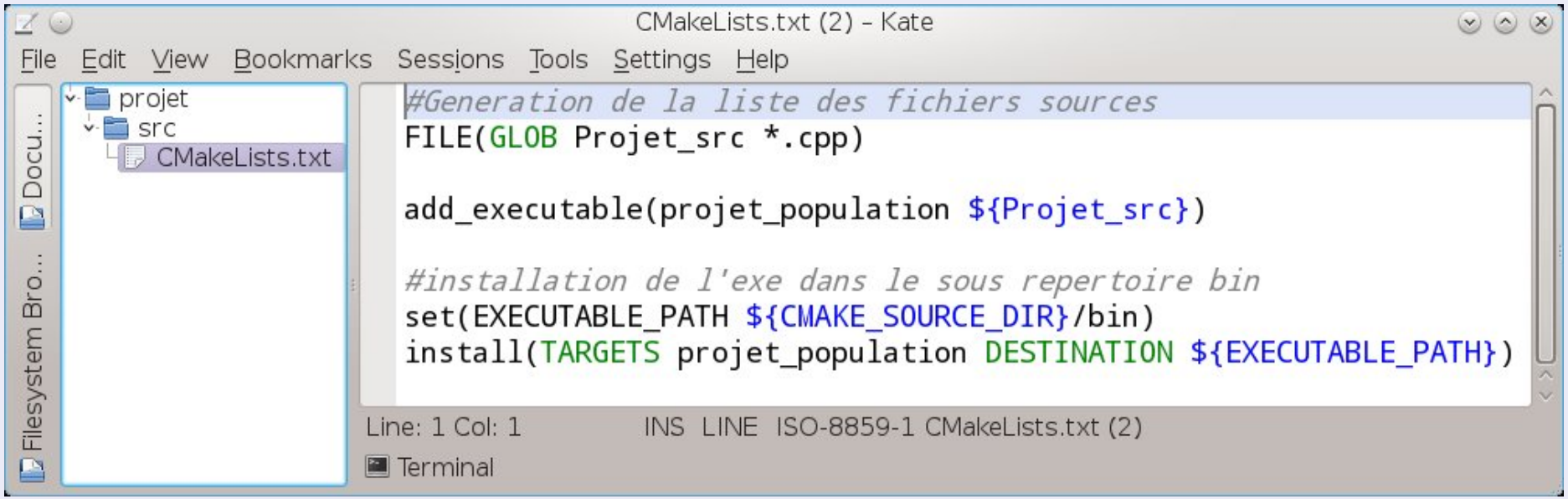
Construction

$ cmake .
$ make

Exemple 3 : automatiser l'installation

CMakeLists.txt





Exemple 3 (suite)

Construction - Installation

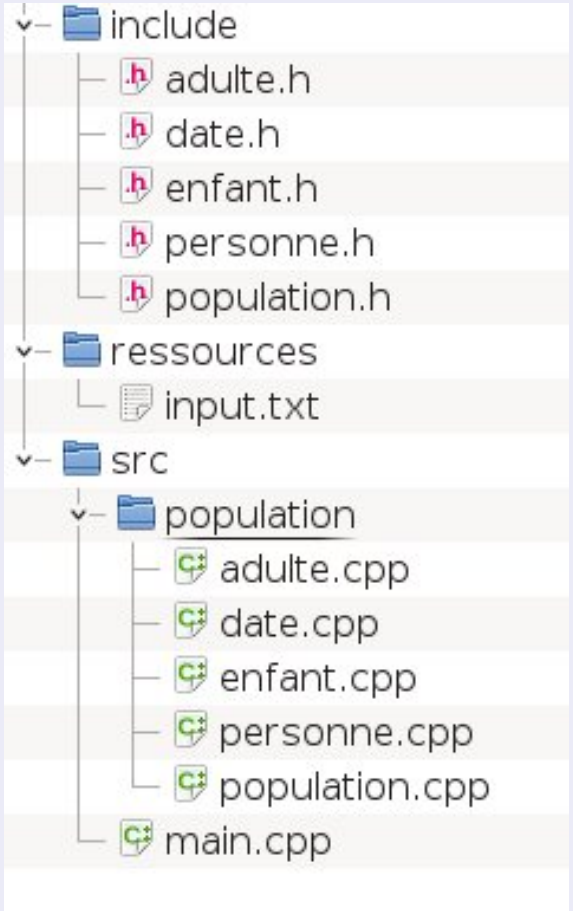
Construction et installation de l'exécutable projet_population dans le sous-répertoire bin

$ cmake .
$ make
$make install

PLAN

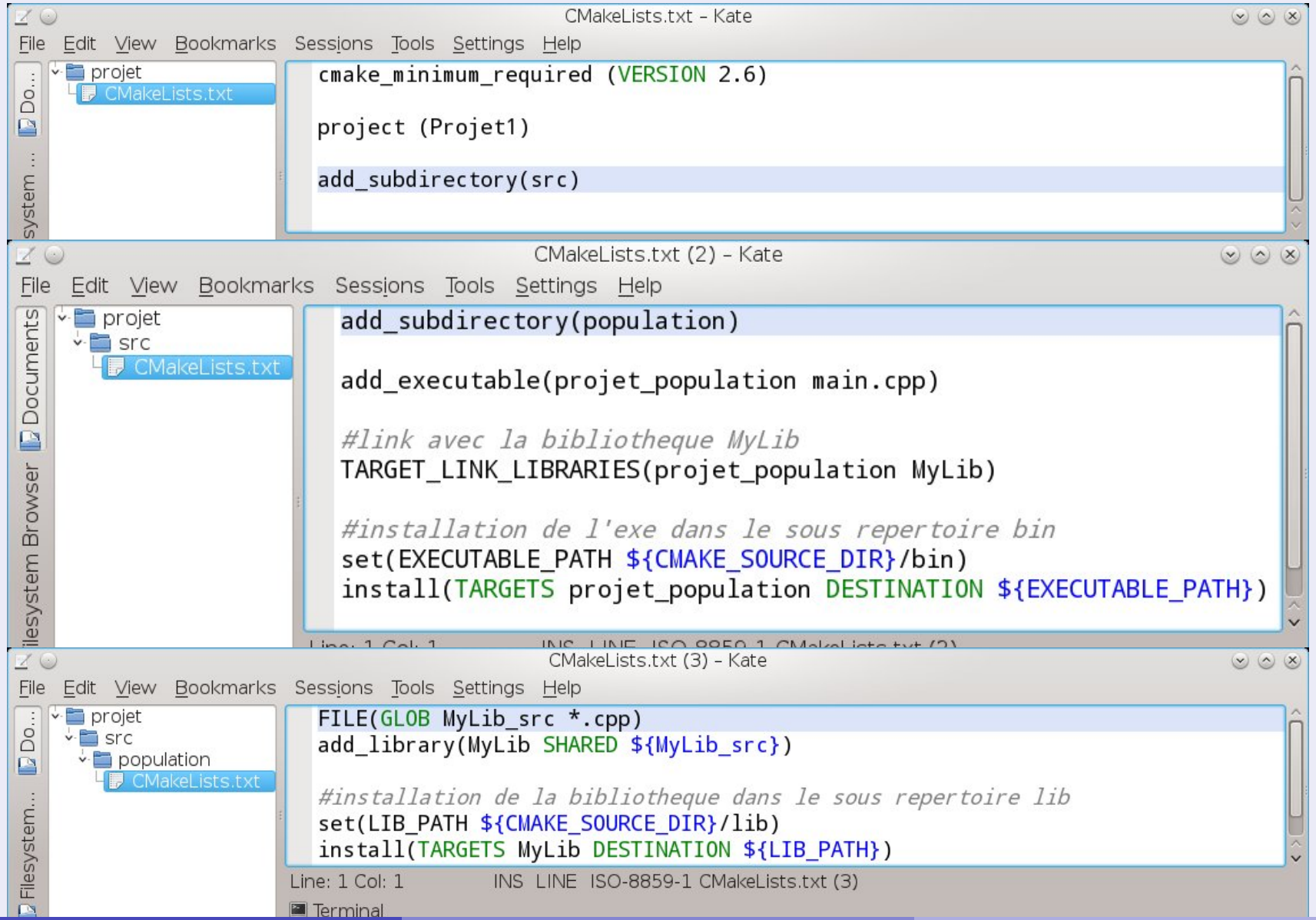
Construction d'une bibliothèque et d'un exécutable

Arborescence du projet



Bibliothèque et exécutable

CMakeLists.txt



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

Construction de packages avec cpack

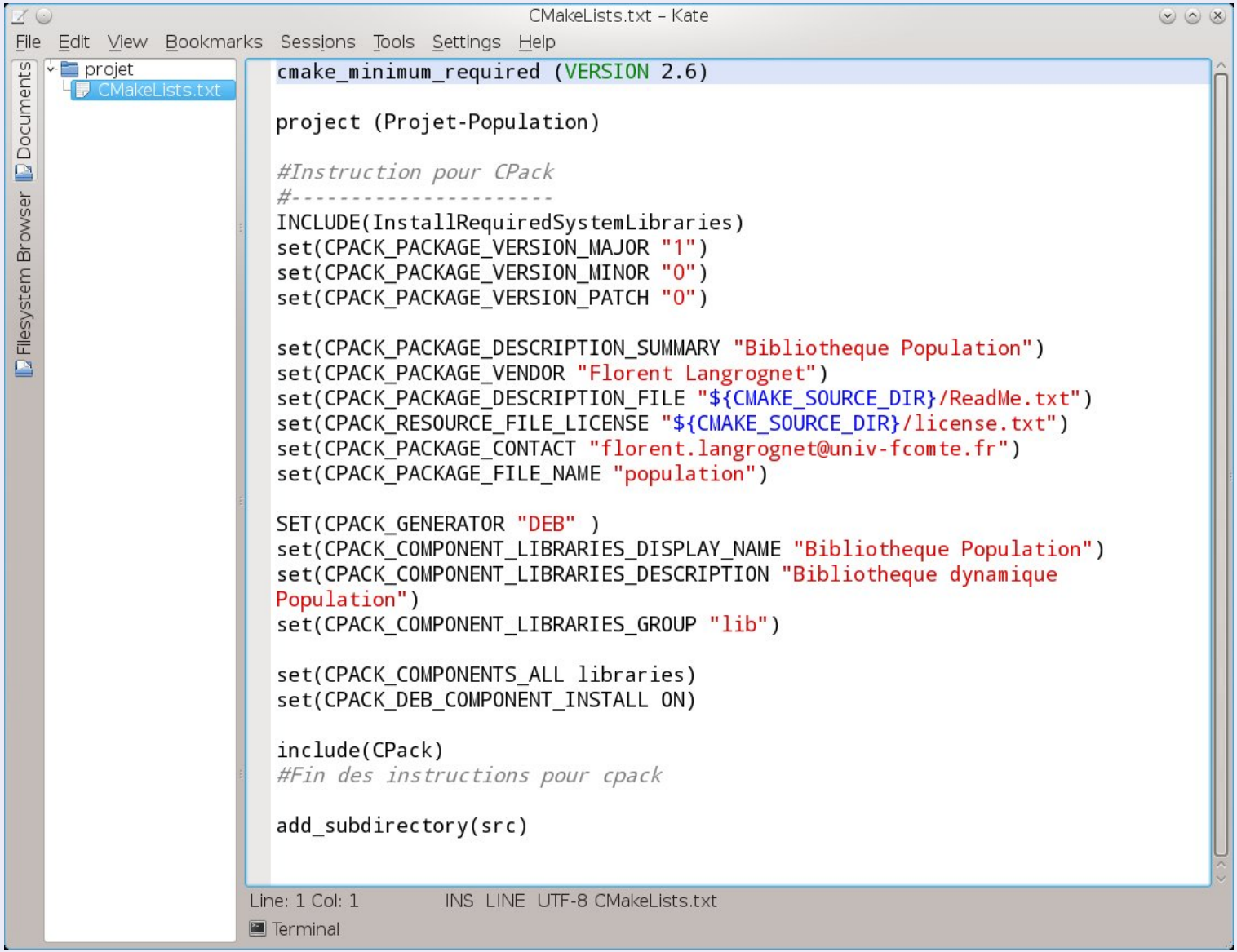
cpack

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



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