

CMake

Cross-platform Make

Pierre AUBERT

<http://www.cmake.org/>

- 1 Why use CMake?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program
- 5 How to create a library
- 6 Library creation with subdirectory example
- 7 Some usefull things

- 1 Why use CMake?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program
- 5 How to create a library
- 6 Library creation with subdirectory example
- 7 Some usefull things

- Easier than Make
 - but the same way of thinking
 - generate the *Makefile*
- Separate the compilation from the sources
- Multi-platforms
- Very flexible
- Check if the libraries/programs are available on your system
- File generator (**configure__file**)
- Calling programs or scripts (**doxygen**)
- One of the new standards

- 1 Why use CMake ?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program
- 5 How to create a library
- 6 Library creation with subdirectory example
- 7 Some usefull things

Ubuntu/Debian

```
sudo apt-get install cmake
```

Mac

```
fink install cmake
```

Fedora

```
yum install cmake
```

Others

<https://cmake.org/download/>

- 1 Why use CMake ?
- 2 How to install CMake
- 3 Few basic examples**
- 4 How to install a program
- 5 How to create a library
- 6 Library creation with subdirectory example
- 7 Some usefull things

C++ program (main.cpp)

```
#include <iostream>
int main(int argc, char** argv){
    std::cout << "Hello world" << std::endl;
    return 0;
}
```

CMakeLists.txt

```
project(Example)
cmake_minimum_required(VERSION 2.8)

add_executable(test main.cpp)
```



```
cd path/to/your/project
mkdir build
cd build
cmake ..
make
./test
```

```
$cmake ..
```

- The C compiler identification is GNU 4.9.1
- The CXX compiler identification is GNU 4.9.1
- 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 :/1-HelloWorld/build

```
$ make
```

```
Scanning dependencies of target test
```

```
Building CXX object CMakeFiles/test.dir/main.cpp.o
```

```
Linking CXX executable test
```

```
Built target test
```

```
$ ./test
```

```
Hello world
```

- 1 Why use CMake ?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program**
- 5 How to create a library
- 6 Library creation with subdirectory example
- 7 Some usefull things

C++ program (main.cpp)

```
#include <iostream>
int main(int argc, char** argv){
    std::cout << "Hello world" << std::endl;
    return 0;
}
```

CMakeLists.txt

```
project(Example)
cmake_minimum_required(VERSION 2.8)
add_executable(test main.cpp)
install(TARGETS test RUNTIME DESTINATION bin)
```

Install options

- **TARGETS** : install the programs and libraries
- **FILES** : install files like headers, configs...
- **PROGRAMS** : install executable scripts (bash python ...)
- **DIRECTORY** : install a directory (documentation)

Where is my program now ?

In the install path

- **CMAKE_INSTALL_PREFIX** : by default */usr/local*

In the terminal

```
cmake .. -DCMAKE_INSTALL_PREFIX=$HOME/usr
```

In the *CMakeLists.txt* (But it's not a good idea)

```
project(Example)
cmake_minimum_required(VERSION 2.8)
set(CMAKE_INSTALL_PREFIX my/install/prefix)
add_executable(test main.cpp)
install(TARGETS test RUNTIME DESTINATION bin)
```

- 1 Why use CMake ?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program
- 5 How to create a library**
- 6 Library creation with subdirectory example
- 7 Some usefull things

Library (shadok.h)

```
#ifndef __SHADOK_H__
#define __SHADOK_H__
#include <string>
void myPrint(const std::string & str);
#endif
```

Library (shadok.cpp)

```
#include <iostream>
#include "shadok.h"
void myPrint(const std::string & str){
    std::cout << str << std::endl;
}
```

C++ program (main.cpp)

```
#include <iostream>
#include "shadok.h"
int main(int argc, char** argv){
    myPrint("Hello world");
    return 0;
}
```

CMakeLists.txt

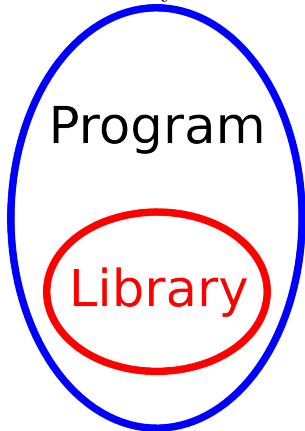
```
project(Example)
cmake_minimum_required(VERSION 2.8)

add_library(shadok SHARED shadok.cpp)
install(TARGETS shadok DESTINATION lib)

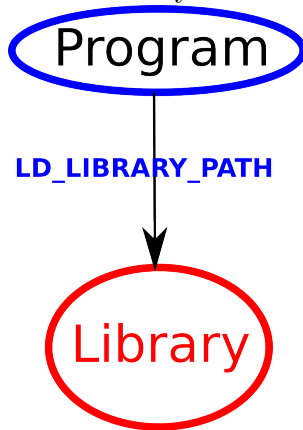
add_executable(test main.cpp)
target_link_libraries(test shadok)

install(TARGETS test RUNTIME DESTINATION bin)
```

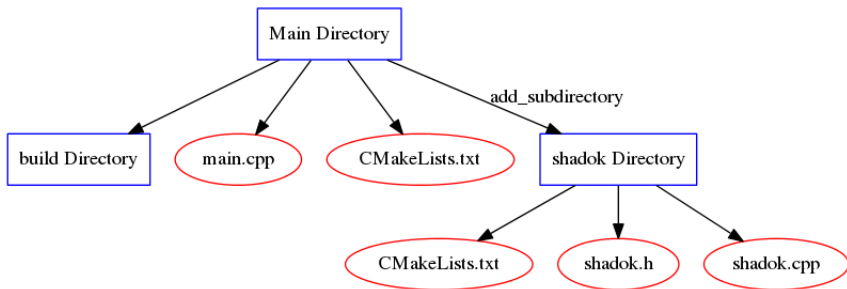

Static library



Shared library



- 1 Why use CMake ?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program
- 5 How to create a library
- 6 Library creation with subdirectory example**
- 7 Some usefull things



Lib directory *shadok/CMakeLists.txt*

```
project(Example)
cmake_minimum_required(VERSION 2.8)

add_library(shadok SHARED shadok.cpp)
install(TARGETS shadok DESTINATION lib)
```

Main directory *CMakeLists.txt*

```
project(Example)
cmake_minimum_required(VERSION 2.8)

include_directories(shadok)

add_executable(test main.cpp)
target_link_libraries(test shadok)

install(TARGETS test RUNTIME DESTINATION bin)

add_subdirectory(shadok)
```

- 1 Why use CMake ?
- 2 How to install CMake
- 3 Few basic examples
- 4 How to install a program
- 5 How to create a library
- 6 Library creation with subdirectory example
- 7 Some usefull things

Usefull CMake variables

- `${CMAKE_INSTALL_PREFIX}` : the install directory
- `${CMAKE_CURRENT_SOURCE_DIR}` : the directory of the current *CMakeLists.txt*
- `${CMAKE_CURRENT_BINARY_DIR}` : the build directory of the current *CMakeLists.txt*
- `${CMAKE_MODULE_PATH}` : the directory of the modules/libraries finders

Usefull CMake variables

- `include_directories` : like the `-I`
- `link_directories` : like the `-L`

Get all the sources files in the current directory

```
file(GLOB sources
"${CMAKE_CURRENT_SOURCE_DIR}/*.cpp")
add_executable(prog ${sources})
```



Thanks for your attention