CMAKE Tutos

Jean-Charles Lambert

28 septembre 2016

Outline

- Description
- Simple example
- Advanced example
- Exercises

Online tutorial : http://goo.gl/8jh3JS

Description : what is it ?

- A cross platform Makefile generator
  - To compile source code
  - To create libraries (shared or static)
  - To build executables
  - To create bundle and installation process
- Generate building process for compiled languages
  - C/C++/Fortran
- Based on a single script (CMakeLists.txt) for specifying rules
- Can target differents platforms (Linux, MacOS, Windows)
- Popularity (Netflix, MysQL, HDF group, KDE)
- Developped at LAM : Amazed, cpf-redshift, Glnemo2, Unsio :))

Description : concept

My awesome project

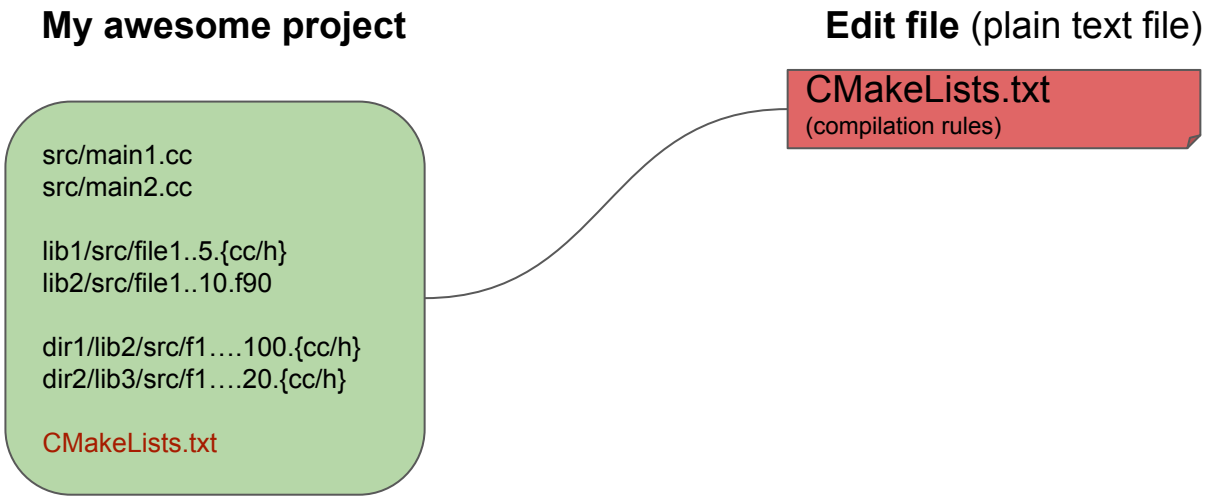
src/main1.cc
src/main2.cc

lib1/src/file1..5.{cc/h}
lib2/src/file1..10.f90

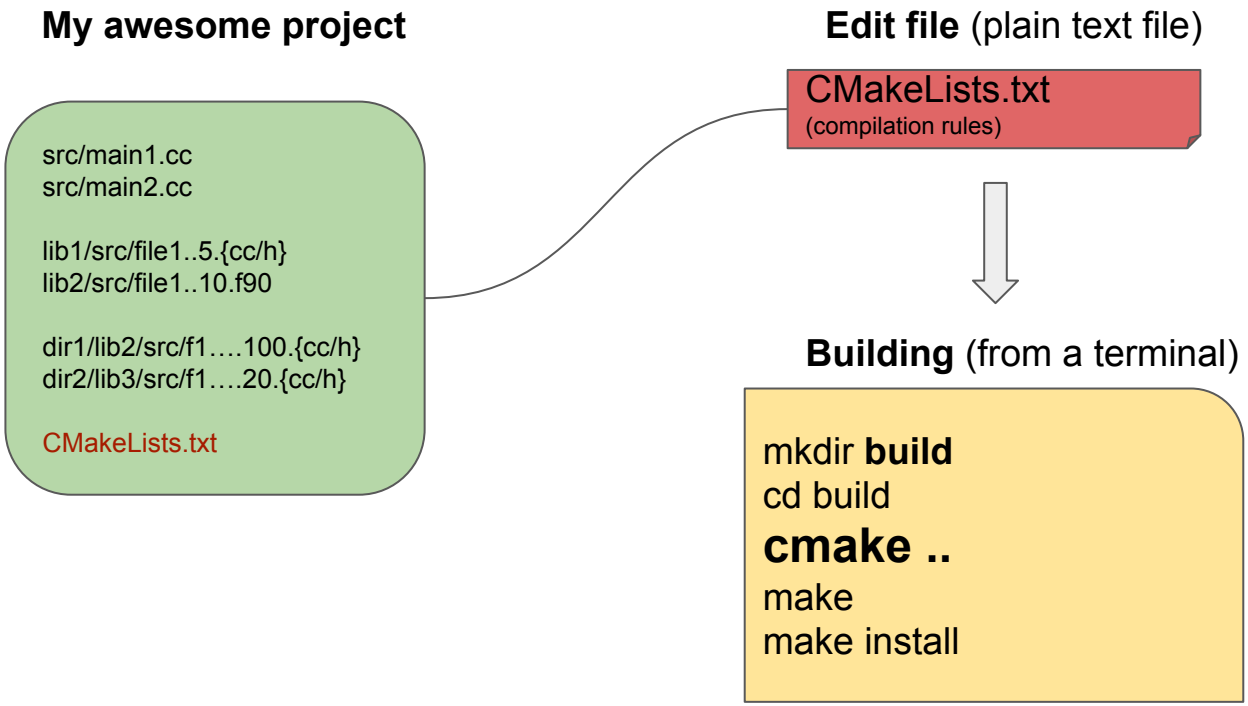
dir1/lib2/src/f1….100.{cc/h}
dir2/lib3/src/f1….20.{cc/h}

CMakeLists.txt

Description : concept



Description : concept



Description : concept

Building (from a terminal)

```
mkdir build
cd build
cmake ..
make
make install
```

Cmake command generates :

- Native build systems
- Intermediate files
- Object files
- Build ouput
- Libraries
- Binaries

Build directory can be removed

Description : installation

- Linux : use your paquet manager
  - urpmi cmake
  - apt-get install cmake
  - zypper install cmake
  - yum install cmake
  - dnf install cmake
  - ….
- MacOs : dmg, port, brew whatever…
- Or download and compile it ! https://cmake.org/

Simple example

hello.cc

```
#include <iostream>

int main(void) {
    std::cerr << "Hello World\n”;
    return(0);
}
```

CMakeLists.txt

```
#Specify the version being used as well as the language
cmake_minimum_required(VERSION 2.6)

#Name your project here
project(hello)

#Add “-g -O2” options to the gcc compiler
add_definitions(-g -O2)

# specify executable name and dependencies
add_executable(hello hello.cc)
```

Simple example

mkdir **build**
cd build
cmake ..
make

cmake ..
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/jcl/works/GIT/cmake-tutos/ex01/build

make
Scanning dependencies of target hello
[100%] Building CXX object CMakeFiles/hello.dir/hello.cc.o
Linking CXX executable hello
[100%] Built target hello

*We create **build** directory to*
Isolate compilations files from sources files

Advanced example : goal

- Create a project with several files
- How to compile a library
- How to specify an installation directory

Advanced example : description

Hello project

src/hello.cc

lib01/CMakeLists.txt

lib01/src/chello.cc
lib01/src/insane.c
lib01/include/chello.h

cmake/SetupInstallLib.cmake

CMakeLists.txt

src/ :
directory to store mains programs

lib01/src/:
directory to store sources lib files

lib01/include/:
directory to store includes lib files

cmake/:
directory to store cmake modules

Advanced example : source files

src/hello.cc

```
#include <iostream>
#include <chello.h>

int main(void) {
  CHello hello("hello world");
  hello.display();
  hello.crazy();
}
```

lib01/src/chello.h

```
#include <iostream>
#include <string>
extern "C" {
  int insane(int,int,char*);
}
class CHello {
 public:
 CHello(const std::string myhello):mystring(myhello) {
  }
  void display();
  void crazy();
 private:
  std::string mystring;
};
```

lib01/include/chello.cc

```
#ifndef CHELLO_H
#define CHELLO_H
#include <chello.h>
void CHello::display()
{
  std::cerr << mystring << "\n";
}
void CHello::crazy()
{
  insane(1,1,(char *) "");
}
#endif //CHELLO_H
```

lib01/src/insane.c

```
#include <stdio.h>
int insane(int t,int _, char * a)
{
return!0<t?t<3?insane(-79,-13,a+insane(-87,1-_,insane(-86,0,a+1)+a)):
1,t<_?insane(t+1,_,a):3,insane(-94,-27+t,a)&&t==2?_<13?
insane(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?insane(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l+,/n{n+,/+#n+
,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){}#}w')){}{nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;:{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' '}#\
 }'+}##(!!/")

:t<-50?_==*a?putchar(31[a]):insane(-65,_,a+1):insane((*a=='/')+t,_,a+1)
 :0<t?insane(2,2,"%s"):*a=='/'||insane(0,insane(-61,*a,
 "!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m
.vpbks,fxntdCeghiry"),a+1);
}
```

Advanced example : running cmake

mkdir **build**
cd build
cmake ..
make

cmake ..
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- New executable ==> hello
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/jcl/works/GIT/cmake-tutos/ex02/build

PROCESS:
New executable hello detected from
top level src directory

Advanced example : running make

```
mkdir build
cd build
cmake ..
make
```

```
make
Scanning dependencies of target MYlib
[ 33%] Building CXX object
lib01/CMakeFiles/MYlib.dir/src/chello.cc.o
[ 66%] Building C object
lib01/CMakeFiles/MYlib.dir/src/insane.c.o
Linking CXX shared library ../lib/libMYlib.so
[ 66%] Built target MYlib
Scanning dependencies of target hello
[100%] Building CXX object CMakeFiles/hello.dir/src/hello.cc.o
Linking CXX executable hello
[100%] Built target hello
```

PROCESS:
Compilation of shared library libMYlib.so
Compilation of executable hello

Advanced example : anatomy

Hello project

src/hello.cc

lib01/CMakeLists.txt

lib01/src/chello.cc
lib01/src/insane.c
lib01/include/chello.h

cmake/SetupInstallLib.cmake

CMakeLists.txt

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.6)
#Name your project here
project(hello)
# custom cmake modules path
SET(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH} ${PROJECT_SOURCE_DIR}/cmake)
# Special rules for lib installation
include(SetupInstallLib)
#Add “-g -O2” options to the gcc compiler
add_definitions(-g -O2)
# add library to build (located in "lib01" directory)
add_subdirectory(lib01)
# add lib directory as include search dir
include_directories(lib01/include)
# Find all c++ main exes sources files
FILE(GLOB main_src src/*.cc)
# build cpp executables according to the source
FOREACH(main_exe ${main_src})
  get_filename_component(exe ${main_exe} NAME_WE)  # get full name without directory
  MESSAGE( STATUS "New executable ==> " ${exe})    # print exe name
  add_executable (${exe} ${main_exe})              # specify exe file to cpmpile
  target_link_libraries (${exe}  MYlib )           # specify library dependencies
  INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)  # binaries install directory
ENDFOREACH()

# Install destinattion directory
if (CMAKE_INSTALL_PREFIX_INITIALIZED_TO_DEFAULT)
  set (CMAKE_INSTALL_PREFIX $ENV{HOME}/local  CACHE PATH "" FORCE)
endif()
```

Advanced example : anatomy (CMakeLists.txt)

```
cmake_minimum_required(VERSION 2.6)

project(hello)

SET(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH} ${PROJECT_SOURCE_DIR}/cmake)

include(SetupInstallLib)

add_definitions(-g -O2)
```

- **CMAKE_MODULE_PATH** : cmake module path = files with “**.cmake”** extension
- **${PROJECT_SOURCE_DIR}/lib** : top level source directory for the current project.
- **include(SetupInstallLib)** : load cmake module => SetupInstallLib.cmake

Advanced example : anatomy (CMakeLists.txt)

```
add_subdirectory(lib01)

include_directories(lib01/include)  # similar to compilation option -Ilib01/include
```

- **add_subdirectory(lib01)** : add subdirectory **“lib01”** to the build and process its **CMakeList.txt** file
  You must have a lib01/CMakeLists.txt file

- **include_directories(lib01/include)** : add lib01/include path to compilation include search path
  Equivalent to : g++ **-Ilib01/include** ..

  You can have multiple include_directories() command, path wil be appended

Advanced example : anatomy (CMakeLists.txt)

```
FILE(GLOB main_src src/*.cc)

FOREACH(main_exe ${main_src})
  get_filename_component(exe ${main_exe} NAME_WE)
  MESSAGE( STATUS "New executable ==> " ${exe})
  add_executable (${exe} ${main_exe})
  target_link_libraries (${exe}  MYlib )
  INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
ENDFOREACH()
```

- **FILE(GLOB main_src src/*.cc)** : create a variable **main_src** with all c++ sources files from top level src directory.

Advanced example : anatomy (CMakeLists.txt)

```
FILE(GLOB main_src src/*.cc)

FOREACH(main_exe ${main_src})
  get_filename_component(exe ${main_exe} NAME_WE)
  MESSAGE( STATUS "New executable ==> " ${exe})
  add_executable (${exe} ${main_exe})
  target_link_libraries (${exe}  MYlib )
  INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
ENDFOREACH()
```

- **FILE(GLOB main_src src/*.cc)** : create a variable **main_src** with all c++ sources files from top level src directory.
- **FOREACH(main_exe ${main_src})** : loop over **main_src** variable to fill up **main_exe** variable

Advanced example : anatomy (CMakeLists.txt)

```
FILE(GLOB main_src src/*.cc)

FOREACH(main_exe ${main_src})
  get_filename_component(exe ${main_exe} NAME_WE)
  MESSAGE( STATUS "New executable ==> " ${exe})
  add_executable (${exe} ${main_exe})
  target_link_libraries (${exe}  MYlib )
  INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
ENDFOREACH()
```

- **FILE(GLOB main_src src/*.cc)** : create a variable **main_src** with all c++ sources files from top level src directory.
- **FOREACH(main_exe ${main_src})** : loop over **main_src** variable to fill up **main_exe** variable
- **get_filename_component(exe ${main_exe} NAME_W)** : set **exe** variable from **main_exe** without directory name nor extension : **main_exe=src/hello.cc → exe=hello**

Advanced example : anatomy (CMakeLists.txt)

```
FILE(GLOB main_src src/*.cc)

FOREACH(main_exe ${main_src})
  get_filename_component(exe ${main_exe} NAME_WE)
  MESSAGE( STATUS "New executable ==> " ${exe})
  add_executable (${exe} ${main_exe})
  target_link_libraries (${exe}  MYlib )
  INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
ENDFOREACH()
```

- **FILE(GLOB main_src src/*.cc)** : create a variable **main_src** with all c++ sources files from top level src directory.
- **FOREACH(main_exe ${main_src})** : loop over **main_src** variable to fill up **main_exe** variable
- **get_filename_component(exe ${main_exe} NAME_W)** : set **exe** variable from **main_exe** without diretory name nor extension : **main_exe=src/hello.cc → exe=hello**
- **add_executable (${exe} ${main_exe})** : add new executable exe which depends from main_exe

Advanced example : anatomy (CMakeLists.txt)

```
FILE(GLOB main_src src/*.cc)

FOREACH(main_exe ${main_src})
  get_filename_component(exe ${main_exe} NAME_WE)
  MESSAGE( STATUS "New executable ==> " ${exe})
  add_executable (${exe} ${main_exe})
  target_link_libraries (${exe}  MYlib )
  INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
ENDFOREACH()
```

- **FILE(GLOB main_src src/*.cc)** : create a variable **main_src** with all c++ sources files from top level src directory.
- **FOREACH(main_exe ${main_src})** : loop over **main_src** variable to fill up **main_exe** variable
- **get_filename_component(exe ${main_exe} NAME_W)** : set **exe** variable from **main_exe** without diretory name nor extension : **main_exe=src/hello.cc → exe=hello**
- **add_executable (${exe} ${main_exe})** : add new executable exe which depends from main_exe
- **target_link_libraries (${exe} MYlib)** : executable must be linked against MYlib library (libMYlib.so)

Advanced example : anatomy

Hello project

src/hello.cc

lib01/CMakeLists.txt

lib01/src/chello.cc
lib01/src/insane.c
lib01/include/chello.h

cmake/SetupInstallLib.cmake

CMakeLists.txt

lib01/CMakeLists.txt

```
# Find all library sources files
FILE(GLOB SRCLIB src/*.cc src/*.c) # src sources files relatives too lib01
# add  current directory as include search dir
include_directories(include)
# create library "MYlib"
add_library (MYlib SHARED ${SRCLIB})
# Destination path for the lib
SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)

INSTALL(FILES include/chello.h DESTINATION include)
INSTALL(FILES ${PROJECT_BINARY_DIR}/lib/libMYlib.so DESTINATION lib)
```

Advanced example : anatomy (lib01/CMakeLists.txt)

****REMEMBER**** : **add_subdirectory(lib01)** from top level CMakeLists.txt

```
FILE(GLOB SRCLIB src/*.cc src/*.c)

include_directories(include)
```

- **FILE(GLOB SRCLIB src/*.cc src/*.c)** : create a variable **SRC_LIB** with all c++/c sources files from src subdirectory, actually lib01/src (recursive search)
- **include_directories(include)** : add **include**, actually lib01/include, path to compilation include search path

Advanced example : anatomy (lib01/CMakeLists.txt)

```
add_library (MYlib SHARED ${SRCLIB})

SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
```

- **add_library (MYlib SHARED ${SRCLIB})** : Add a SHARED lib libMYlib.so to the project using the specified source files from SRCLIB variable.
  Note: no **lib** prefix nor **.so** suffix
- **STATIC** instead of **SHARED => static library**
- **SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)** : library directory destination
  PROJECT_BINARY_DIR : directory from where you enter cmake command, here **build**

Advanced example : running make install

```
mkdir build
cd build
cmake ..
make
make install
```

make install

```
[ 66%] Built target MYlib
[100%] Built target hello
Install the project...
-- Install configuration: ""
-- Installing: /home/jcl/local/bin/hello
-- Set runtime path of "/home/jcl/local/bin/hello" to "/home/jcl/local/lib"
-- Installing: /home/jcl/local/include/chello.h
-- Installing: /home/jcl/local/lib/libMYlib.so
```

PROCESS:
Install binaries, libraries and headers in default or specific location

Advanced example : anatomy (installation)

Hello project

src/hello.cc

lib01/CMakeLists.txt

lib01/src/chello.cc
lib01/src/insane.c
lib01/include/chello.h

cmake/SetupInstallLib.cmake

CMakeLists.txt

cmake install statements

```
INSTALL(FILES include/chello.h DESTINATION include)
INSTALL(FILES ${PROJECT_BINARY_DIR}/lib/libMYlib.so DESTINATION lib)
```

```
# Install destinattion directory
if (CMAKE_INSTALL_PREFIX_INITIALIZED_TO_DEFAULT)
  set (CMAKE_INSTALL_PREFIX $ENV{HOME}/local  CACHE PATH "" FORCE)
endif()
SET(CMAKE_SKIP_BUILD_RPATH  FALSE)
# when building, don't use the install RPATH already
# (but later on when installing)
SET(CMAKE_BUILD_WITH_INSTALL_RPATH FALSE)
SET(CMAKE_INSTALL_RPATH "${CMAKE_INSTALL_PREFIX}/lib")
# add the automatically determined parts of the RPATH
# which point to directories outside the build tree to the install RPATH
SET(CMAKE_INSTALL_RPATH_USE_LINK_PATH TRUE)
# check RPATH to be used when installing is not a system directory
LIST(FIND CMAKE_PLATFORM_IMPLICIT_LINK_DIRECTORIES "${CMAKE_INSTALL_PREFIX}/lib"
isSystemDir)
IF("${isSystemDir}" STREQUAL "-1")
   SET(CMAKE_INSTALL_RPATH "${CMAKE_INSTALL_PREFIX}/lib")
ENDIF("${isSystemDir}" STREQUAL "-1")
```

```
INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
```

Advanced example : anatomy (cmake/SetupInstallLib)

****REMEMBER**** : **include(SetupInstallLib)** from top level CMakeLists.txt

```
if (CMAKE_INSTALL_PREFIX_INITIALIZED_TO_DEFAULT)
  set (CMAKE_INSTALL_PREFIX $ENV{HOME}/local  CACHE PATH "" FORCE)
endif()

SET(CMAKE_SKIP_BUILD_RPATH  FALSE)
SET(CMAKE_BUILD_WITH_INSTALL_RPATH FALSE)
SET(CMAKE_INSTALL_RPATH "${CMAKE_INSTALL_PREFIX}/lib")
SET(CMAKE_INSTALL_RPATH_USE_LINK_PATH TRUE)
LIST(FIND CMAKE_PLATFORM_IMPLICIT_LINK_DIRECTORIES "${CMAKE_INSTALL_PREFIX}/lib" isSystemDir)
IF("${isSystemDir}" STREQUAL "-1")
   SET(CMAKE_INSTALL_RPATH "${CMAKE_INSTALL_PREFIX}/lib")
ENDIF("${isSystemDir}" STREQUAL "-1")
```

- **CMAKE_INSTALL_PREFIX** : specify install base directory path
- **set (CMAKE_INSTALL_PREFIX $ENV{HOME}/local CACHE PATH "" FORCE)** : installation by default in ${HOME}/local directory
- To change default install directory path, re-run :
  cmake .. **-DCMAKE_INSTALL_PREFIX=**/mynewpath/directory
  make install

Advanced example : anatomy (CMakeLists.txt)

```
FOREACH(main_exe ${main_src})
  ….
  ….
 INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)
ENDFOREACH()
```

- **INSTALL(TARGETS ${exe} RUNTIME DESTINATION bin)** : install every **exe** binary file to **installation_path/bin** directory

Advanced example : anatomy (from lib01/CMakeLists.txt)

```
INSTALL(FILES include/chello.h DESTINATION include)

INSTALL(FILES ${PROJECT_BINARY_DIR}/lib/libMYlib.so DESTINATION lib)
```

- **INSTALL(FILES include/chello.h DESTINATION include)** : install **chello.h** to **installation_path/include** directory
- **INSTALL(FILES ${PROJECT_BINARY_DIR}/lib/libMYlib.so DESTINATION lib)** : install library **libMYlib.so to** installation_path/lib directory

  Note : library comes from **${PROJECT_BINARY_DIR}/lib directory**

PART TWO : exercises

- Clone **cmake-tutos** project from gitlab.lam.fr

  ⇨ git clone https://gitlab.lam.fr/jclamber/cmake-tutos.git

Exercise : 01 (dir ex01)

- Use cmake to generate Makefile
  - mkdir build
  - cd build
  - cmake ..
- Run “ls -l” command in your build directory
- Compile
  - make
- Run
  - ./hello

Exercise : 2 (dir ex02)

- Use cmake to generate Makefile
  - mkdir build
  - cd build
  - cmake ..
- Compile
  - make
- Run
  - ./hello (crazy no ? :))

Exercise : 2 (dir ex02)

- Add a new “main” source file
  - Example “ src/mymain.cc
  - Recompile with **make** command (what happens ?)

Exercise : 2 (dir ex02)

- Add a new “main” source file
  - Example “ src/mymain.cc
  - Recompile with **make** command (what happens ?)

Solution: you must re-run cmake, when you add a new file

  - cd build
  - cmake ..
  - make

Exercise : 2 (dir ex02)

- Unix “**touch**” command, modify file date and time to the current date
  - touch lib01/hello.h and re-compile
  - touch lib01/insane.c and re-compile

Exercise : 2 (dir ex02)

- **make VERBOSE=1** command display complete compilation process
  - touch lib01/hello.h and re-compile

  This command is useful to check compilation parameters

Exercise : 2 (dir ex02)

- Install your project
  - make install (where does it install by default ?)

Exercise : 2 (dir ex02)

- Install your project
  - make install (where does it install by default ?)
- Install in another location
  - example : in /tmp/local

Exercise : 2 (dir ex02)

- Install your project
  - make install (where does it install by default ?)
- Install in another location
  - example : in /tmp/local

SOLUTION :

- cd build
- cmake .. -DCMAKE_INSTALL_PREFIX=/tmp/local
- make install

Exercise : 2 (dir ex02)

- Check binary dependency
  - Linux : **ldd** installation_path/bin/hello
  - MacOS : **otool -L** installation_path/bin/hello