
CMake
Cross-platform Make

Contexte

Vous avez développé un super logiciel et vous voulez que le plus de monde possible l'utilise !

Problèmes :

- Il a l'air bien ton logiciel, mais ça compile pas chez moi :(
- Pourtant chez moi ça marche. Tu as quoi comme configuration ?
- J'ai une Ubuntu XX et j'utilise la version 4.2 de la bibliothèque libTruc …
- Dans ce cas il faut modifier la variable BIDULE le Makefile pour que ça compile.
- Oui mais il trouve pas la bibliothèque libTruc :(
- Ah oui il faut aussi…
- OK c'est bon.
- Je vais compléter le fichier INSTALL.txt (+ 100lignes…)
- …

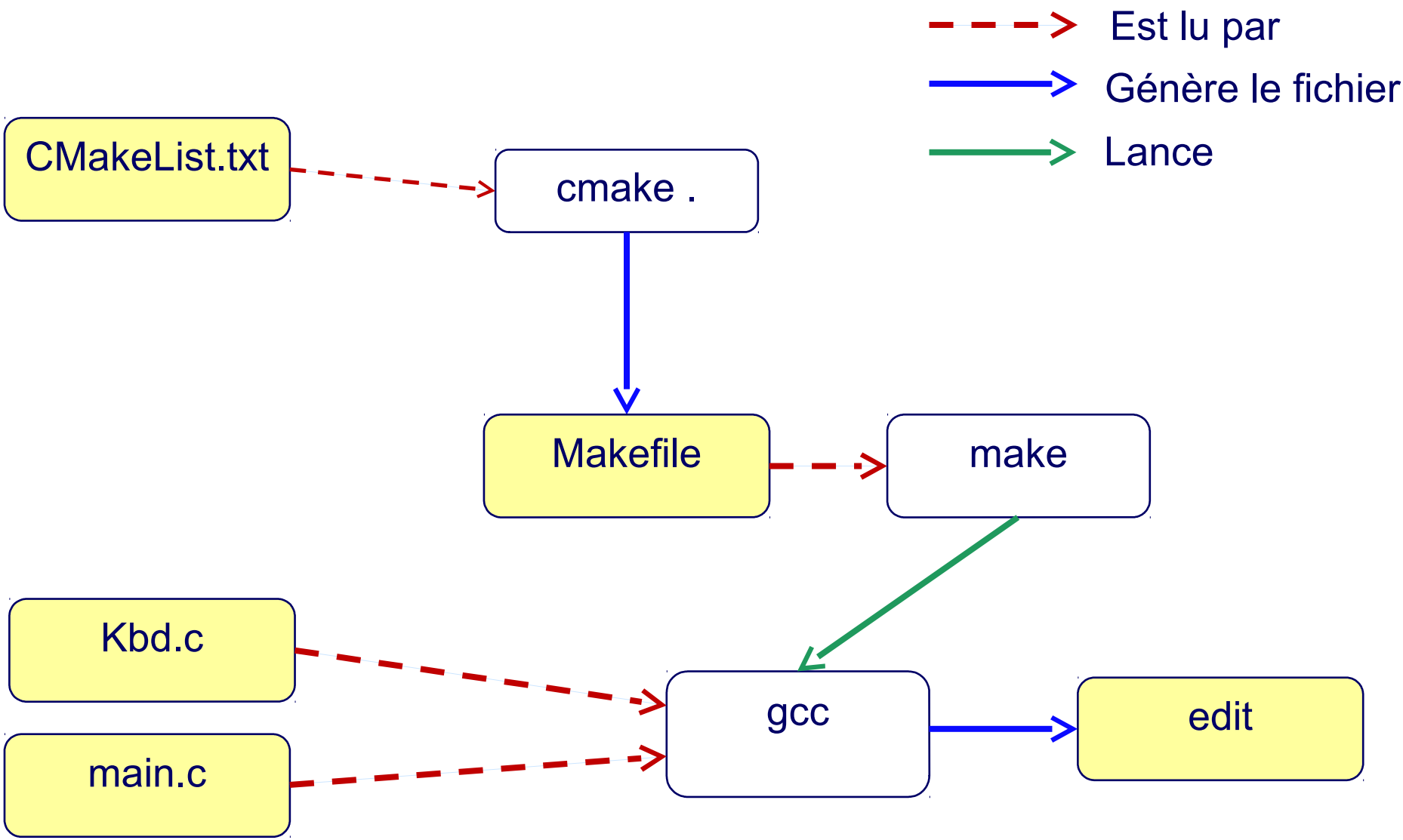
Les limites de make

- Marche bien pour une seule plateforme
- Très dépendant de l'écosystème
  - Système d'exploitation (.so ou .dll ?)
  - Quelles sont les bibliothèques installées
    - Quelle version ?
    - Dans quel répertoire ?
  - Quelles options de compilation
- Bref les Makefile ne sont pas très portables…
- Solutions :
  - Autotools (non-présenté)
  - cmake

INSTALL.txt

- autotools
  - `./configure`
  - `make`
  - `make install`
- cmake
  - `cmake .`
  - `make`
  - `make install`

cmake : fonctionnement



Makefile V3

```
CFLAGS = -g -Wall -std=c99
CPPFLAGS = -I ../include
SOURCES = $(wildcard *.c)
OBJETS = $(SOURCES:.c=.o)

edit : $(OBJETS)
        $(CC) -o $@ $^

main.o : main.c defs.h
kbd.o : kbd.c defs.h command.h
command.o : command.c defs.h command.h
display.o : display.c defs.h
files.o : files.c defs.h command.h

.PHONY : clean #indique à make d'ignorer l'éventuel fichier clean.
clean :
        rm edit $(OBJETS)
```

Mon premier CMakeLists.txt

```
#version minimale de cmake nécessaire
cmake_minimum_required(VERSION 2.6)

#nom du projet
project (Edit C)

#positionne la variable
set(CMAKE_C_FLAGS "-std=c99 -g -Wall")

#définit le nom du programme ainsi que
#ses sources
add_executable(edit main.c kbd.c command.c
display.c files.c)

#les dépendances sont calculées automatiquement
```

CMakeLists.txt : syntaxe

- Simple : séquence d'appels de commande :
- Appel de commande :
  `<nom_commande>(<param1> <param2> <param3>)`
- N'est pas sensible à la casse(majuscule/minuscule).

CMakeLists.txt : Exemples de commandes

```
#affectation
set(<nom variable> <expression>)

# définit un une cible exécutable ainsi que ses
dépendances.
add_executable(<nom_exe> <source1> <source2>…)

#définit les bibliothèques utilisées par <mon_exe>.
target_link_libraries(<mon_exe> <lib1> <lib2>…)

if(<expression>)
  <commande1>
  <commande2>
[else()
  <commande3>]
endif()
```

CMakeLists.txt : Exemples de commandes

```
#comme add_executable mais pour générer une lib
add_library(<nom_lib> <source1> <source2>…)

#ajoute une règle (au Makefile généré)
add_custom_target(<nom> <command1> [DEPENDS <depend1>
<depend2> …])

#recopie des cibles lors de l'installation
install(TARGETS <target1> <target2> DESTINATION <dest>)

add_subdirectory(<dir>)


while(<expression>)
endwhile()
```

Personnaliser…

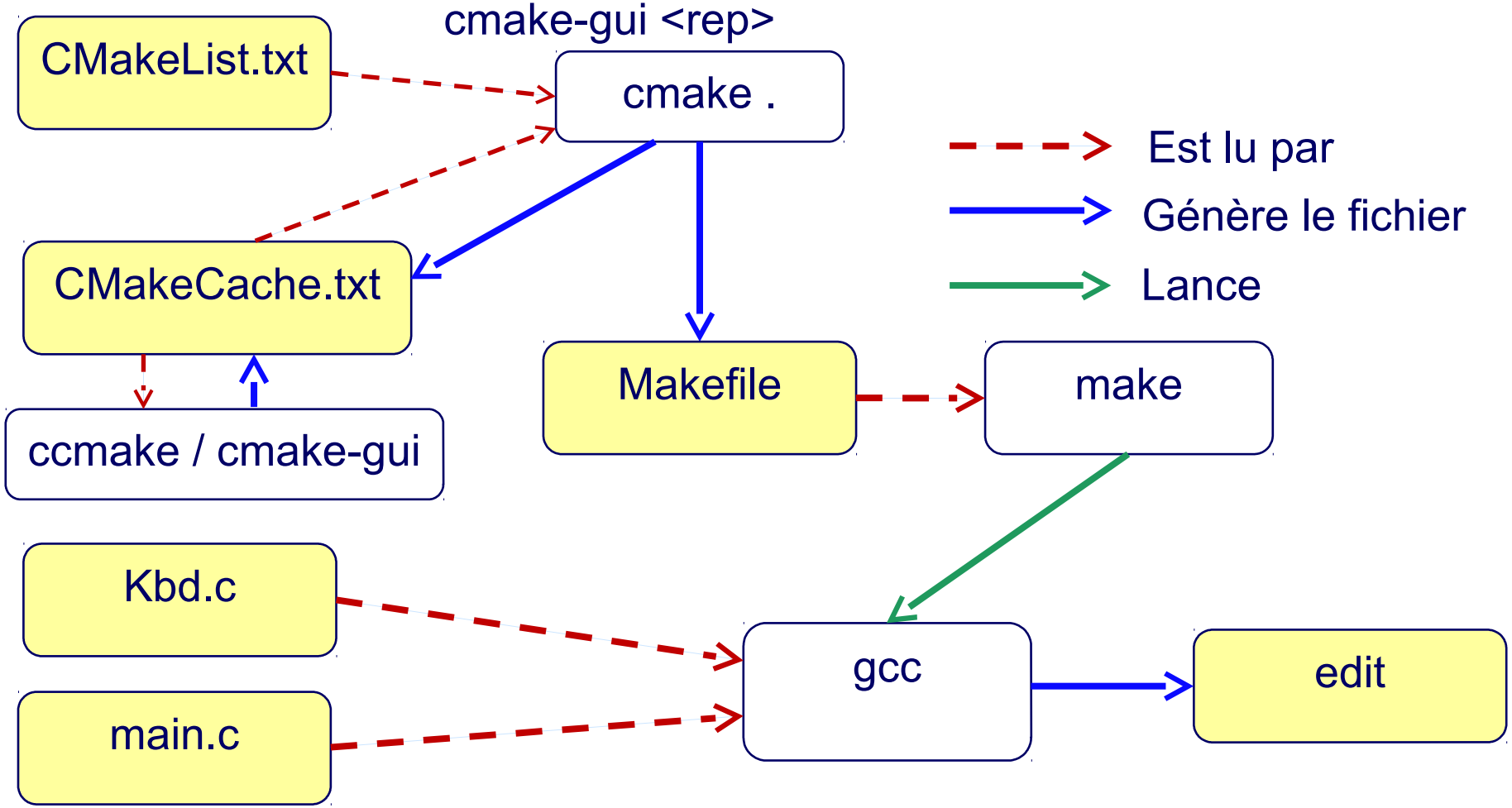
- CMakeLists.txt est indépendant de la plateforme
- Pour personnaliser le processus :
  - cmake <rep> -D<VARIABLE>=<VALEUR> …
  - ccmake <rep>
  - cmake-gui <rep>

- (DEMO)

Personnaliser…

CMakeLists.txt est indépendant de la plateforme
Pour personnaliser le processus :

cmake <rep> –D<VARIABLE>=<VALEUR> …
ccmake <rep>
cmake-gui <rep>



In-source vs out-of-source

- In-source : les objets/exécutables dans les sources
  - Plus facile à débugger
  - Un peu plus le « bazar »… mais ce n'est pas un souci avec svn
- Out-of-source : les objets/exécutables dans un autre répertoire
  - plus pénible à débugger (configuration de gdb…)
  - Le répertoire src n'est pas modifié.
  - Permet d'avoir plusieurs configurations différentes (debug/release)

In-source vs out-of-source

- In-source :
  - `cmake .`
- Out-of-source :
  - `mkdir build`
  - `cd build`
  - `cmake ..`

Tests avec cmake : CMakeLists.txt

```
enable_testing()


#définit le premier test : lancer test-vext-1
add_test(tv1 test-vext-1)

#définit le succès du test : le test réussi si l'exécution de
#test-vext-1 affiche exactement "v1 : 10 40 30"
set_tests_properties (tv1
  PROPERTIES PASS_REGULAR_EXPRESSION "v1 : 10 40 30")

(…)

add_test(tv4 test-vext-4)
#On définit un "timeout" pour stopper le programme au bout de 3
secondes"
set_tests_properties (tv4 PROPERTIES TIMEOUT 3)

(…)

# Crée une cible "check" pour make. Cette cible lance la commande
"ctest"
add_custom_target(check COMMAND ${CMAKE_CTEST_COMMAND})
```

Tests avec cmake : lancement des tests

```
$ ctest .  (ou make check si vous avez créé la bonne cible)
Test project XXX
    Start 1: tv1
1/6 Test #1: tv1 ..............................   Passed    0.07 sec
    Start 2: tv2
2/6 Test #2: tv2 ..............................   Passed    0.04 sec
    Start 3: tv3
3/6 Test #3: tv3 ..............................   Passed    0.04 sec
    Start 4: tv4
4/6 Test #4: tv4 ..............................***Timeout   3.03 sec
    Start 5: tv5
5/6 Test #5: tv5 ..............................***Exception: Other
0.16 sec
    Start 6: tv6
6/6 Test #6: tv6 ..............................***Timeout   3.03 sec

50% tests passed, 3 tests failed out of 6
Total Test time (real) =   6.39 sec
The following tests FAILED:
          4 - tv4 (Timeout)
          5 - tv5 (OTHER_FAULT)
          6 - tv6 (Timeout)
Errors while running CTest
```

Références

- Documentation sur cmake: http://www.cmake.org/cmake/help/documentation.html
- Documentation sur les autotools : http://www.gnu.org/software/automake/manual/