CMake introduction

https://hpcleuven.github.io/CMake-intro/

Vlaanderen is supercomputing

VLAAMS SUPERCOMPUTER CENTRUM | *Innovative Computing for A Smarter Flanders*

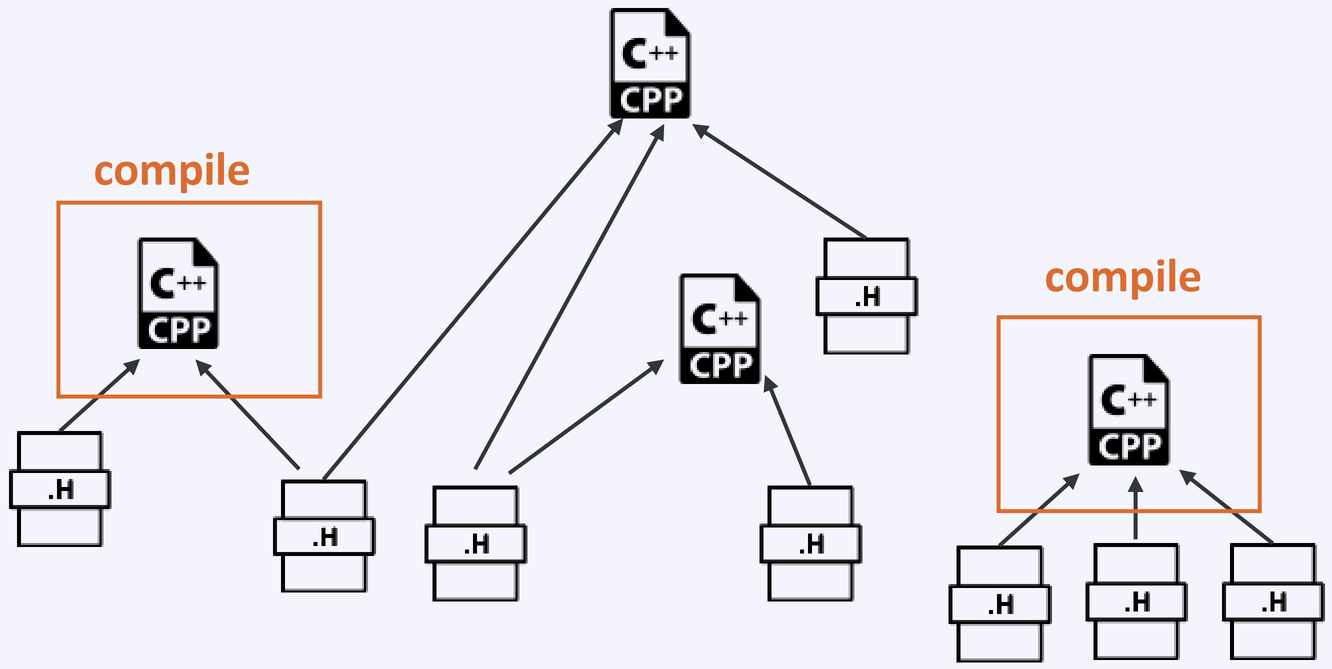
vscentrum.be

Build system

- Build system is the collection of tools for automating program compilations
- At the core there is normally a functional based language which maps a set of sources (files) to the target (executable, library)
- Build systems:
  - Make
  - Ninja
  - Ant
  - Gradle

Build systems

- Why – the number of files can go to hundreds
- Challenge to keep track of files and dependencies
- Compilation may take a lot of files, not well maintained – need to start from beginning
- Build systems – automated source code compilation and linking process
- `$ g++ main.cpp addition.cpp division.cpp print_result.cpp -o calculator`

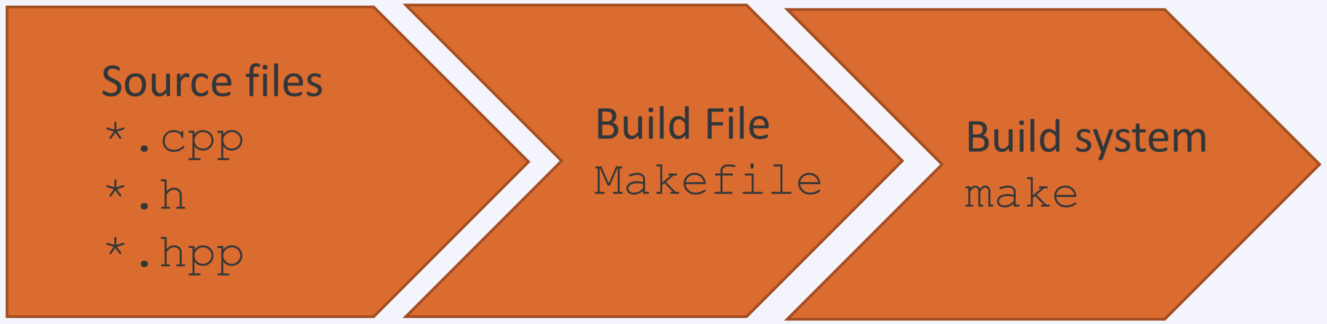


VLAAMS SUPERCOMPUTER CENTRUM

Build systems

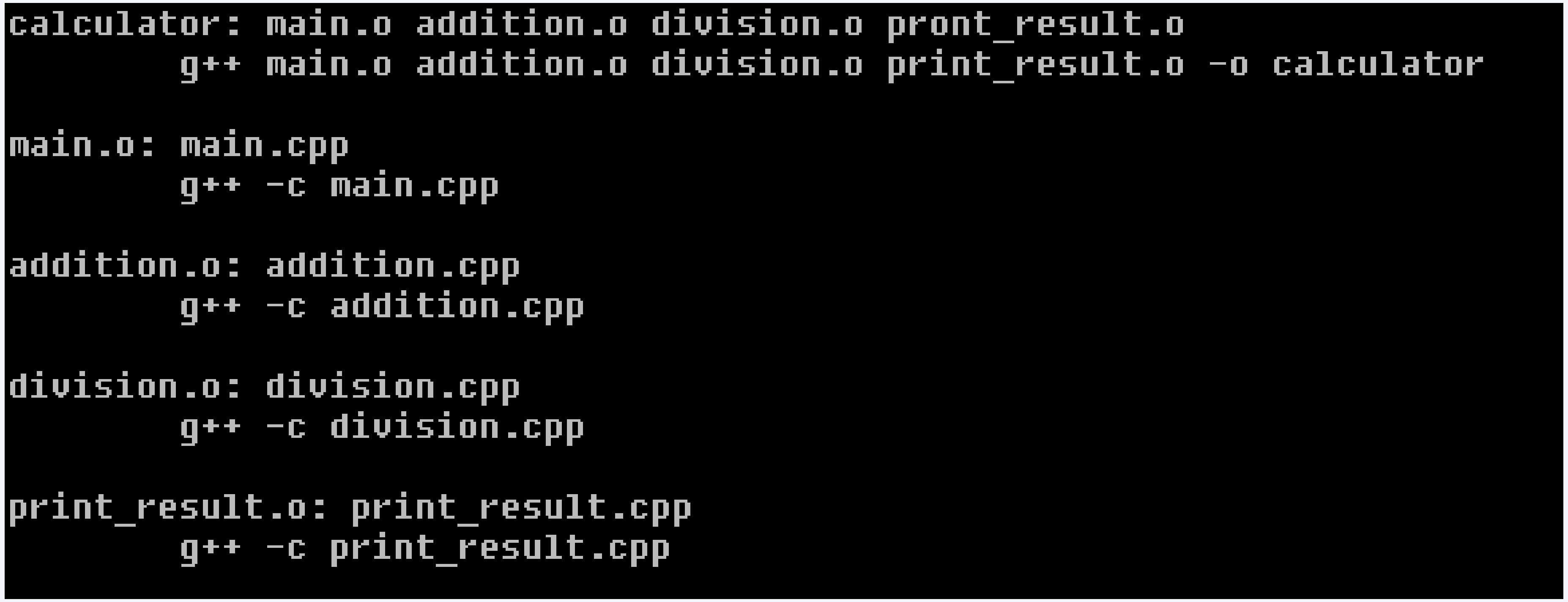
- Build file – to compile and link source code

Build process



Build systems

- Build file – to compile and link source code



```
calculator: main.o addition.o division.o pront_result.o
        g++ main.o addition.o division.o print_result.o -o calculator

main.o: main.cpp
        g++ -c main.cpp

addition.o: addition.cpp
        g++ -c addition.cpp

division.o: division.cpp
        g++ -c division.cpp

print_result.o: print_result.cpp
        g++ -c print_result.cpp
```

VLAAMS SUPERCOMPUTER CENTRUM

Build systems

- CMake is able to write Makefiles for us
- Why – building of the project is not standard across platfoms - > CMake will create the platform-based build system files

- Make vs CMake
  - Make - Uses build system files to generate executable
  - CMake - Used to generate build system files

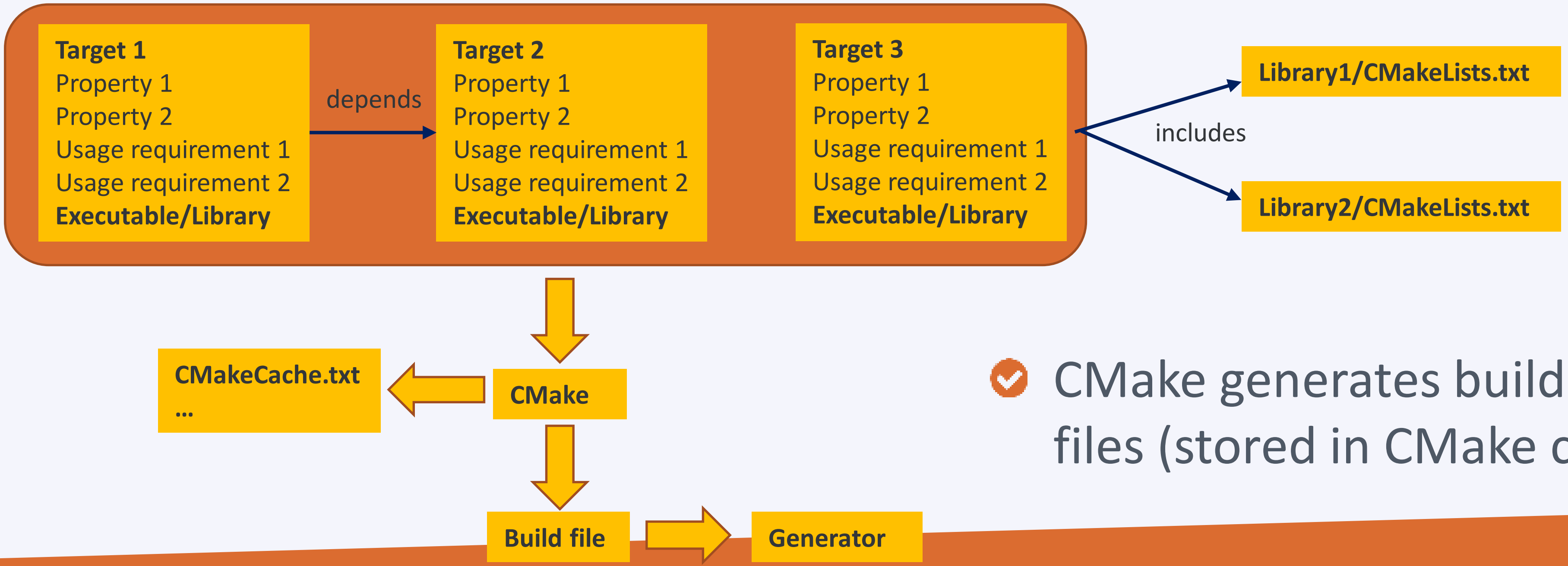
CMake

- CMake is the cross-platform make,
- It is a build system manager
- Can build multiple make systems (like make - default)
- Build systems are called generators
- Provides possibilities to manage all unit tests (ctest)

- Why CMake?
  - Open source
  - Can be used for multiple programming languages

CMake

- Treat as project-oriented programming
- CMake is organized in targets (libraries, executables created in the end)
- Every target contains
  - properties (e.g., compiler options) and
  - usage requirements (e.g., dependencies between targets)



- CMake generates build files and meta files (stored in CMake cache)

VLAAMS SUPERCOMPUTER CENTRUM

CMake on the cluster

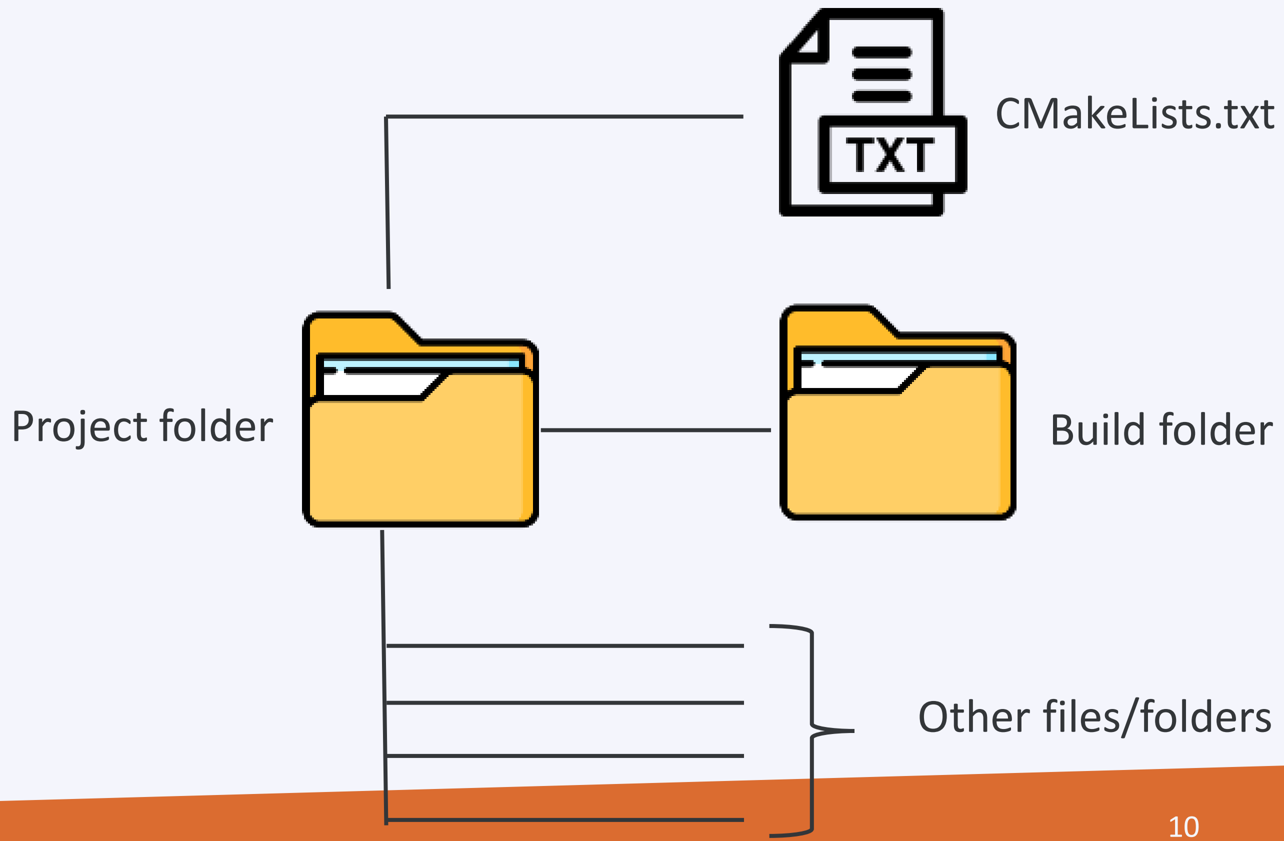
- Which module to choose on the cluster – avoid conflicts (go for GCC/GCCcore)
- Compiled code and source files available – in case new version necessary

- `$module load CMake/3.22.1-GCCcore-10.3.0`

Requirements to run CMake

- CMakeLists.txt file (fixed name – renamed will cause CMake error)
- Separate directory to store build system files (can be anything)

Using CMake



example0

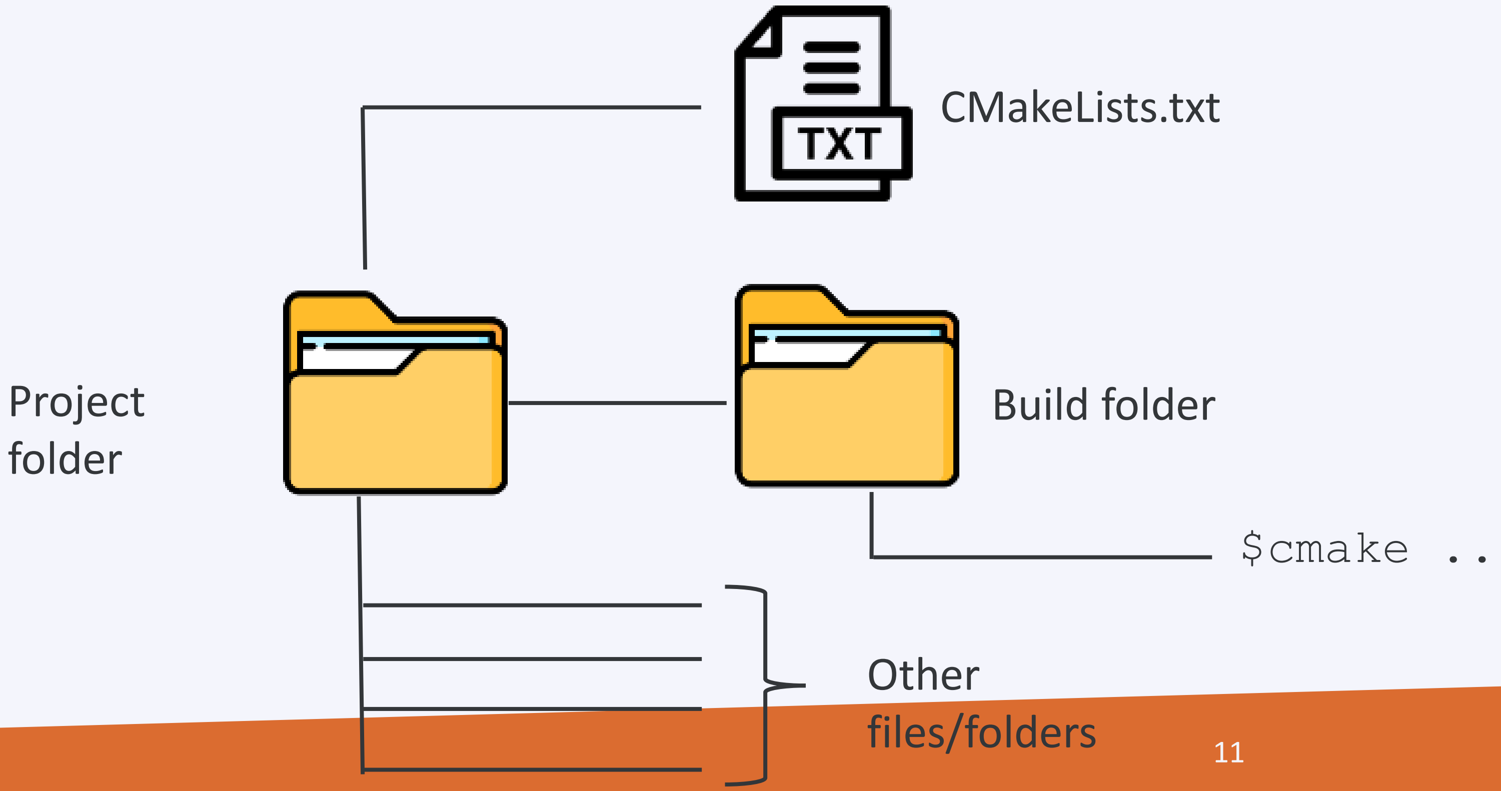
VLAAMS SUPERCOMPUTER CENTRUM

Requirements to run CMake

- ✓ CMakeLists.txt file (fixed name – renamed will cause CMake error)
- ✓ Separate directory to store build system files (can be anything)

`cmake ..` – run in the build dir,

.. – informs that CMakeLists.txt is in the main dir



example0

Requirements to run CMake

- ✓ CMakeLists.txt file (fixed name – renamed will cause CMake error)
- ✓ Separate directory to store build system files (can be anything)

`cmake ..` – run in the build dir,

`..` – informs that CMakeLists.txt is in the parent folder

`Makefile` created – we can run make

VLAAMS SUPERCOMPUTER CENTRUM

CMakeLists.txt

- Used to create executable
- Name of executable should be the 1st argument
- CMakeLists.txt composition

```
command1 (arg_a1 arg_a2 arg_a3 … )

command2 (arg_b1 arg_b2 arg_b3 … )

command3 (arg_c1
arg_c2
… )

command4 (arg_d1 arg_d2 arg_d3 … )
```

Project

- project() - Sets the name of the project, and stores it in the variable PROJECT_NAME. When called from the top-level CMakeLists.txt also stores the project name in the variable CMAKE_PROJECT_NAME.

```
project(<project-name> VERSION
<major.minor.patch>  … )
```

VLAAMS
SUPERCOMPUTER
CENTRUM

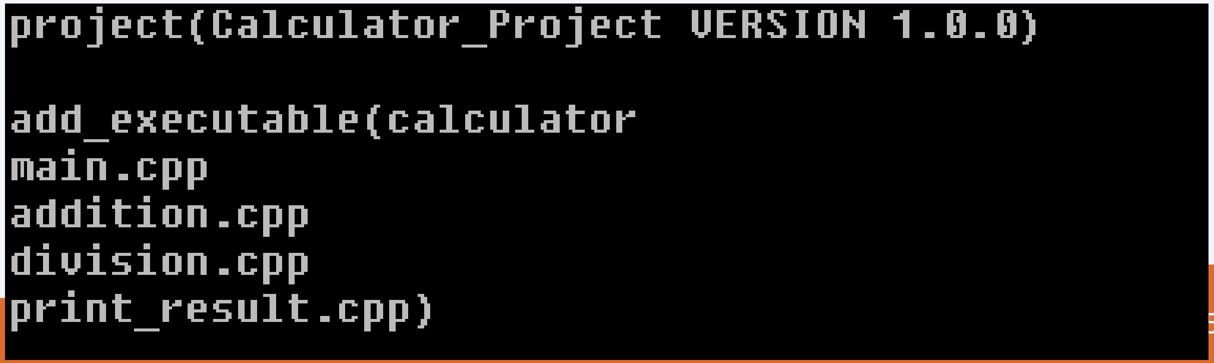
add_executable

- add_executable – executable with the given name is created
- executable created from compiling the files given after the name
- Arguments

```
add_executable(
calculator        ✓
addition.cpp
division.cpp
print_result.cpp
main.cpp)
```

```
add_executable(
calculator        ✓
main.cpp
addition.cpp
division.cpp
print_result.cpp)
```

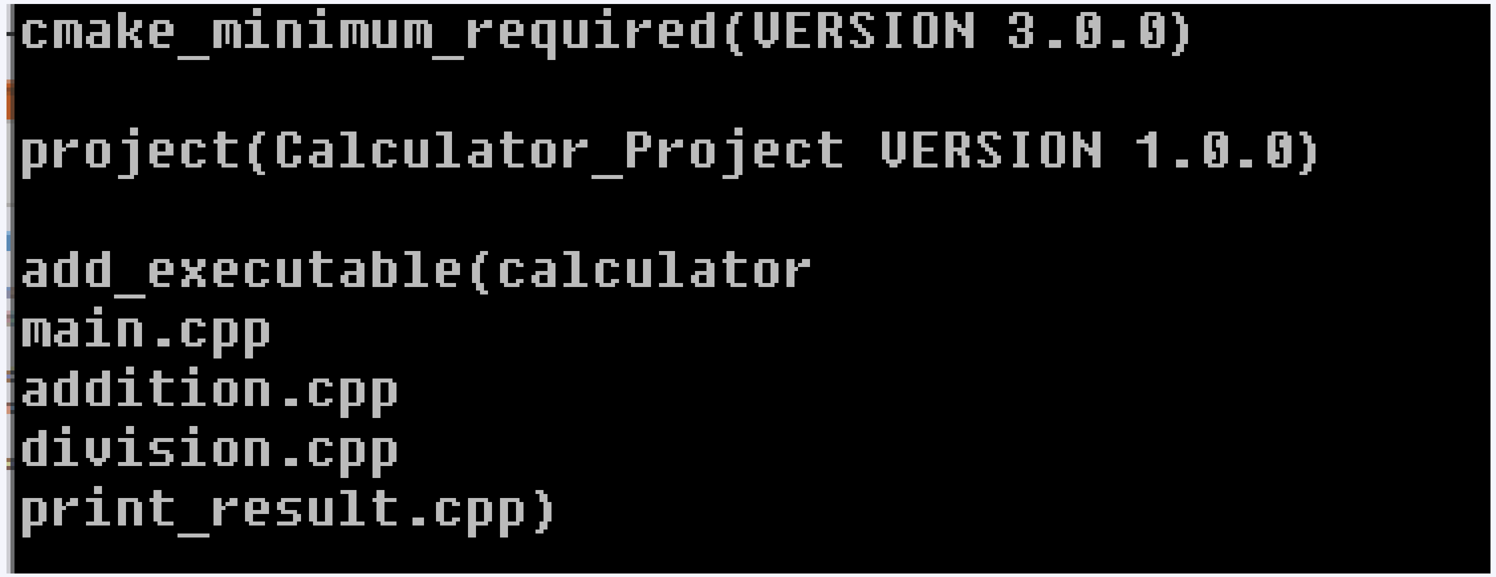
```
add_executable(
addition.cpp
calculator        ✗
division.cpp
print_result.cpp
main.cpp)
```



```
project(Calculator_Project VERSION 1.0.0)

add_executable(calculator
main.cpp
addition.cpp
division.cpp
print_result.cpp)
```

cmake_minimum_required

- Minimum version requirement – developer must test – depending on features used



```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)

add_executable(calculator
main.cpp
addition.cpp
division.cpp
print_result.cpp)
```

example1

VLAAMS
SUPERCOMPUTER
CENTRUM

add_library

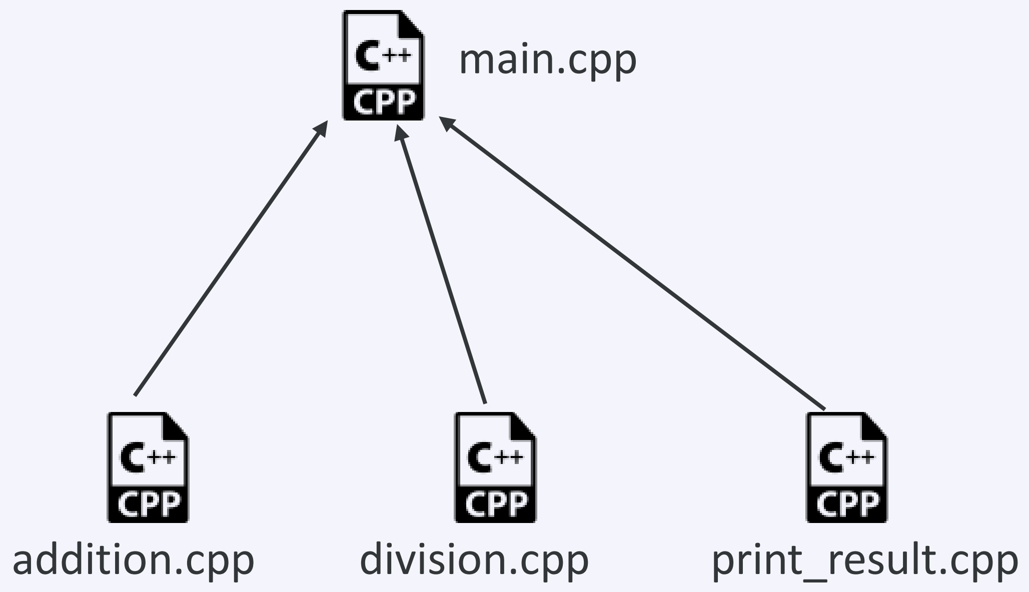
- add_library - To add a library we use the add_library() command and specify which source files should make up the library. Rather than placing all of the source files in one directory, we can organize our project with one or more subdirectories. In this case, we will create a subdirectory specifically for our library.

```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)

add_library(my_math
addition.cpp
division.cpp)

add_library(my_print
print_result.cpp)

add_executable(calculator
main.cpp)
```

example2

Hierarchy

- Making project modular - hierarchy



VLAAMS SUPERCOMPUTER CENTRUM

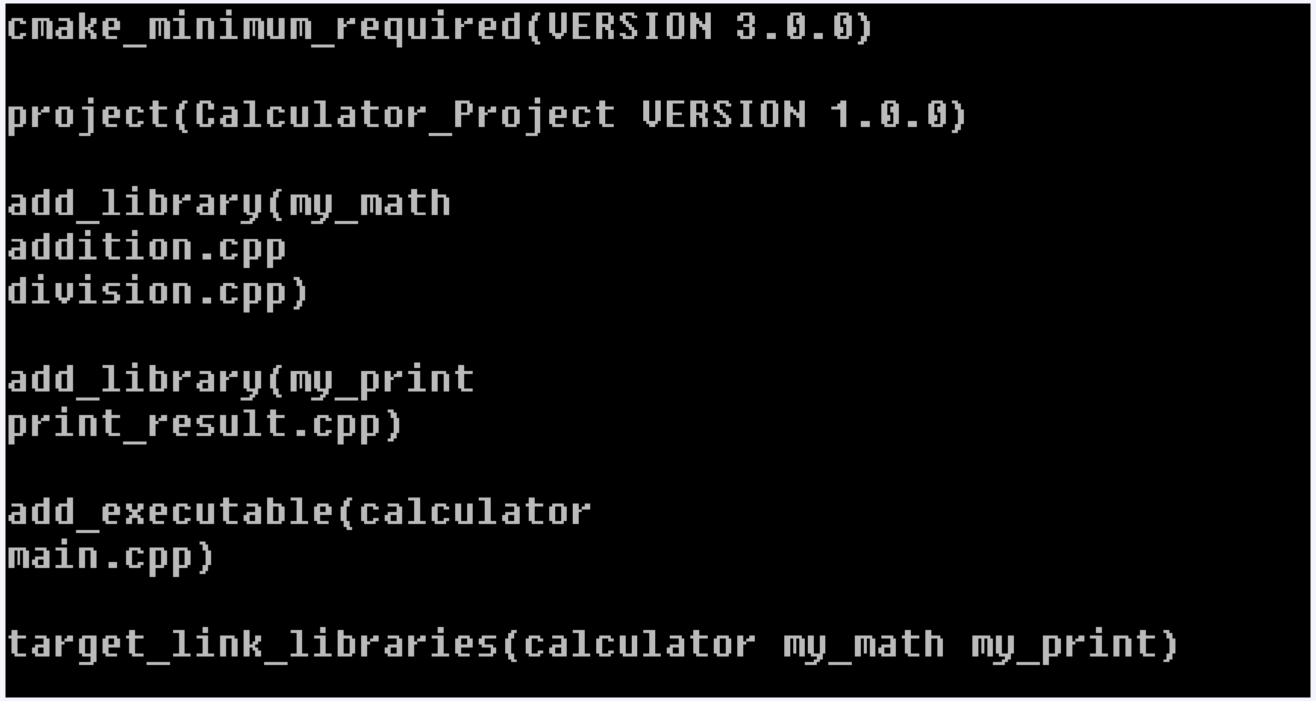
target_link_libraries

- target_link_libraries is responsible for adding a library into the linker's command line. If you use some library but do not specify it for the linker, you will get an "undefined reference" (or an "unresolved externals") error when creating an executable or a shared library

```
target_link_libraries(<executable> <lib1> <lib2>)
```

target_link_libraries

- Target link libraries



```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)

add_library(my_math
addition.cpp
division.cpp)

add_library(my_print
print_result.cpp)

add_executable(calculator
main.cpp)

target_link_libraries(calculator my_math my_print)
```

example3

VLAAMS
SUPERCOMPUTER
CENTRUM

Targets

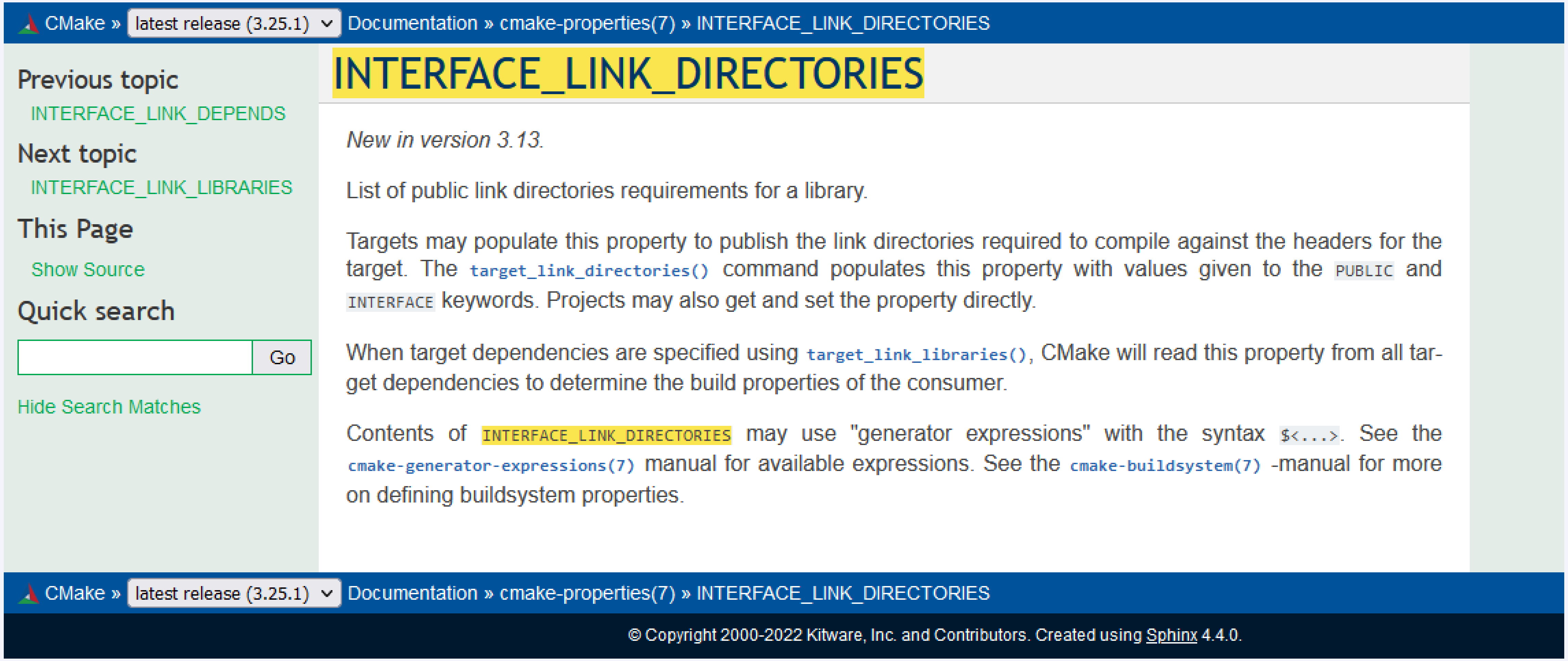
- **Targets**
  - Libraries
  - Executables

Target Properties

- Every target has properties and dependencies
- Hint: check CMake help webpage https://cmake.org/cmake/help/latest

- **Target Properties**
  - INTERFACE_LINK_DIRECTORIES
  - INCLUDE_DIRECTORIES
  - VERSION
  - SOURCES

Target Properties



CMake » latest release (3.25.1) Documentation » cmake-properties(7) » INTERFACE_LINK_DIRECTORIES

Previous topic
INTERFACE_LINK_DEPENDS

Next topic
INTERFACE_LINK_LIBRARIES

This Page
Show Source

Quick search

Go

Hide Search Matches

INTERFACE_LINK_DIRECTORIES

New in version 3.13.

List of public link directories requirements for a library.

Targets may populate this property to publish the link directories required to compile against the headers for the target. The `target_link_directories()` command populates this property with values given to the `PUBLIC` and `INTERFACE` keywords. Projects may also get and set the property directly.

When target dependencies are specified using `target_link_libraries()`, CMake will read this property from all target dependencies to determine the build properties of the consumer.

Contents of `INTERFACE_LINK_DIRECTORIES` may use "generator expressions" with the syntax `$<...>`. See the `cmake-generator-expressions(7)` manual for available expressions. See the `cmake-buildsystem(7)` -manual for more on defining buildsystem properties.

CMake » latest release (3.25.1) Documentation » cmake-properties(7) » INTERFACE_LINK_DIRECTORIES

© Copyright 2000-2022 Kitware, Inc. and Contributors. Created using Sphinx 4.4.0.

VLAAMS SUPERCOMPUTER CENTRUM

Properties

- Modify or retrieve properties
- In CMake, we can set target properties as either PRIVATE, PUBLIC, or INTERFACE. Both PUBLIC and INTERFACE properties are inherited by any targets that depend on the current target.

set_target_properties()

set_property

get_property

get_target_property()

Properties

Include Inheritance	Description
PUBLIC	All the directories following PUBLIC will be used for the current target and the other targets that have dependencies on the current target, i.e., appending the directories to INCLUDE_DIRECTORIES and INTERFACE_INCLUDE_DIRECTORIES.
PRIVATE	All the include directories following PRIVATE will be used for the current target only, i.e., appending the directories to INCLUDE_DIRECTORIES.
INTERFACE	All the include directories following INTERFACE will NOT be used for the current target but will be accessible for the other targets that have dependencies on the current target, i.e., appending the directories to INTERFACE_INCLUDE_DIRECTORIES.

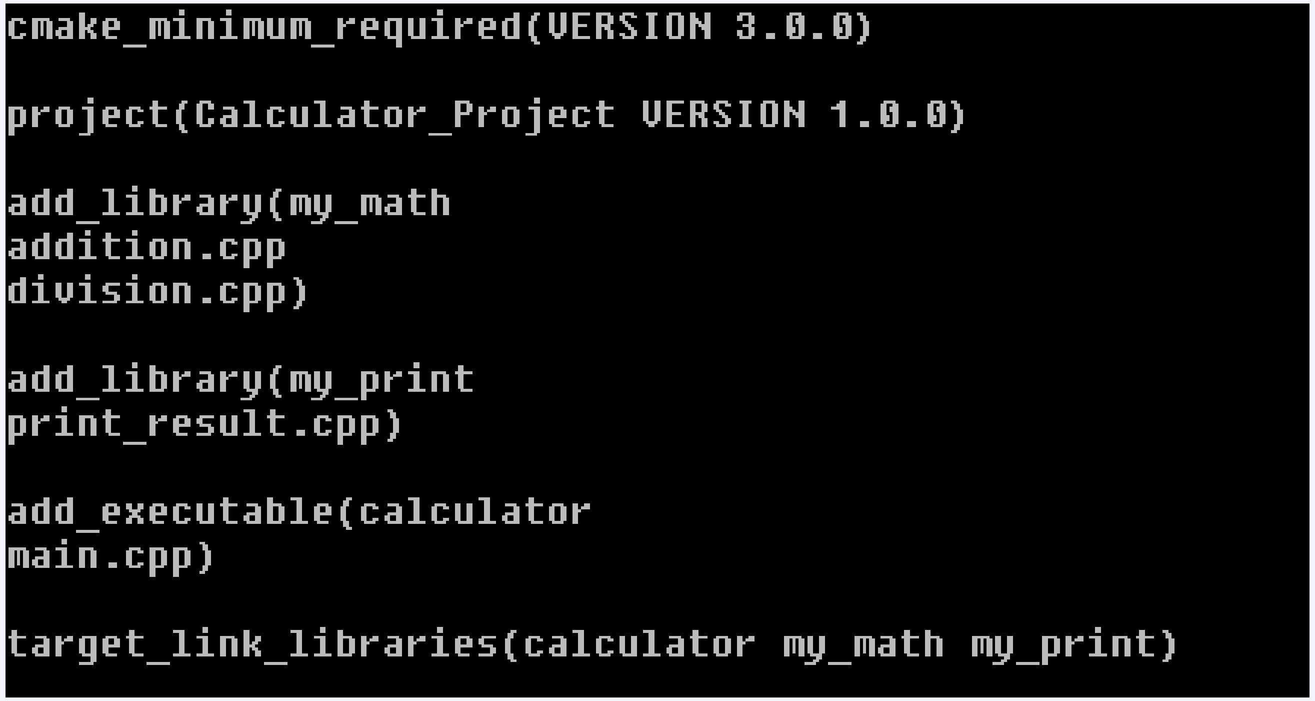
Target dependecy

- **Target dependency**

$$\begin{tikzcd}
\text{Target A} \\
\text{Target B (Dependency of Target A)} \arrow[u] \\
\text{Target C (Dependency of Target B)} \arrow[u]
\end{tikzcd}$$

Target vs. dependency

- calculator is the **target**
- my_math and my_print are the **dependencies**



```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)

add_library(my_math
addition.cpp
division.cpp)

add_library(my_print
print_result.cpp)

add_executable(calculator
main.cpp)

target_link_libraries(calculator my_math my_print)
```

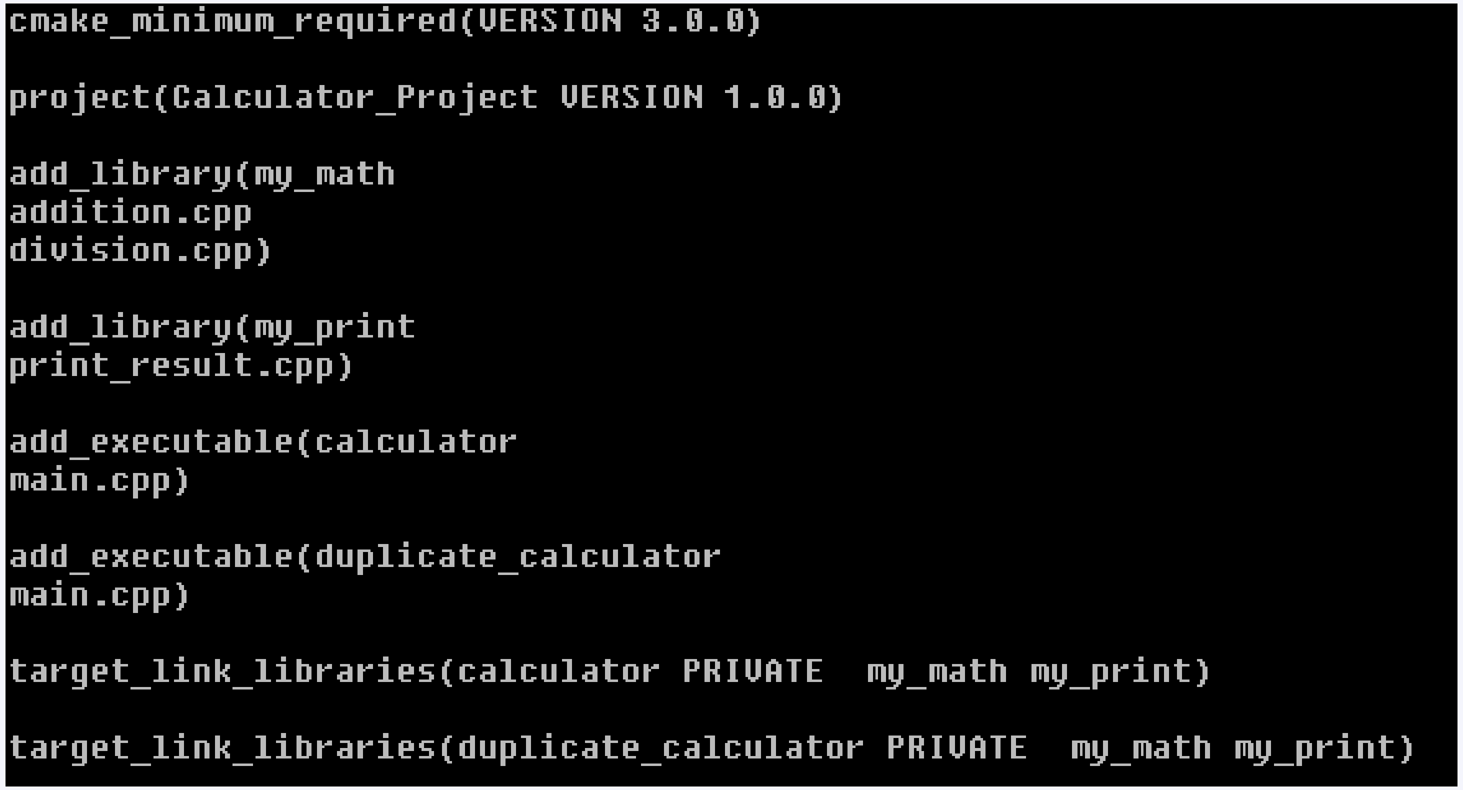
Target properties

- PRIVATE, PUBLIC, INTERFACE
- Propagating target properties

- target_link_libraries (myapp PUBLIC <item1> <item2>)
  - Property: INTERFACE_LINK_LIBRARIES
- target_link_libraries (myapp INTERFACE <item1> <item2>)
  - Property: INTERFACE_LINK_LIBRARIES

Requirements to run CMake

- Can we have multiple executables in CMakelists
- Yes

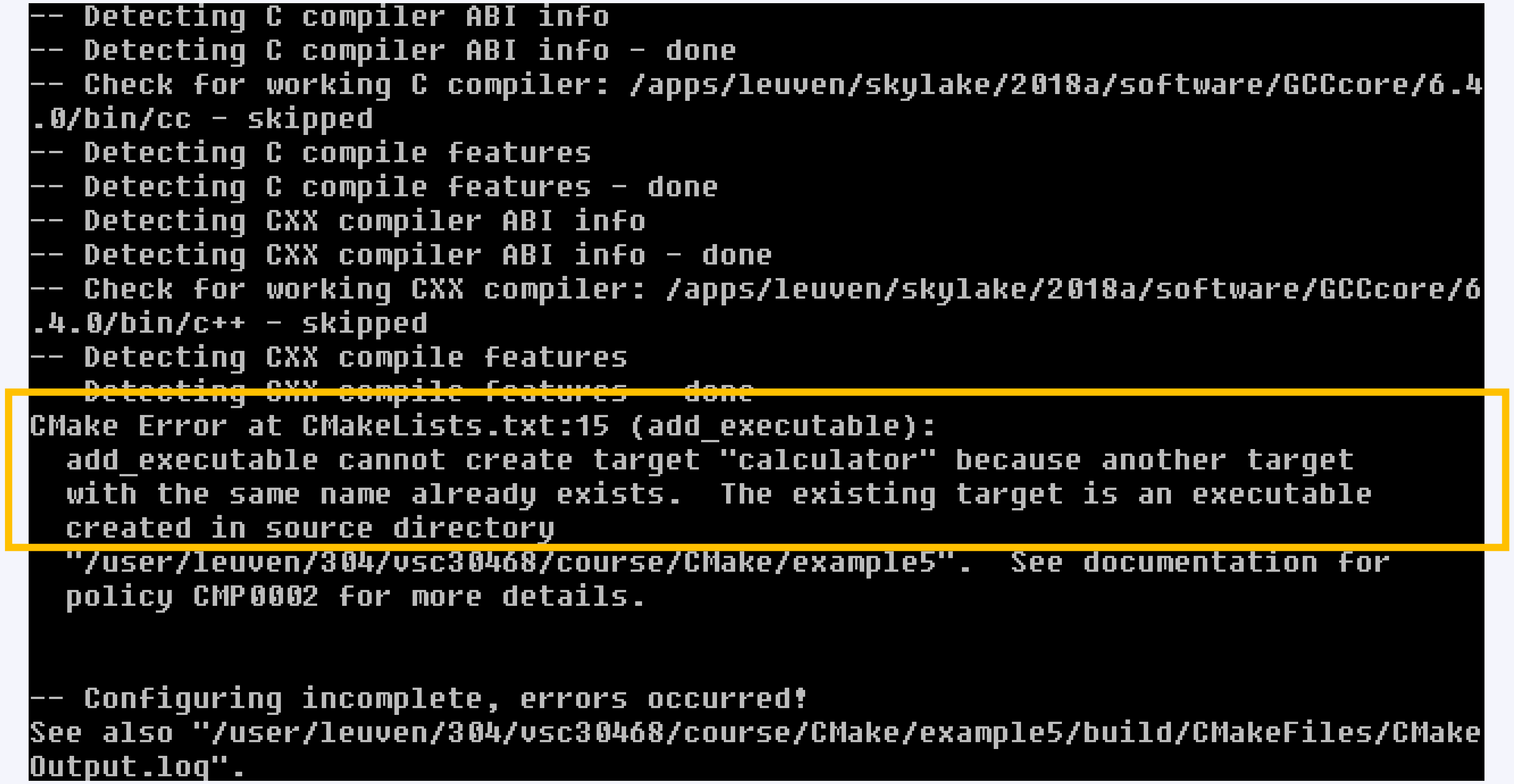


```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)

add_library(my_math
addition.cpp
division.cpp)

add_library(my_print
print_result.cpp)

add_executable(calculator
main.cpp)

add_executable(duplicate_calculator
main.cpp)

target_link_libraries(calculator PRIVATE  my_math my_print)

target_link_libraries(duplicate_calculator PRIVATE  my_math my_print)
```

example4

Requirements to run CMake

- Can we have 2 targets with the same name – no
- CMake will return the error



```
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /apps/leuven/skylake/2018a/software/GCCcore/6.4
.0/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /apps/leuven/skylake/2018a/software/GCCcore/6
.4.0/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
CMake Error at CMakeLists.txt:15 (add_executable):
  add_executable cannot create target "calculator" because another target
  with the same name already exists.  The existing target is an executable
  created in source directory
  "/user/leuven/304/vsc30468/course/CMake/example5".  See documentation for
  policy CMP0002 for more details.


-- Configuring incomplete, errors occurred!
See also "/user/leuven/304/vsc30468/course/CMake/example5/build/CMakeFiles/CMake
Output.log".
```

example5

VLAAMS SUPERCOMPUTER CENTRUM

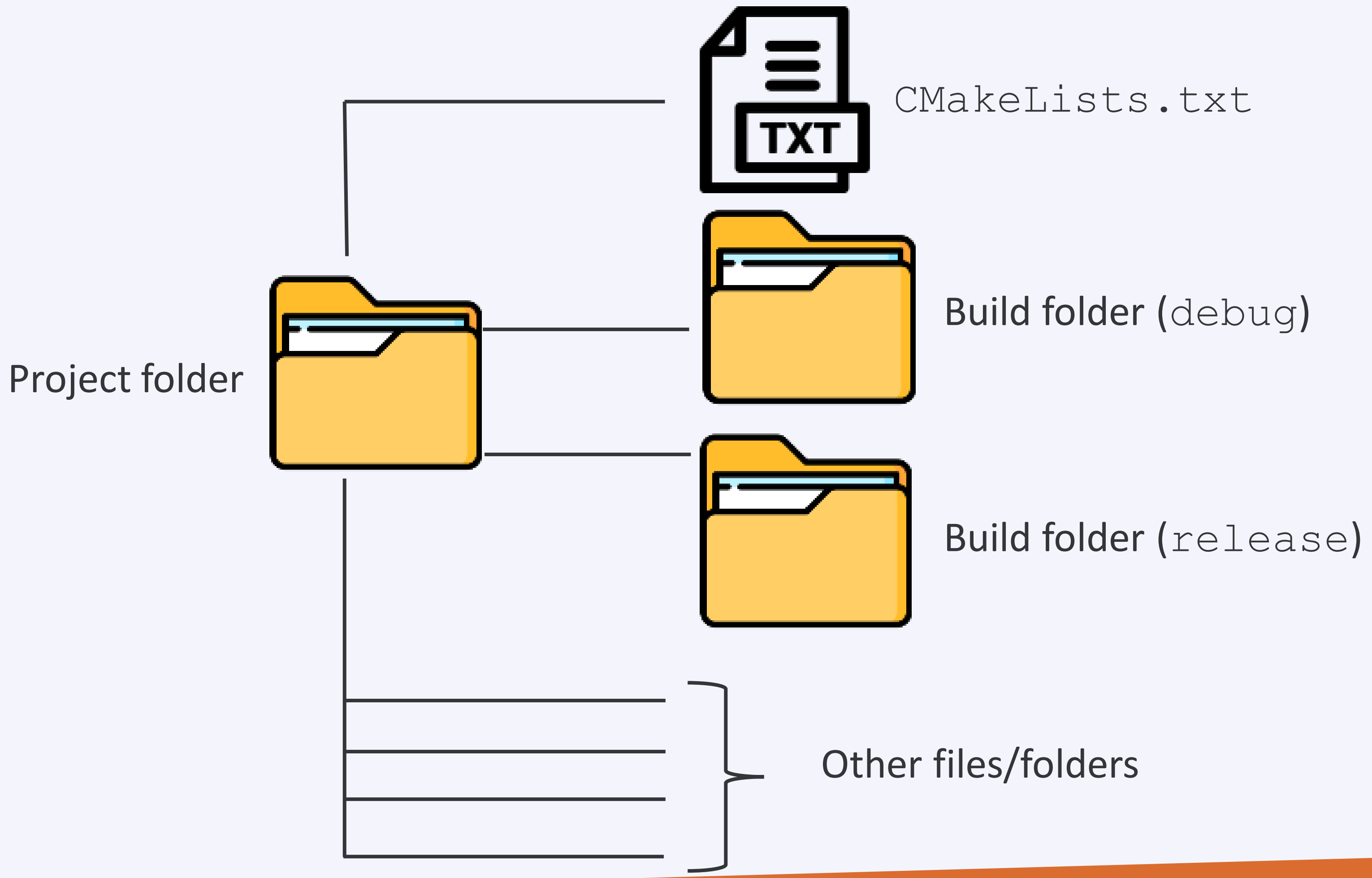
Target files: prefix and suffix

- Do we have target files saved on computer – yes
- In build dir
- Actual name can be different
- Library `.lib .dll .so .a` – depending on operating system
- Prefix and extension
  - calculator -> calculator.exe
  - my_math -> my_math.lib
  - my_math -> libmy_math.so

VLAAMS SUPERCOMPUTER CENTRUM

Subdirectories

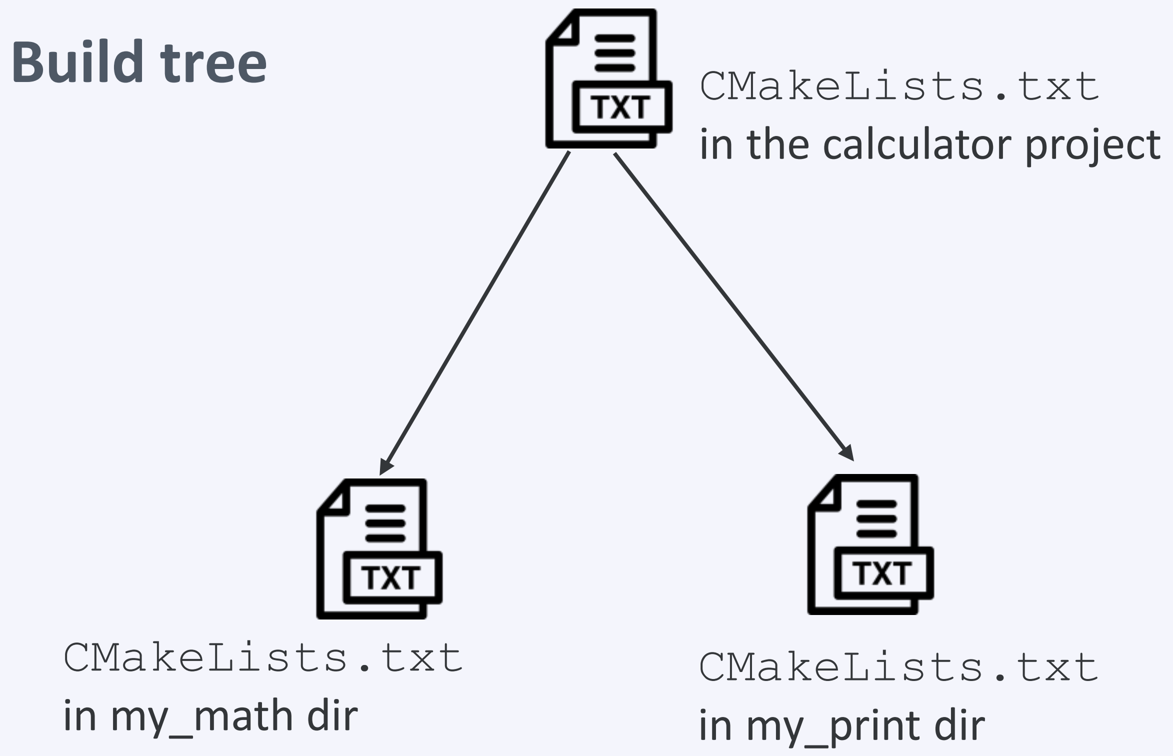
- Build or debug+release for separate compiler options



VLAAMS SUPERCOMPUTER CENTRUM

Subdirectories

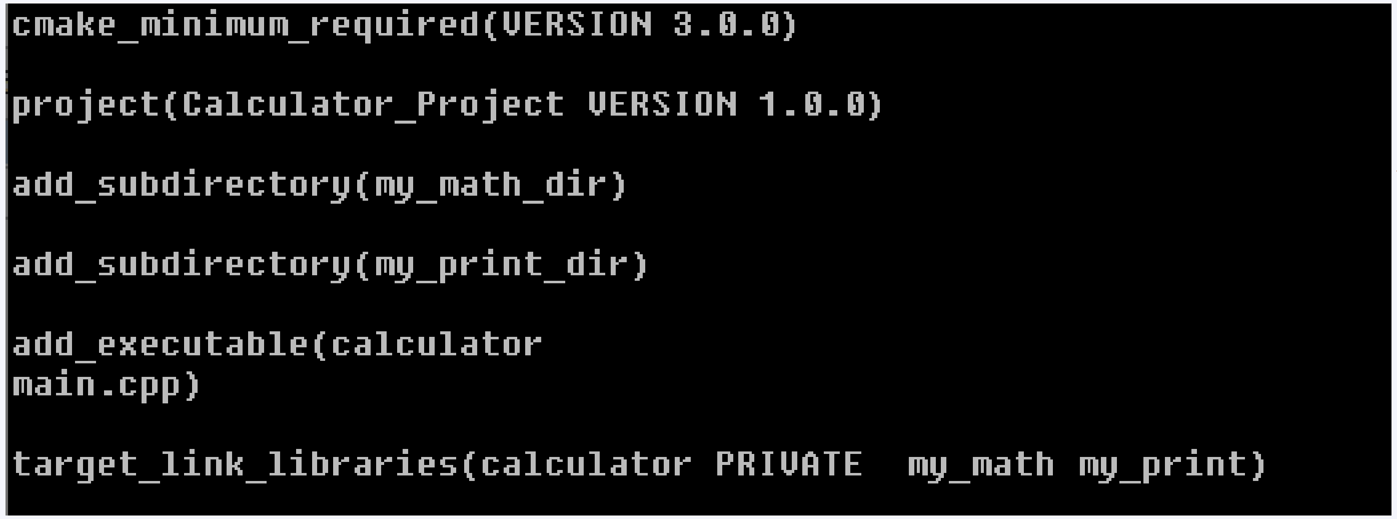
- Subdirs – to go to subdirectory, find another CMakefile.txt file there and run one by one



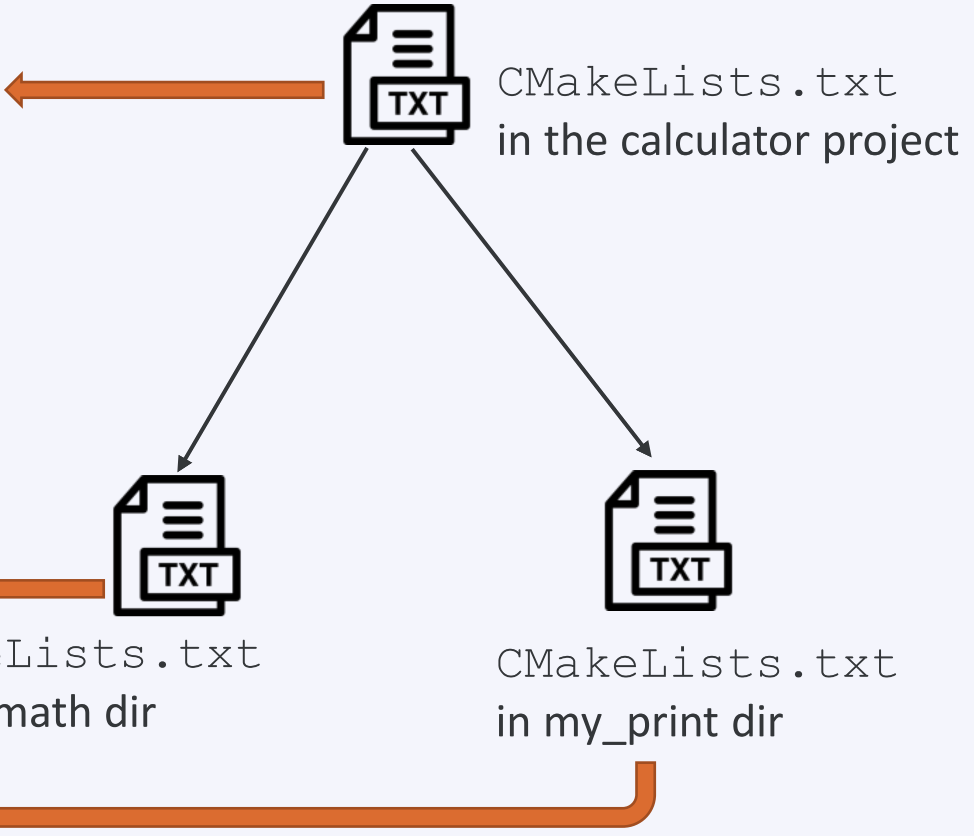
VLAAMS SUPERCOMPUTER CENTRUM

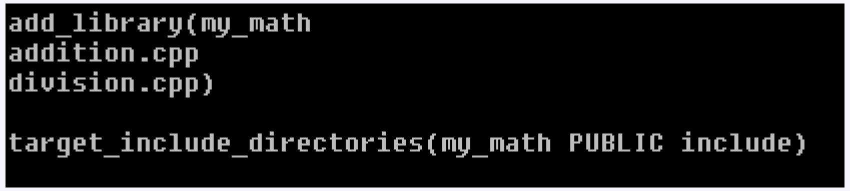
Subdirectories

- Subdirs – to go to subdirectory, find another CMakefile.txt file there and run one by one

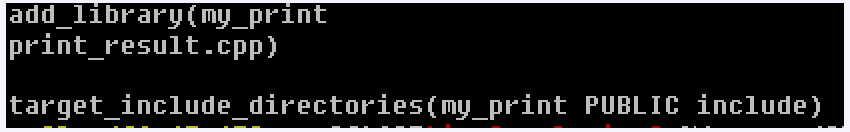


```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)

add_subdirectory(my_math_dir)

add_subdirectory(my_print_dir)

add_executable(calculator
main.cpp)

target_link_libraries(calculator PRIVATE  my_math my_print)
```





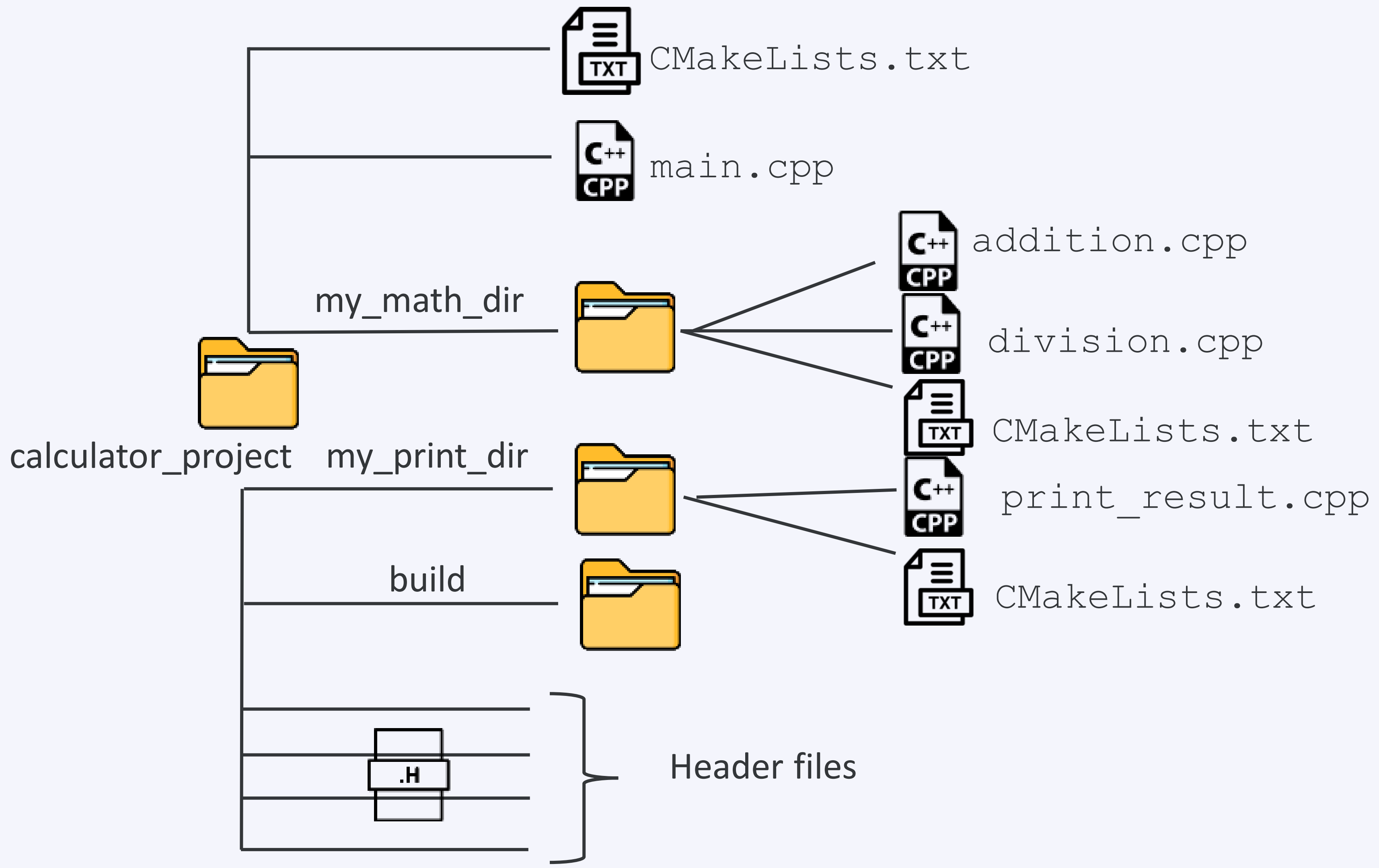
```
add_library(my_math
addition.cpp
division.cpp)

target_include_directories(my_math PUBLIC include)
```



```
add_library(my_print
print_result.cpp)

target_include_directories(my_print PUBLIC include)
```

VLAAMS SUPERCOMPUTER CENTRUM

Subdirectories

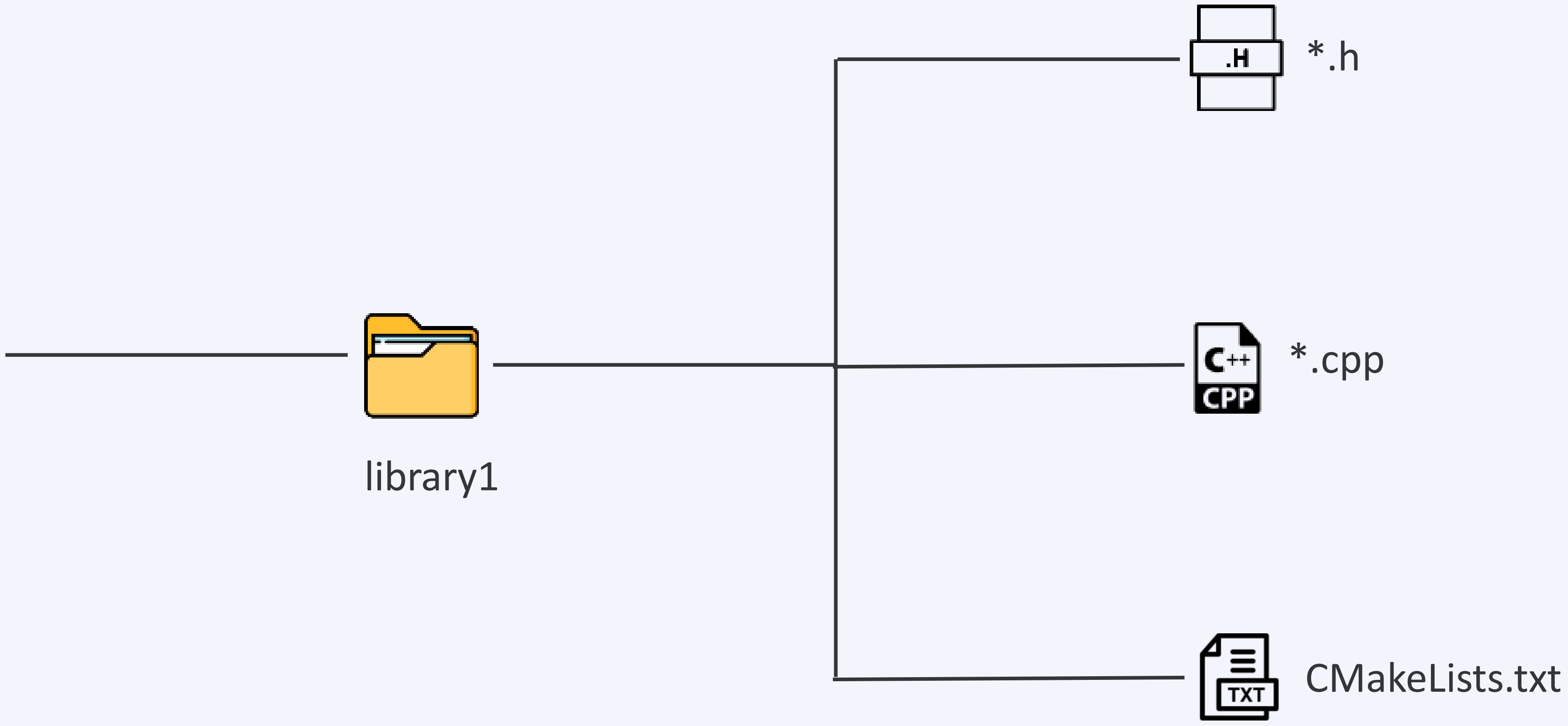


example6

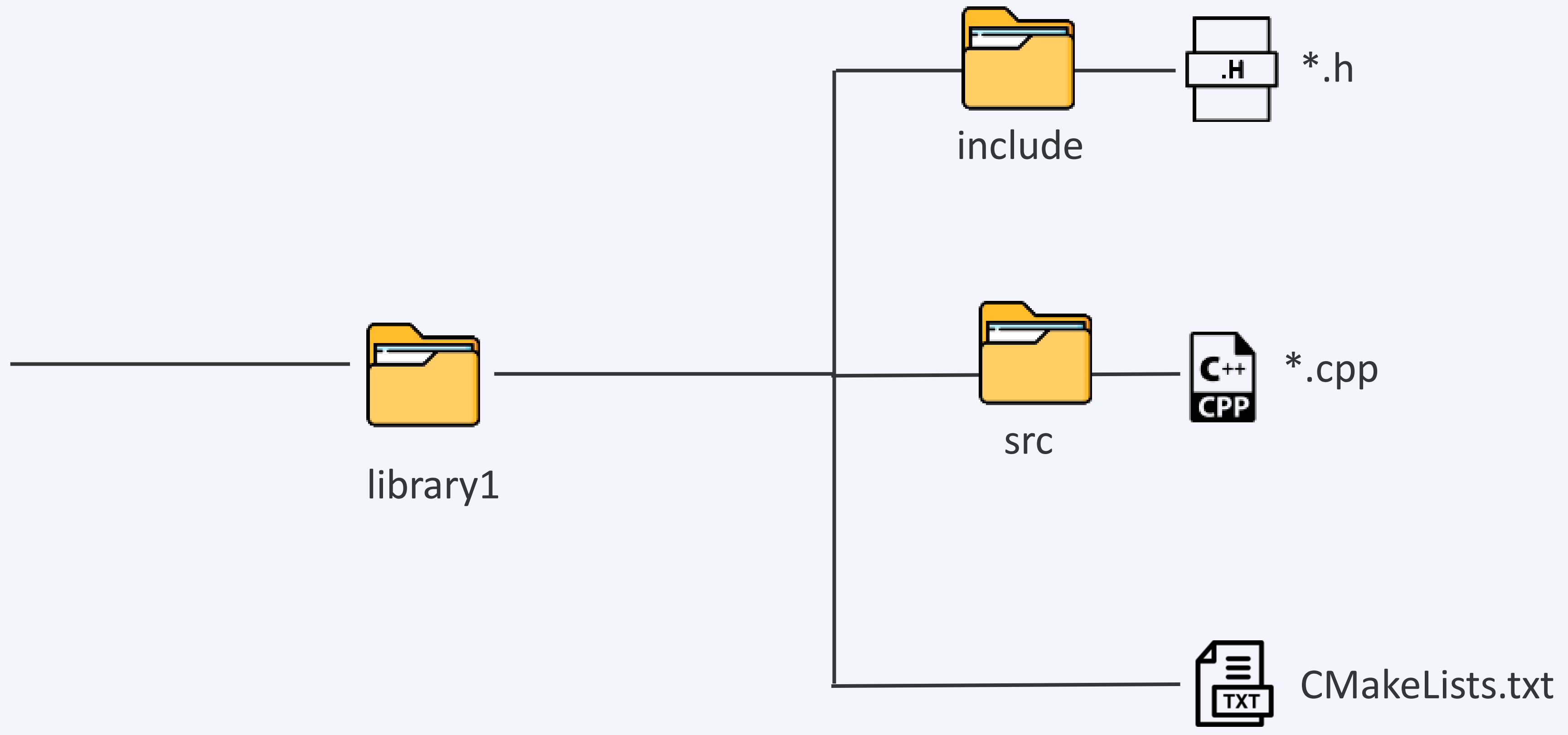
VLAAMS SUPERCOMPUTER CENTRUM

Library

- Good practice – keep files belonging to one library in the same place



Library Folder Structure



VLAAMS SUPERCOMPUTER CENTRUM

Libraries

- Avoiding compatibility issues

addition.cpp

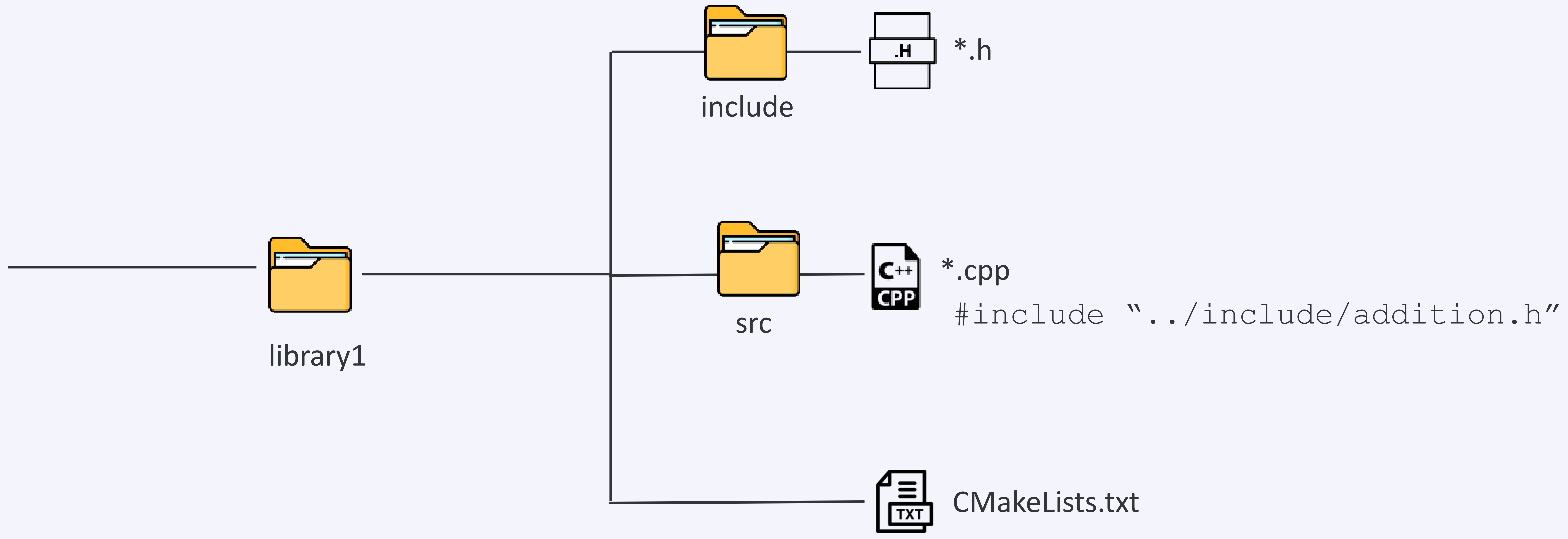
```
#include <addition.h>
…
…
…
```

division.cpp

```
#include <division.h>
…
…
…
```

Header files

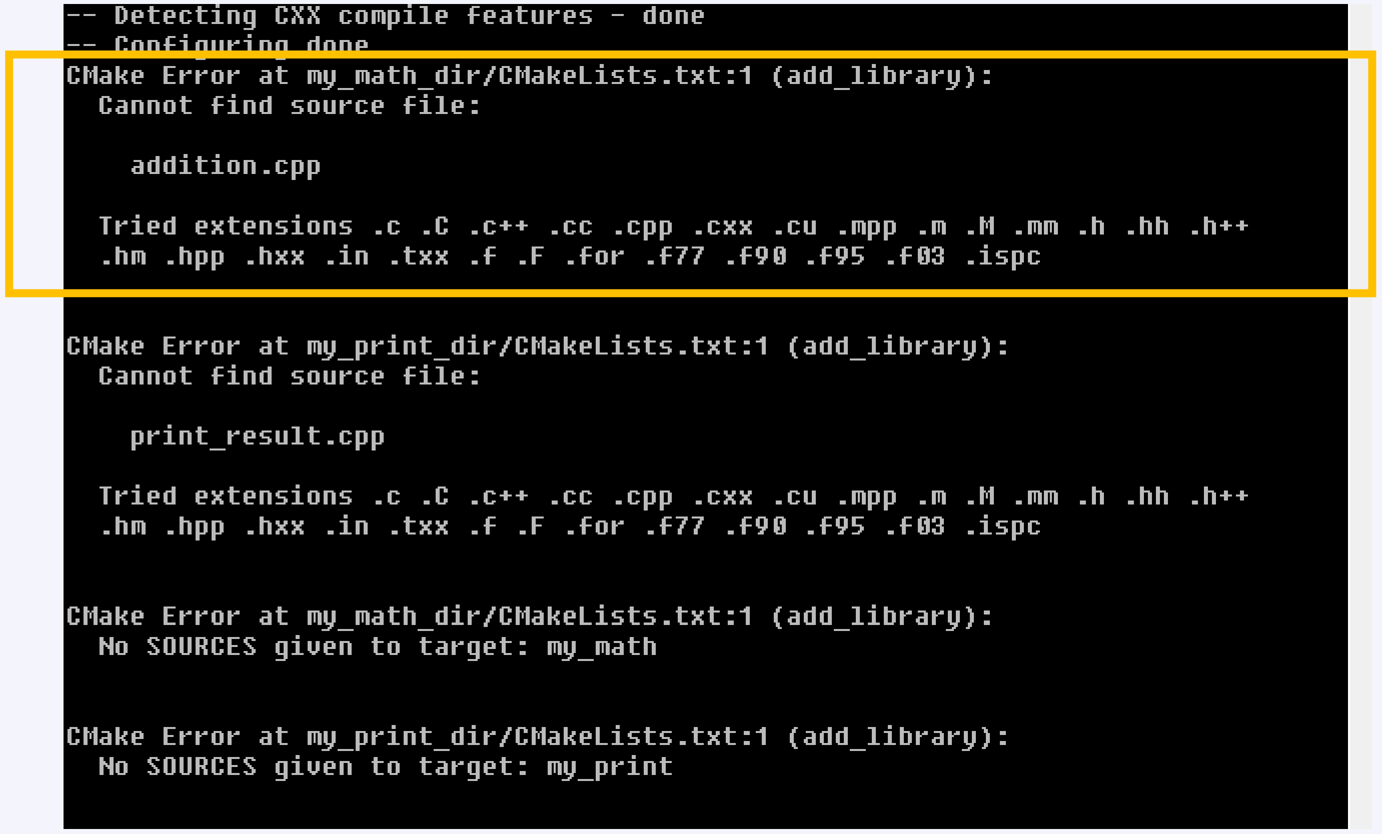
- If header file is in another folder – we need to inform preprocessor where (define the dir)



VLAAMS SUPERCOMPUTER CENTRUM

Source files

- If header or source file is in another folder – we need to inform preprocessor where (**define the dir**)



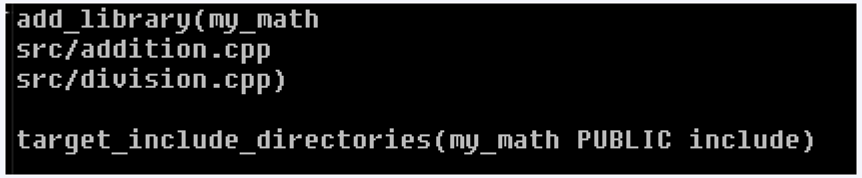
```
-- Detecting CXX compile features - done
-- Configuring done
CMake Error at my_math_dir/CMakeLists.txt:1 (add_library):
  Cannot find source file:

    addition.cpp

  Tried extensions .c .C .c++ .cc .cpp .cxx .cu .mpp .m .M .mm .h .hh .h++
  .hm .hpp .hxx .in .txx .f .F .for .f77 .f90 .f95 .f03 .ispc


CMake Error at my_print_dir/CMakeLists.txt:1 (add_library):
  Cannot find source file:

    print_result.cpp

  Tried extensions .c .C .c++ .cc .cpp .cxx .cu .mpp .m .M .mm .h .hh .h++
  .hm .hpp .hxx .in .txx .f .F .for .f77 .f90 .f95 .f03 .ispc


CMake Error at my_math_dir/CMakeLists.txt:1 (add_library):
  No SOURCES given to target: my_math


CMake Error at my_print_dir/CMakeLists.txt:1 (add_library):
  No SOURCES given to target: my_print
```

example7

VLAAMS SUPERCOMPUTER CENTRUM

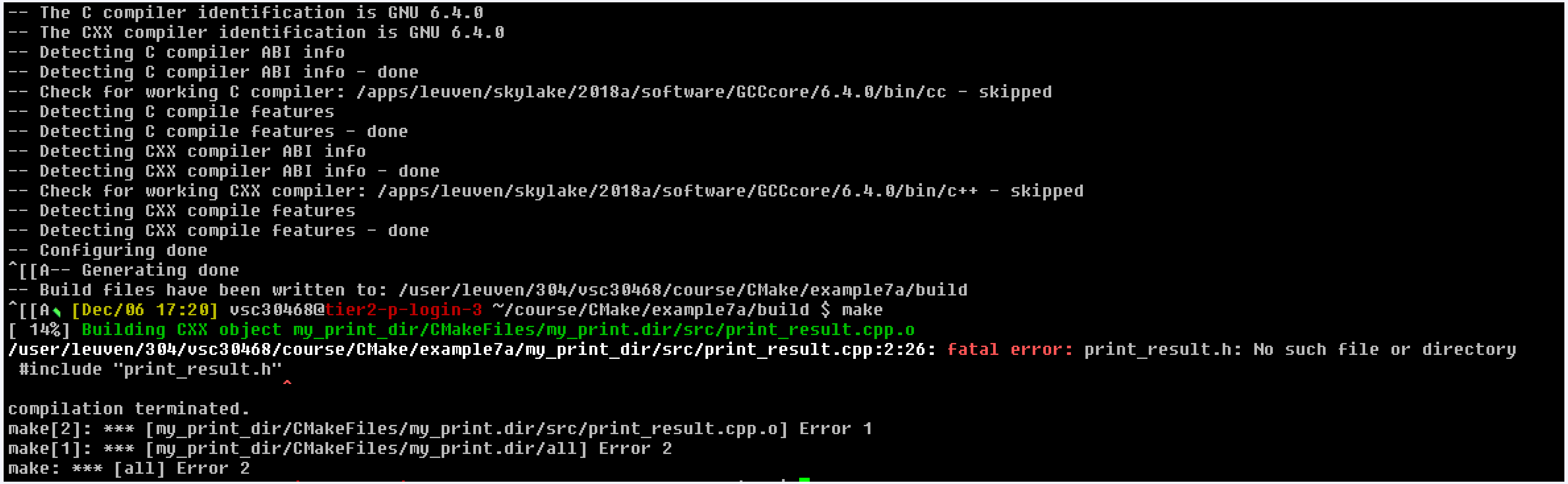
Header files

CMakeLists.txt



```
add_library(my_math
src/addition.cpp
src/division.cpp)

target_include_directories(my_math PUBLIC include)
```

example7a

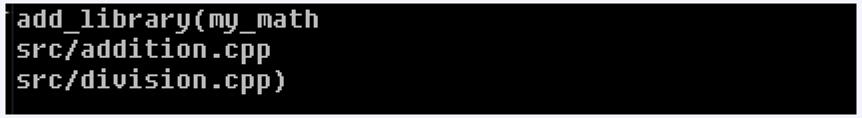


```
-- The C compiler identification is GNU 6.4.0
-- The CXX compiler identification is GNU 6.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
^[[A-- Generating done
-- Build files have been written to: /user/leuven/304/vsc30468/course/CMake/example7a/build
^[[A [Dec/06 17:20] vsc30468@tier2-p-login-3 ~/course/CMake/example7a/build $ make
[ 14%] Building CXX object my_print_dir/CMakeFiles/my_print.dir/src/print_result.cpp.o
/user/leuven/304/vsc30468/course/CMake/example7a/my_print_dir/src/print_result.cpp:2:26: fatal error: print_result.h: No such file or directory
 #include "print_result.h"
                          ^
compilation terminated.
make[2]: *** [my_print_dir/CMakeFiles/my_print.dir/src/print_result.cpp.o] Error 1
make[1]: *** [my_print_dir/CMakeFiles/my_print.dir/all] Error 2
make: *** [all] Error 2
```

VLAAMS
SUPERCOMPUTER
CENTRUM

Header files

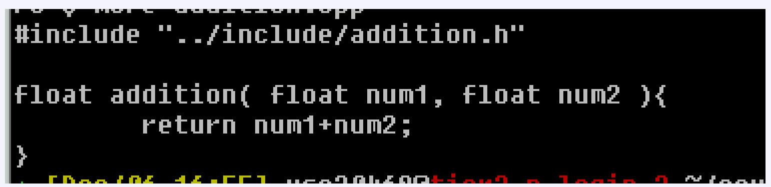
CMakeLists.txt



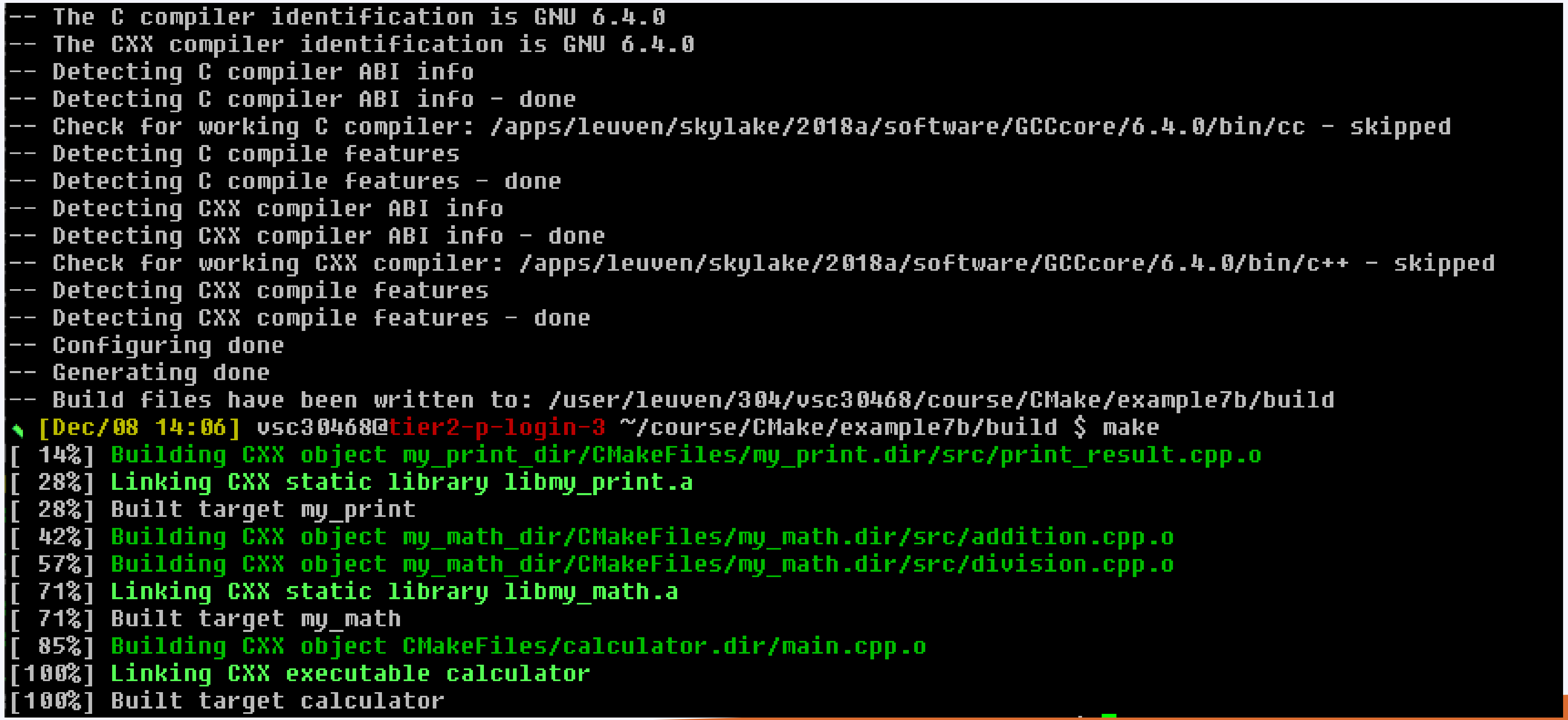
```
add_library(my_math
src/addition.cpp
src/division.cpp)
```

example7b

addition.cpp



```
#include "../include/addition.h"

float addition( float num1, float num2 ){
        return num1+num2;
}
```



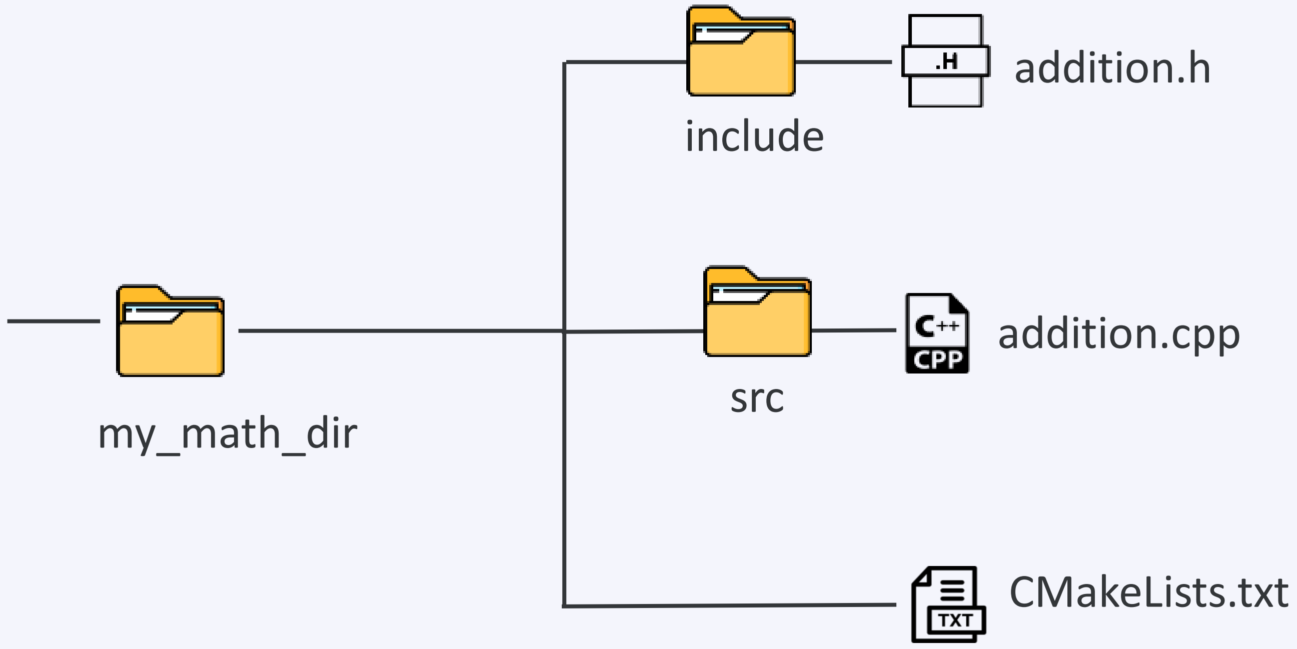
```
-- The C compiler identification is GNU 6.4.0
-- The CXX compiler identification is GNU 6.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /user/leuven/304/vsc30468/course/CMake/example7b/build
 [Dec/08 14:06] vsc30468@tier2-p-login-3 ~/course/CMake/example7b/build $ make
[ 14%] Building CXX object my_print_dir/CMakeFiles/my_print.dir/src/print_result.cpp.o
[ 28%] Linking CXX static library libmy_print.a
[ 28%] Built target my_print
[ 42%] Building CXX object my_math_dir/CMakeFiles/my_math.dir/src/addition.cpp.o
[ 57%] Building CXX object my_math_dir/CMakeFiles/my_math.dir/src/division.cpp.o
[ 71%] Linking CXX static library libmy_math.a
[ 71%] Built target my_math
[ 85%] Building CXX object CMakeFiles/calculator.dir/main.cpp.o
[100%] Linking CXX executable calculator
[100%] Built target calculator
```

VLAAMS SUPERCOMPUTER CENTRUM

Header files

- Ask CMake to take care of it

```
target_include_directories(<target>
<scope>
<dir1>
<dir2>
…
)
```



VLAAMS SUPERCOMPUTER CENTRUM

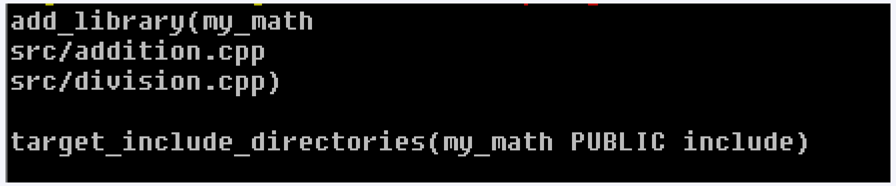
Header files

- Ask CMake to take care of it

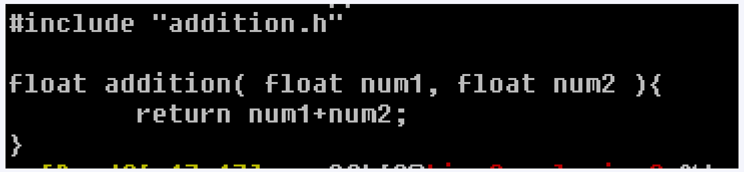
CMakeLists.txt

example7c

addition.cpp



```
add_library(my_math
src/addition.cpp
src/division.cpp)

target_include_directories(my_math PUBLIC include)
```

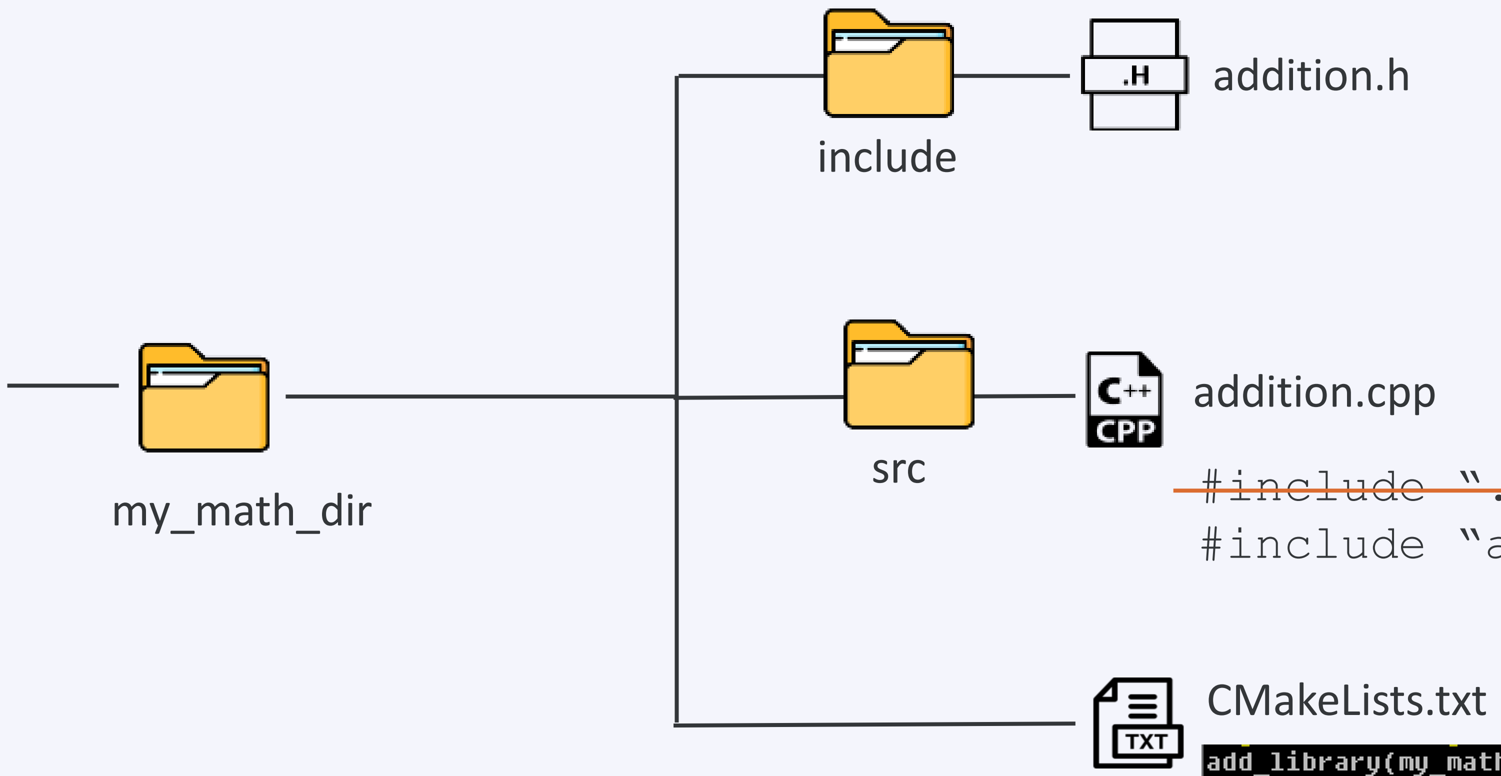


```
#include "addition.h"

float addition( float num1, float num2 ){
        return num1+num2;
}
```

VLAAMS SUPERCOMPUTER CENTRUM

Header files

- We can include multiple dirs at the same time



```
~~#include "../include/addition.h"~~
#include "addition.h"
```

```
add_library(my_math
src/addition.cpp
src/division.cpp)

target_include_directories(my_math PUBLIC include)
```

VLAAMS SUPERCOMPUTER CENTRUM

Header files

- In main.cpp we do not need any path for include header files – the **PUBLIC** keyword:

- PUBLIC: All the directories following PUBLIC will be used for the current target and the other targets that have dependencies on the current target, i.e., appending the directories to INCLUDE_DIRECTORIES and INTERFACE_INCLUDE_DIRECTORIES.

Target properties

- target_include_directories(my_print PUBLIC include)
- target_include_directories(my_print INTERFACE include)

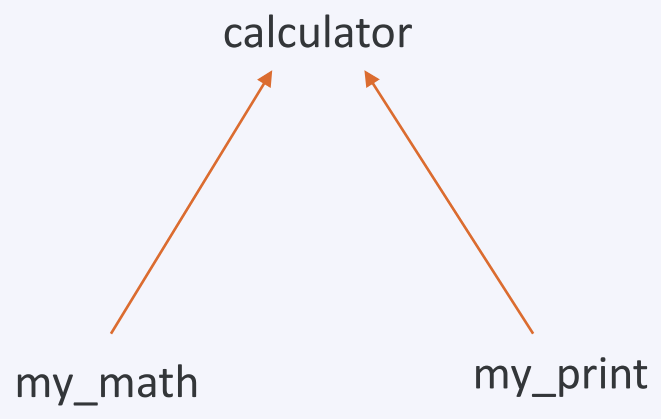
- Property: `INTERFACE_INCLUDE_DIRECTORIES` of target `my_print`

Is set to `$VSC_HOME/courses/Cmake/exercise7c/my_print/include`

VLAAMS SUPERCOMPUTER CENTRUM

Target properties

- Calculator dependencies
- target_link_libraries(calculator my_math my_print)



Property propagation

- Properties are visible to main.cpp
- **property propagation**

Properties of my_math	**Properties of my_print**
Property A=aaa	Property X=xxx
Property B=bbb	Property Y=yyy
Property C=ccc	Property Z=zzz

Property propagation

- PUBLIC, PRIVATE vs INTERFACE
- **Private** – does not set the interface of include dir or any other property, main.cpp will not be able to find them

- target_include_directories(my_print PRIVATE include)

- Property: `INTERFACE_INCLUDE_DIRECTORIES` of target `my_print`
  Is set to: `<not-set>`

Property propagation

- PUBLIC, INTERFACE and PRIVATE

Question:	Answer	Answer	Answer
Does ‘my_math’ need the directory	Yes	No	Yes
Do the other targets, depending upon ‘my_math’ are going to need this include directory?	Yes	Yes	No
	PUBLIC	INTERFACE	PRIVATE

Property propagation

- Other commands that will use it:

Command	Property set by the command
target_compile_definitions	INTERFACE_COMPILE_DEFINITIONS
target_sources	INTERFACE_SOURCES
target_compile_features	INTERFACE_COMPILE_FEATURES
target_compile_options	INTERFACE_COMPILE_OPTIONS
target_link_directories	INTERFACE_LINK_DIRECTORIES
target_link_libraries	INTERFACE_LINK_LIBRARIES
target_link_options	INTERFACE_LINK_OPTIONS
target_precompile_headers	INTERFACE_PRECOMPILE_HEADERS

Scopes

- Scope specifier allows to propagate requirements to another top-level target

```
    target_include_directories(my_print PRIVATE include)
target_include_directories(Target Name | Scope | Arguments)
```

VLAAMS SUPERCOMPUTER CENTRUM

Property propagation

- Target with both private and public

```
target_include_directories(target PRIVATE xxx PUBLIC yyy)
```

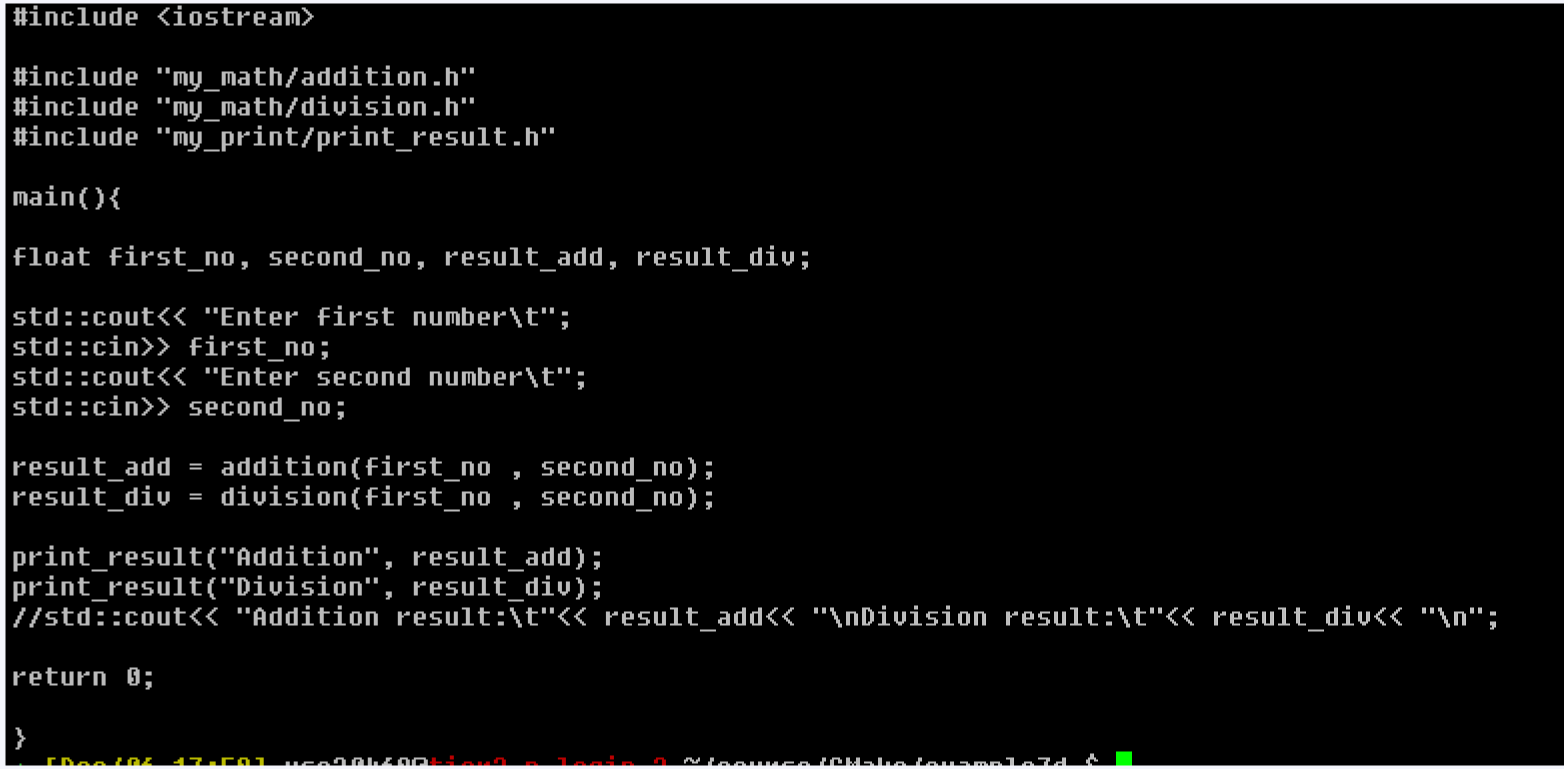
or

```
target_include_directories(target PRIVATE xxx)
target_include_directories(target PUBLIC yyy)
```

Property propagation

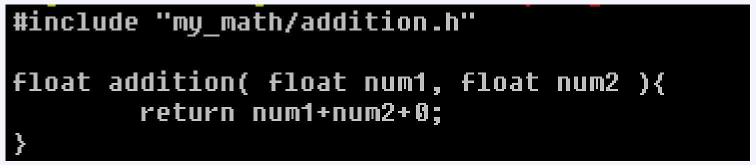
- How to know to which code lib belongs?

main.cpp



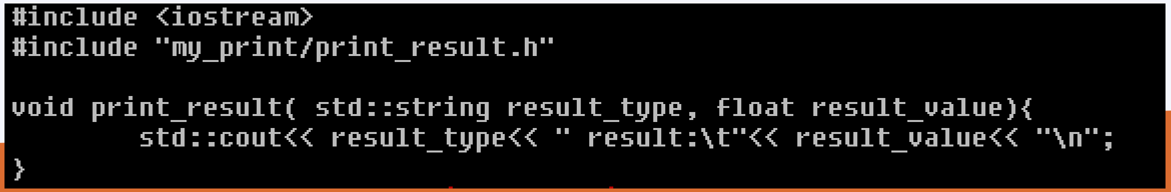
```
#include <iostream>

#include "my_math/addition.h"
#include "my_math/division.h"
#include "my_print/print_result.h"

main(){

float first_no, second_no, result_add, result_div;

std::cout<< "Enter first number\t";
std::cin>> first_no;
std::cout<< "Enter second number\t";
std::cin>> second_no;

result_add = addition(first_no , second_no);
result_div = division(first_no , second_no);

print_result("Addition", result_add);
print_result("Division", result_div);
//std::cout<< "Addition result:\t"<< result_add<< "\nDivision result:\t"<< result_div<< "\n";

return 0;

}
```

addition.cpp



```
#include "my_math/addition.h"

float addition( float num1, float num2 ){
        return num1+num2+0;
}
```

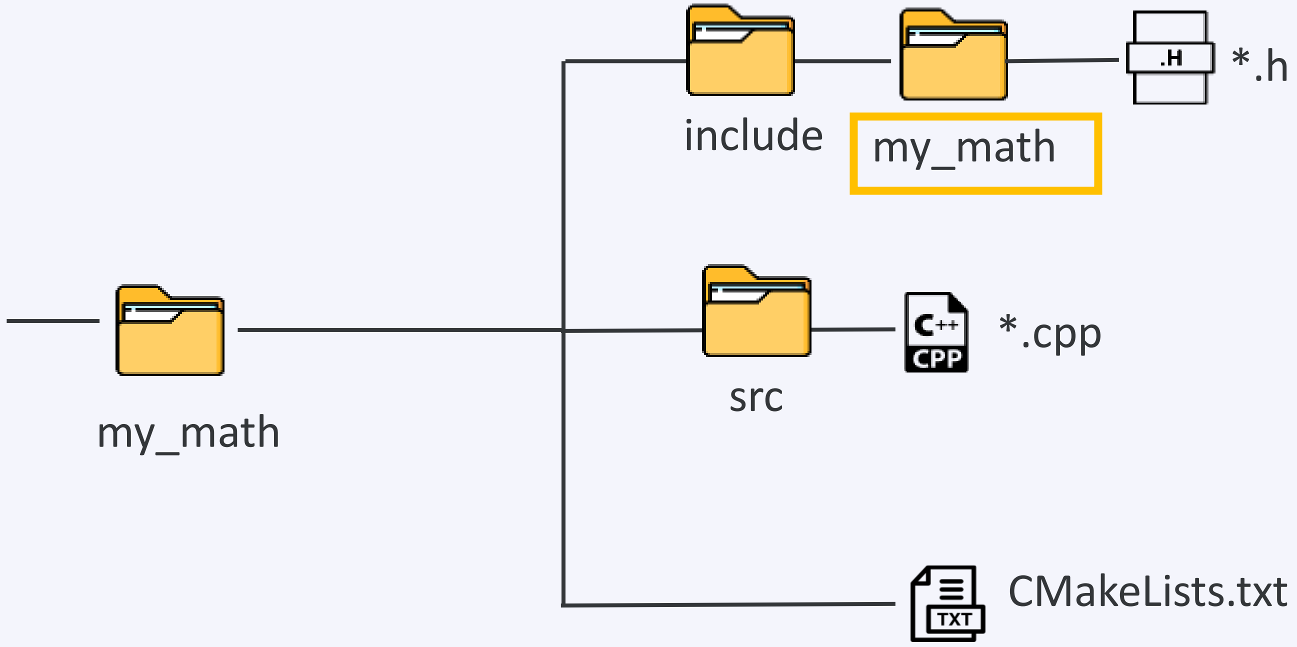
print_result.cpp



```
#include <iostream>
#include "my_print/print_result.h"

void print_result( std::string result_type, float result_value){
        std::cout<< result_type<< " result:\t"<< result_value<< "\n";
}
```

VLAAMS SUPERCOMPUTER CENTRUM

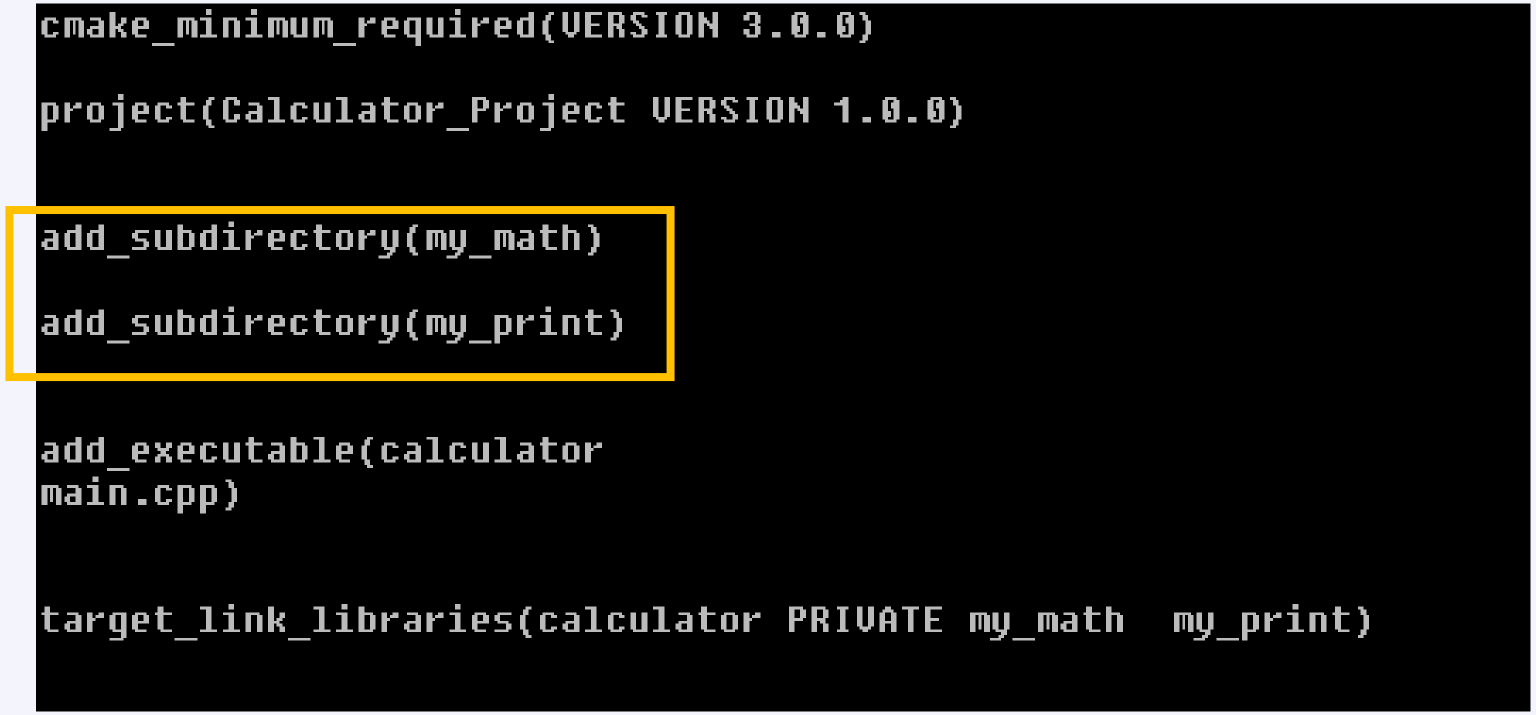
Property propagation



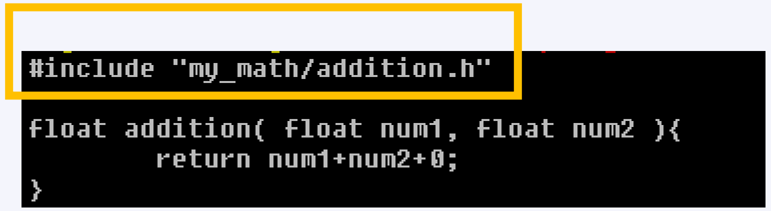
example7d

Requirements to run CMake

- Upper-level vs lower-level dir



```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)


add_subdirectory(my_math)

add_subdirectory(my_print)


add_executable(calculator
main.cpp)


target_link_libraries(calculator PRIVATE my_math  my_print)
```



```
#include "my_math/addition.h"

float addition( float num1, float num2 ){
        return num1+num2+0;
}
```

VLAAMS
SUPERCOMPUTER
CENTRUM

Message

- Message records the specified message text in the log. If more than one message string is given, they are concatenated into a single message with no separator between the strings.

```
message(<mode-of-display> “the message”)
```

Message

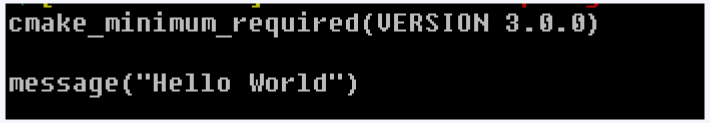
- Modes of message display

- `message(“Hello world”)`
- `message(STATUS “Hello world”)`
- `message(DEBUG “Hello world”)`
- `message(WARNING “Hello world”)`
- `message(FATAL ERROR “Hello world”)`

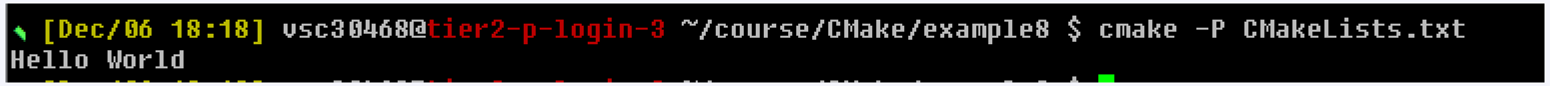
CMake scripting

- Scripts run commands and do not build any target
- `$ cmake -P filename` (any name)
- The message will be printed

example8

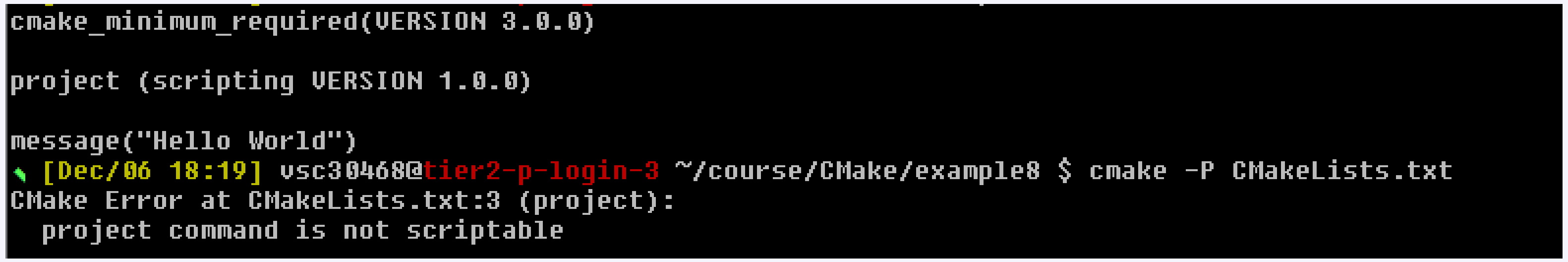


```
cmake_minimum_required(VERSION 3.0.0)

message("Hello World")
```



```
[Dec/06 18:18] vsc30468@tier2-p-login-3 ~/course/CMake/example8 $ cmake -P CMakeLists.txt
Hello World
```

- Project will give us error as it is not descriptible



```
cmake_minimum_required(VERSION 3.0.0)

project (scripting VERSION 1.0.0)

message("Hello World")
[Dec/06 18:19] vsc30468@tier2-p-login-3 ~/course/CMake/example8 $ cmake -P CMakeLists.txt
CMake Error at CMakeLists.txt:3 (project):
  project command is not scriptable
```

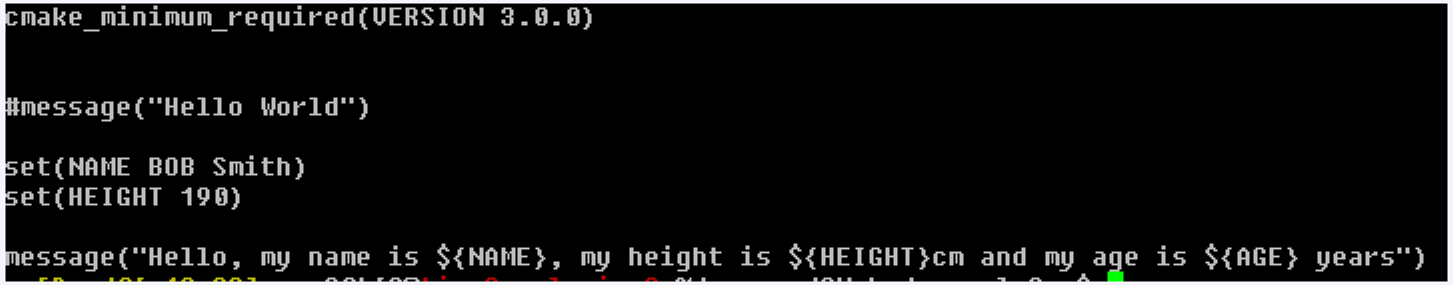
VLAAMS SUPERCOMPUTER CENTRUM

CMake scripting

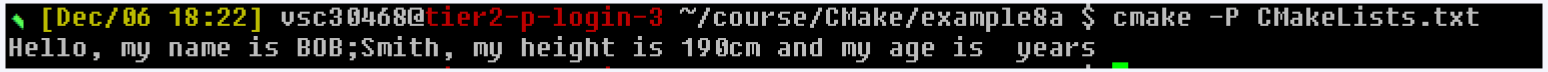
- All variables in CMake are string type
- Referencing to not set variable – will give empty string

- CMake command:
  `set(<variable_name> <variable_value>)`

- Variable dereferencing:
  `${variable_name}`



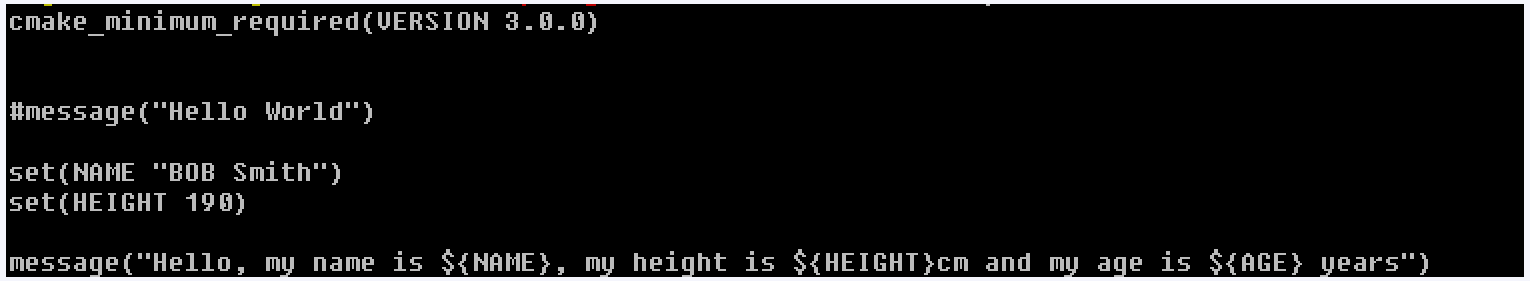
```
cmake_minimum_required(VERSION 3.0.0)


#message("Hello World")

set(NAME BOB Smith)
set(HEIGHT 190)

message("Hello, my name is ${NAME}, my height is ${HEIGHT}cm and my age is ${AGE} years")
```



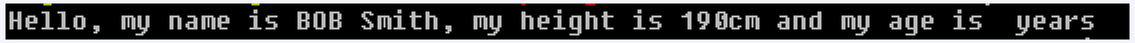
```
[Dec/06 18:22] vsc30468@tier2-p-login-3 ~/course/CMake/example8a $ cmake -P CMakeLists.txt
Hello, my name is BOB;Smith, my height is 190cm and my age is  years
```

CMake strings/lists

- Without “”
- A list in CMake is items with ; separation
- String is a list with single item

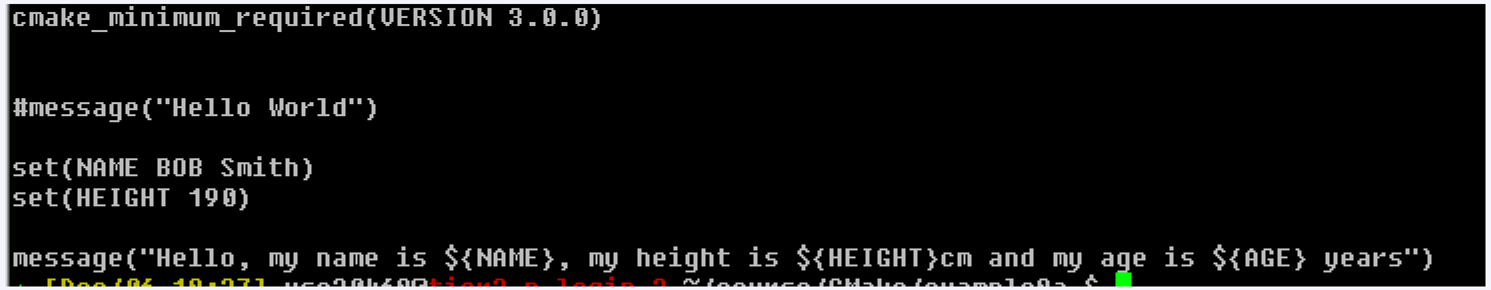


```
cmake_minimum_required(VERSION 3.0.0)


#message("Hello World")

set(NAME "BOB Smith")
set(HEIGHT 190)

message("Hello, my name is ${NAME}, my height is ${HEIGHT}cm and my age is ${AGE} years")
```

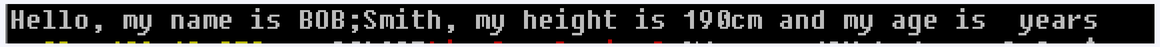


```
Hello, my name is BOB Smith, my height is 190cm and my age is  years
```

Set(Name “BOB Smith”) -> String ‘Name’=BOB Smith



```
cmake_minimum_required(VERSION 3.0.0)


#message("Hello World")

set(NAME BOB Smith)
set(HEIGHT 190)

message("Hello, my name is ${NAME}, my height is ${HEIGHT}cm and my age is ${AGE} years")
```



```
Hello, my name is BOB;Smith, my height is 190cm and my age is  years
```

Set(Name BOB Smith) -> List ‘Name’=BOB;Smith

String/List

- CMake commands:

- `list()` – List operations. A list in CMake is a ; separated group of strings. To create a list the set command can be used. For example, set(var a b c d e) creates a list with a;b;c;d;e, and set(var "a b c d e") creates a string or a list with one item in it.

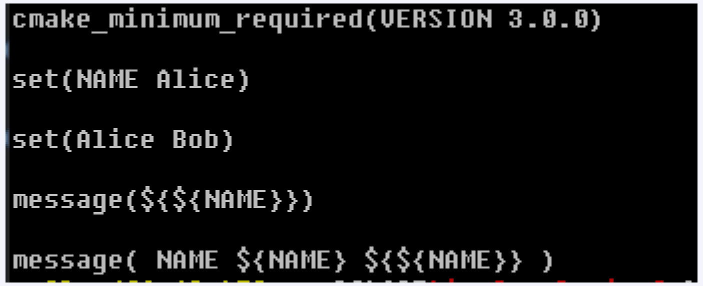
- `string()` – String operations.

String/List

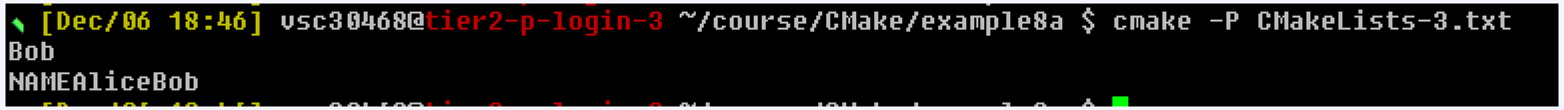
Set command	Value of VAR	message(${VAR})	message(“${VAR}”)
set(VAR aa bb cc)	aa;bb;cc	aabbcc	aa;bb;cc
set(VAR aa;bb;cc)	aa;bb;cc	aabbcc	aa;bb;cc
set(VAR “aa” “bb” “cc”)	aa;bb;cc	aabbcc	aa;bb;cc
set(VAR “aa bb cc”)	aa bb cc	aa bb cc	aa bb cc
set(VAR “aa;bb;cc”)	aa;bb;cc	aabbcc	aa;bb;cc

String/List

- CMake combined everything as 1 string



```
cmake_minimum_required(VERSION 3.0.0)

set(NAME Alice)

set(Alice Bob)

message(${${NAME}})

message( NAME ${NAME} ${${NAME}} )
```



```
[Dec/06 18:46] vsc30468@tier2-p-login-3 ~/course/CMake/example8a $ cmake -P CMakeLists-3.txt
Bob
NAMEAliceBob
```

VLAAMS SUPERCOMPUTER CENTRUM

String/List

Message command	Output
message(NAME ${NAME} ${${NAME}})	NAMEAliceBob
message(NAME${NAME}${${NAME}})	NAMEAliceBob
message(“NAME ${NAME} ${${NAME}}“)	NAME Alice Bob

VLAAMS SUPERCOMPUTER CENTRUM

Generator expressions

- Variable ${VAR}
- Generator Expressions: $<TARGET_FILE:library>

- Generator expressions are evaluated during build system generation to produce information specific to each build configuration. They have the form $<...>. For example:

```
target_include_directories(PRIVATE /opt/include/$<CXX_COMPILER_ID>)
```

This would expand to /opt/include/GNU etc. depending on the C++ compiler used.

- Generator expressions are allowed in the context of many target properties, such as LINK_LIBRARIES, INCLUDE_DIRECTORIES, COMPILE_DEFINITIONS and others. They may also be used when using commands to populate those properties, such as target_link_libraries(), target_include_directories(), target_compile_definitions() and others. They enable conditional linking, conditional definitions used when compiling, conditional include directories, and more.

Lists

- Indexing of list starts from 0
- Negative index – indexing done from the end, last index is -1
- List() command

  `list(<subcommand>…)`

  - `APPEND`
  - `INSERT`
  - `FILTER`
  - `GET`
  - `JOIN`

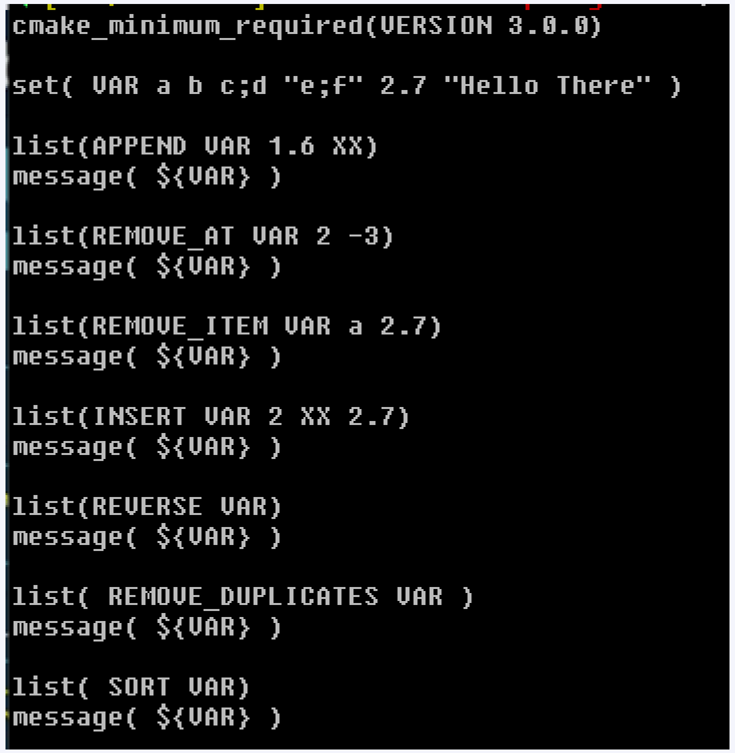
  https://cmake.org/cmake/help/latest/command/list.html?highlight=list

Lists

- Items to modify input list
  - `APPEND`
  - `REMOVE_ITEM`
  - `REMOVE_AT`
  - `INSERT`
  - `REVERSE`
  - `REMOVE_DUPLICATES`
  - `SORT`

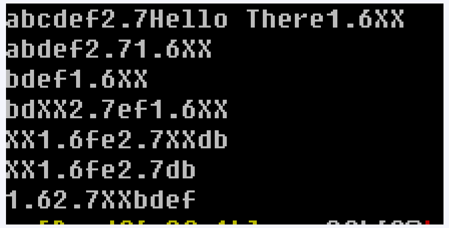
- Add to the list
- Remove 2 items at position 2 and -3
- Remove item 2.7 from the list
- Inset 2 items at position 2
- Reverse the list
- Remove the duplicates
- Sort the list

Lists



```
cmake_minimum_required(VERSION 3.0.0)

set( VAR a b c;d "e;f" 2.7 "Hello There" )

list(APPEND VAR 1.6 XX)
message( ${VAR} )

list(REMOVE_AT VAR 2 -3)
message( ${VAR} )

list(REMOVE_ITEM VAR a 2.7)
message( ${VAR} )

list(INSERT VAR 2 XX 2.7)
message( ${VAR} )

list(REVERSE VAR)
message( ${VAR} )

list( REMOVE_DUPLICATES VAR )
message( ${VAR} )

list( SORT VAR)
message( ${VAR} )
```

example9



```
abcdef2.7Hello There1.6XX
abdef2.71.6XX
bdef1.6XX
bdXX2.7ef1.6XX
XX1.6fe2.7XXdb
XX1.6fe2.7db
1.62.7XXbdef
```

- Add to the list
- Remove 2 items at position 2 and -3
- Remove item 2.7 from the list
- Inset 2 items at position 2
- Reverse the list
- Remove the duplicates
- Sort the list

VLAAMS SUPERCOMPUTER CENTRUM

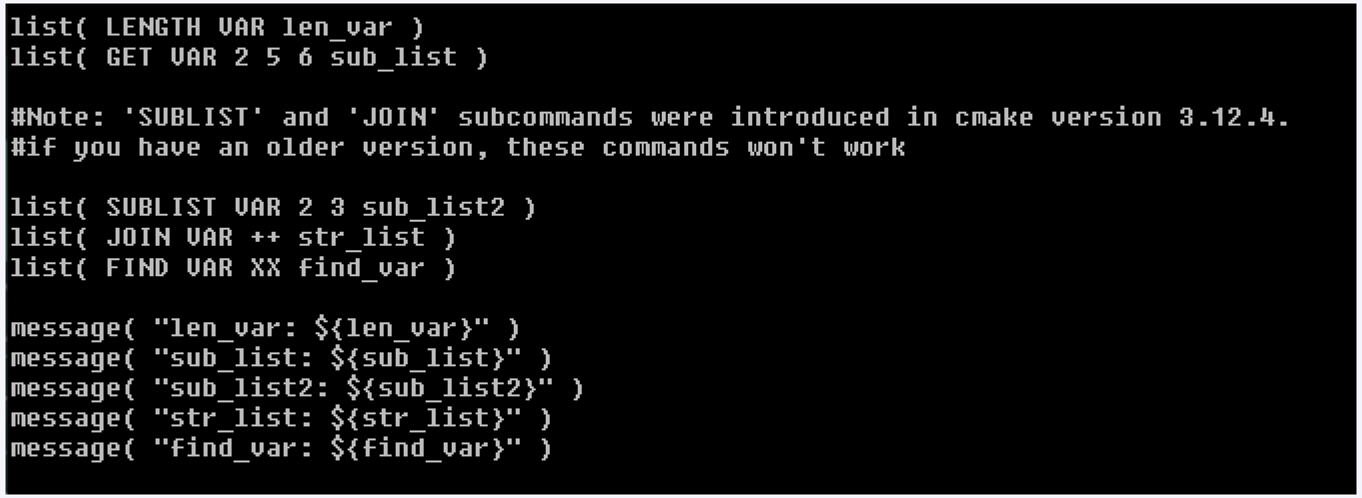
Lists

- set(VAR a b c;d "e;f" 2.7 "Hello There")

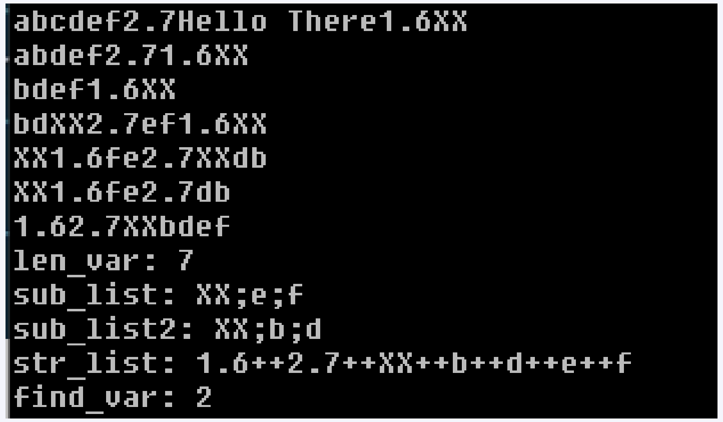
List commands	Output	Action
list(APPEND VAR 1.6 XX)	abcdef2.7Hello There1.6XX	Add to the list
list(REMOVE_AT VAR 2 -3)	abdef2.71.6XX	Remove 2 items at position 2 and -3
list(REMOVE_ITEM VAR a 2.7)	bdef1.6XX	Remove item 2.7 from the list
list(INSERT VAR 2 XX 2.7)	bdXX2.7ef1.6XX	Inset 2 items at position 2
list(REVERSE VAR)	XX1.6fe2.7XXdb	Reverse the list
list(REMOVE_DUPLICATES VAR)	XX1.6fe2.7db	Remove the duplicates
list(SORT VAR)	1.62.7XXbdef	Sort the list

Lists

```
set( VAR a b c;d "e;f" 2.7 "Hello There" )
```



```
list( LENGTH VAR len_var )
list( GET VAR 2 5 6 sub_list )

#Note: 'SUBLIST' and 'JOIN' subcommands were introduced in cmake version 3.12.4.
#if you have an older version, these commands won't work

list( SUBLIST VAR 2 3 sub_list2 )
list( JOIN VAR ++ str_list )
list( FIND VAR XX find_var )

message( "len_var: ${len_var}" )
message( "sub_list: ${sub_list}" )
message( "sub_list2: ${sub_list2}" )
message( "str_list: ${str_list}" )
message( "find_var: ${find_var}" )
```



```
abcdef2.7Hello There1.6XX
abdef2.71.6XX
bdef1.6XX
bdXX2.7ef1.6XX
XX1.6fe2.7XXdb
XX1.6fe2.7db
1.62.7XXbdef
len_var: 7
sub_list: XX;e;f
sub_list2: XX;b;d
str_list: 1.6++2.7++XX++b++d++e++f
find_var: 2
```

```
list(SUBLIST <list> <begin> <length> <output variable>)
```

New in version 3.12.

Returns a sublist of the given list. If <length> is 0, an empty list will be returned. If <length> is -1 or the list is smaller than <begin>+<length> then the remaining elements of the list starting at <begin> will be returned.

Lists

- Negative – if specified element not found

- Current list: 1.6;2.7;XX;b;d;e;f
- Index: 0 1 2 3 4 5 6

`list( LENGTH VAR len_var )`	len_var: 7
`list( GET VAR 2 5 6 sub_list )`	sub_list: XX;e;f
`list( SUBLIST VAR 2 3 sub_list2 )`	sub_list2: XX;b;d
`list( JOIN VAR ++ str_list )`	str_list: 1.6++2.7++XX++b++d++e++f
`list( FIND VAR XX find_var )`	find_var: 2

- `list(JOIN <list> <glue> <output variable>)`

New in version 3.12.

Returns a string joining all list's elements using the glue string. To join multiple strings, which are not part of a list, use JOIN operator from string() command.

Strings

- Commands for string
  - `FIND`
  - `REPLACE`
  - `PREPEND`
  - `APPEND`
  - `TOLOWER`
  - `TOUPPER`
  - `COMPARE`

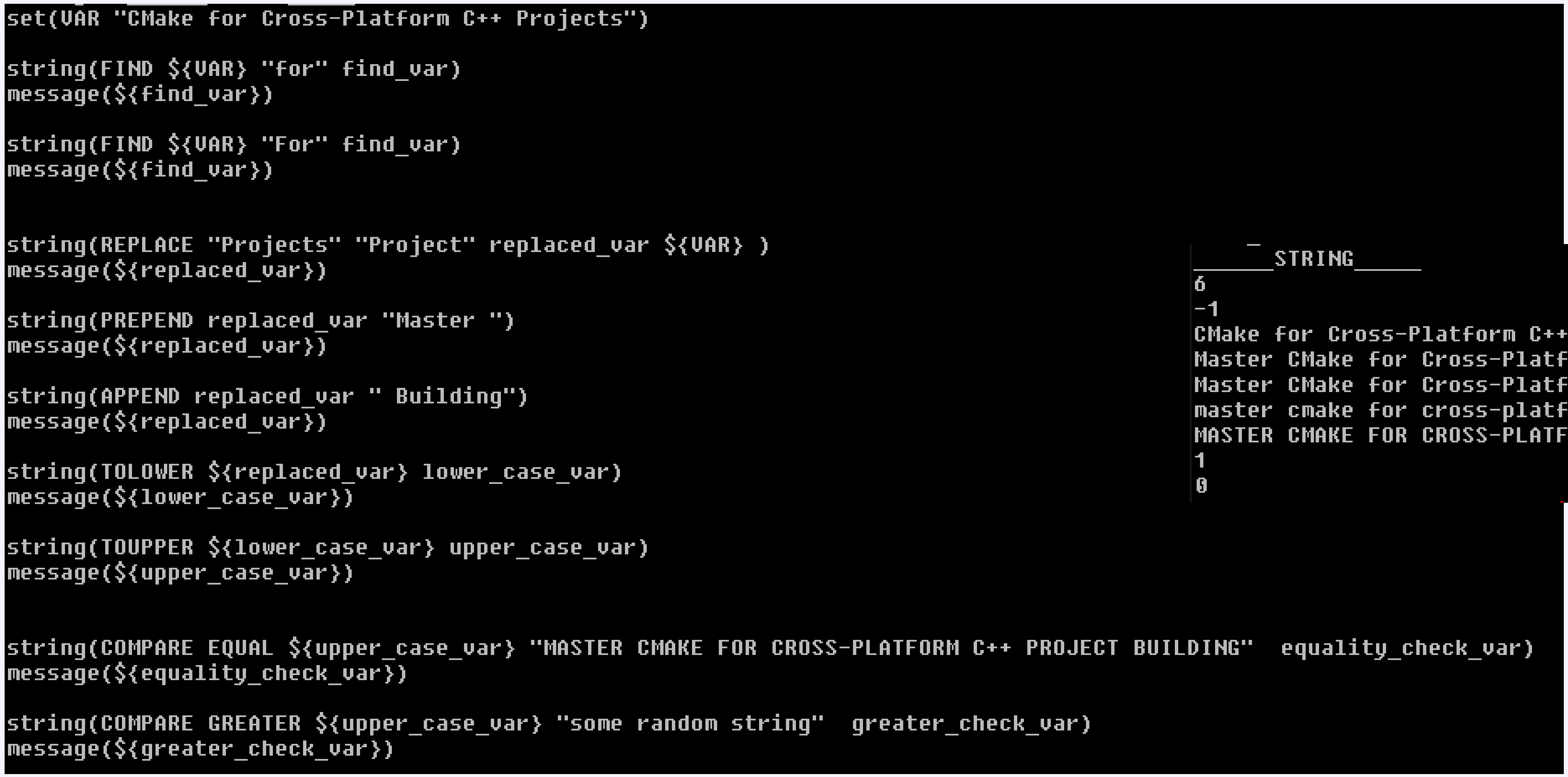
Strings

- String() command
- `set(VAR “CMake for Cross-Platform C++Projects”)`

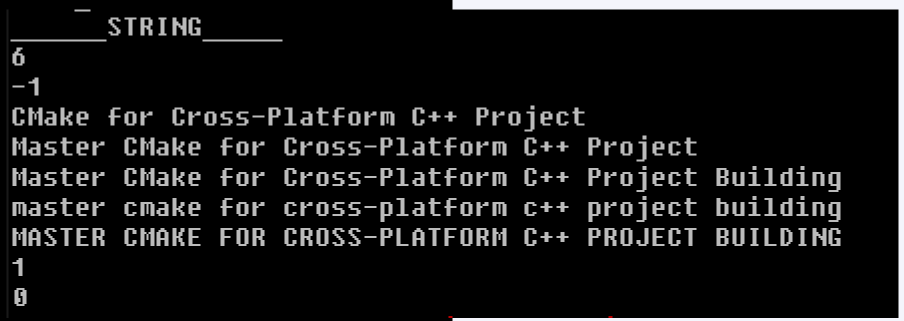
- `string(FIND ${VAR} “for” find_var)`
- `message(${find_var})` -> 6

- `string(REPLACE “Projects” “Project” replaced_var${VAR})`
- `message(${replaced_var})` -> CMake for Cross-Platform C++Project

Strings



```
set(VAR "CMake for Cross-Platform C++ Projects")

string(FIND ${VAR} "for" find_var)
message(${find_var})

string(FIND ${VAR} "For" find_var)
message(${find_var})


string(REPLACE "Projects" "Project" replaced_var ${VAR} )
message(${replaced_var})

string(PREPEND replaced_var "Master ")
message(${replaced_var})

string(APPEND replaced_var " Building")
message(${replaced_var})

string(TOLOWER ${replaced_var} lower_case_var)
message(${lower_case_var})

string(TOUPPER ${lower_case_var} upper_case_var)
message(${upper_case_var})


string(COMPARE EQUAL ${upper_case_var} "MASTER CMAKE FOR CROSS-PLATFORM C++ PROJECT BUILDING"  equality_check_var)
message(${equality_check_var})

string(COMPARE GREATER ${upper_case_var} "some random string"  greater_check_var)
message(${greater_check_var})
```



```
______STRING_____
6
-1
CMake for Cross-Platform C++ Project
Master CMake for Cross-Platform C++ Project
Master CMake for Cross-Platform C++ Project Building
master cmake for cross-platform c++ project building
MASTER CMAKE FOR CROSS-PLATFORM C++ PROJECT BUILDING
1
0
```

example9

VLAAMS SUPERCOMPUTER CENTRUM

File

- command `file()` command to operate on files
  - READ
  - WRITE
  - RENAME
  - REMOVE
  - COPY
  - DOWNLOAD
  - LOCK

Loops

Loopings:

- Flow control commands
  - If-else
  - Loop
    - while
    - foreach
- Function command
- Scopes
- Macro command
- Modules

Loops

```
if(<condition>)
        <command1>
        <command1>
        …
        …
endif()
```

VLAAMS
SUPERCOMPUTER
CENTRUM

Loops

```
if(<condition>)
        <commands>
elseif(<condition>)
        <commands>
else
        <commands>
endif()
```

Constants

- 1, ON, YES, TRUE, Y, a non-zero number: **TRUE**
- 0, OFF, NO, FALSE, N, IGNORE, NOTFOUND, the empty string, string ending with –NOTFOUND: **FALSE**

Constants with if

- Checking the condition

```
if(YES)
     <commands>
endif()
```

```
if(N)
     <commands>
endif()
```

```
if(ON)
     <commands>
endif()
```

```
if(OFF)
     <commands>
endif()
```

Variables with if

- Using variables with if()

Constant

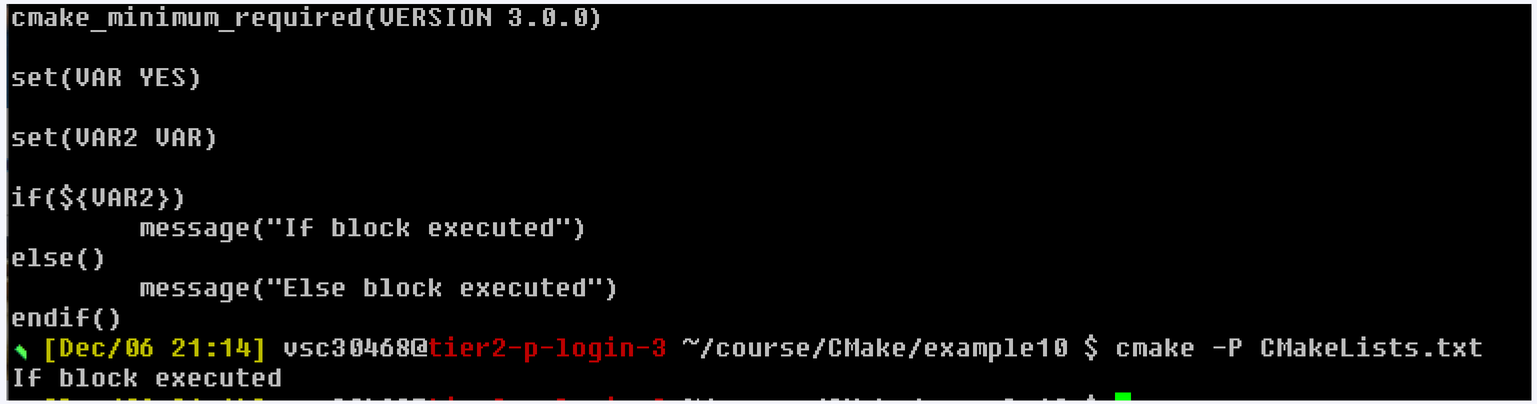
```
if(YES)
        <commands>
endif()
```

Variable

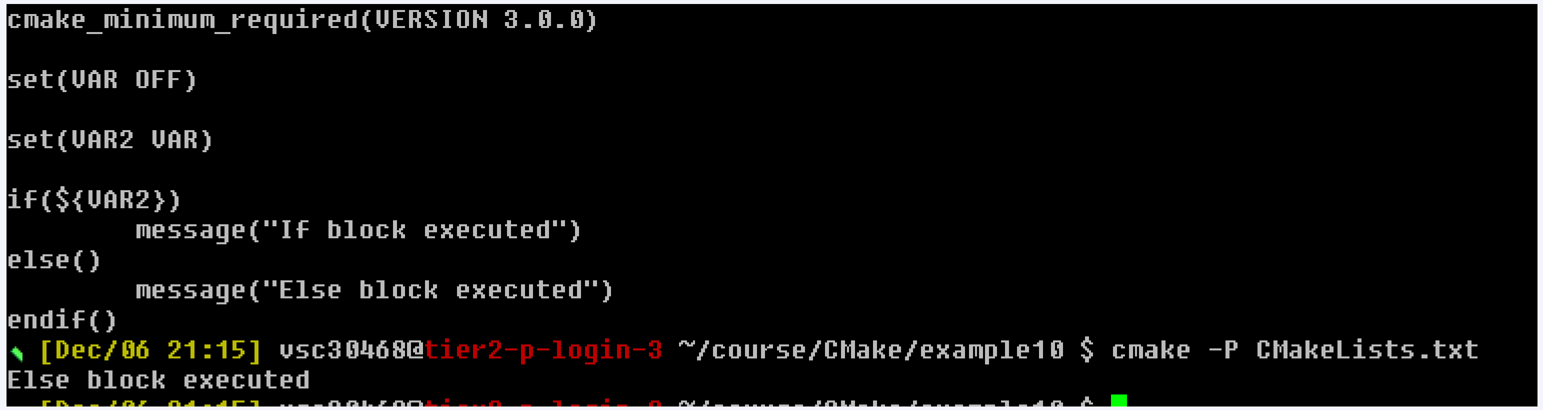
```
if(YE)
        <commands>
endif()
```

Tests

- If-else does dereferencing for us of `VAR`
- If VAR is empty string = > false condition



```
cmake_minimum_required(VERSION 3.0.0)

set(VAR YES)

set(VAR2 VAR)

if(${VAR2})
        message("If block executed")
else()
        message("Else block executed")
endif()
 [Dec/06 21:14] vsc30468@tier2-p-login-3 ~/course/CMake/example10 $ cmake -P CMakeLists.txt
If block executed
```



```
cmake_minimum_required(VERSION 3.0.0)

set(VAR OFF)

set(VAR2 VAR)

if(${VAR2})
        message("If block executed")
else()
        message("Else block executed")
endif()
 [Dec/06 21:15] vsc30468@tier2-p-login-3 ~/course/CMake/example10 $ cmake -P CMakeLists.txt
Else block executed
```

example10

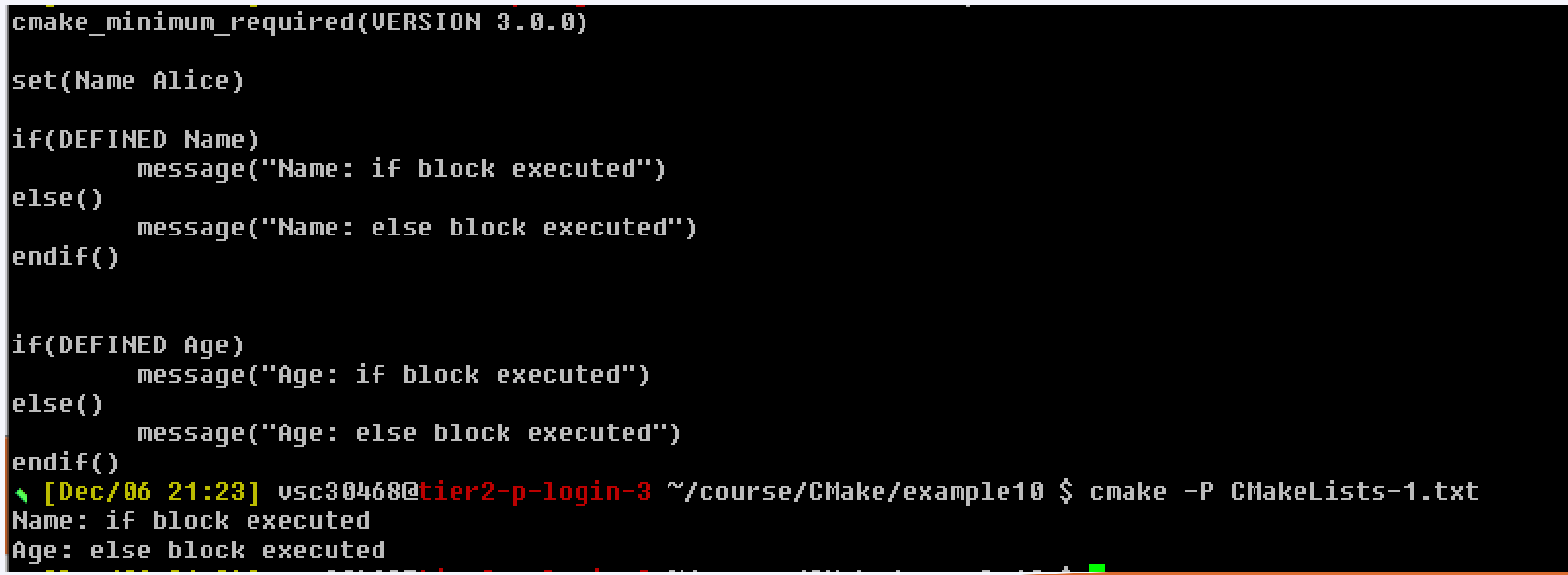
VLAAMS SUPERCOMPUTER CENTRUM

Unary tests

- Unary test– If something exists or not
- Defined – if variable set or not, command – if command exists or not and exists – if file or dir exists or not

- If() Conditions
  - Unary tests
  - Binary tests
  - Boolean operators

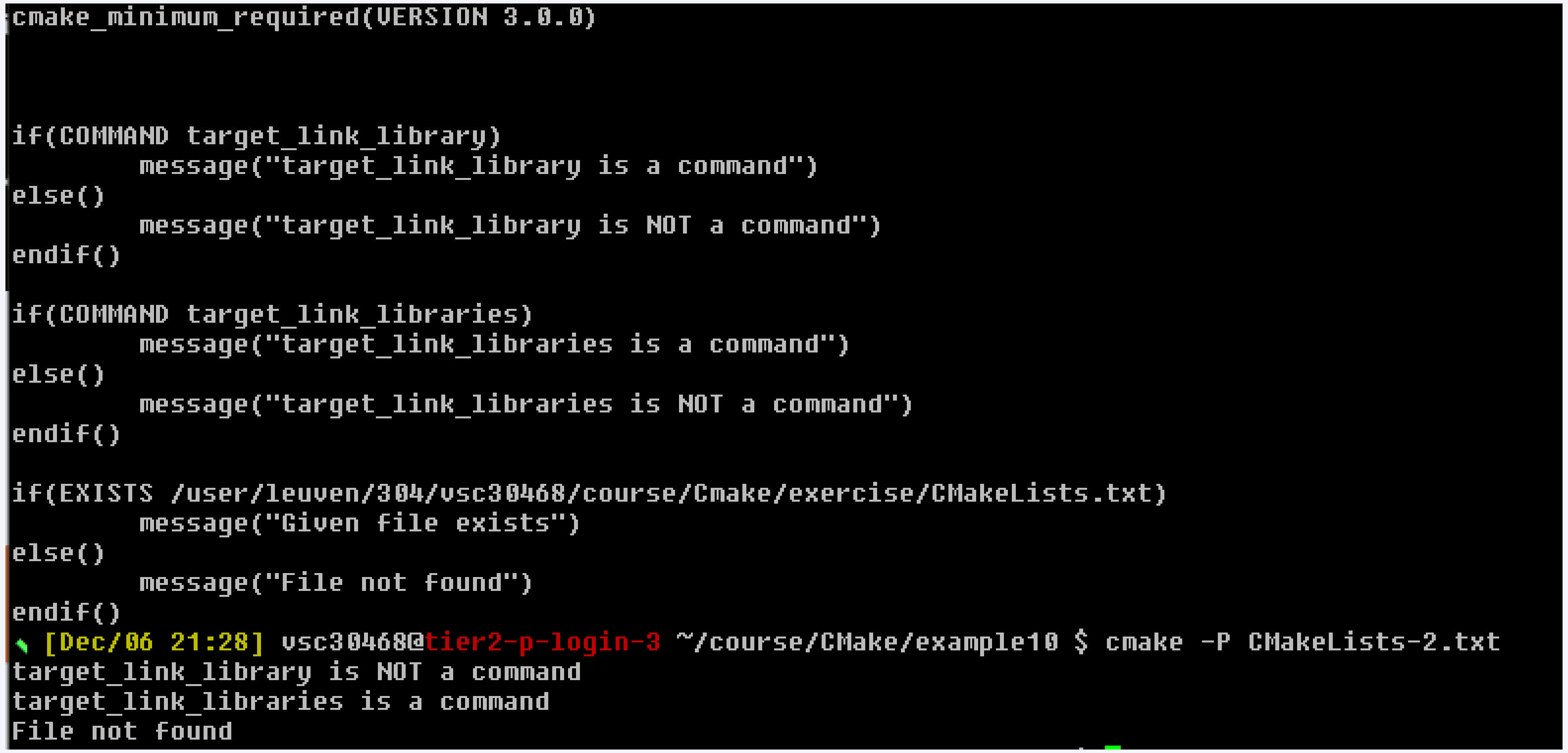
- If() Conditions | Unary tests
  - DEFINED
  - COMMAND
  - EXISTS



```
cmake_minimum_required(VERSION 3.0.0)

set(Name Alice)

if(DEFINED Name)
        message("Name: if block executed")
else()
        message("Name: else block executed")
endif()


if(DEFINED Age)
        message("Age: if block executed")
else()
        message("Age: else block executed")
endif()
 [Dec/06 21:23] vsc30468@tier2-p-login-3 ~/course/CMake/example10 $ cmake -P CMakeLists-1.txt
Name: if block executed
Age: else block executed
```

VLAAMS SUPERCOMPUTER CENTRUM

Unary tests

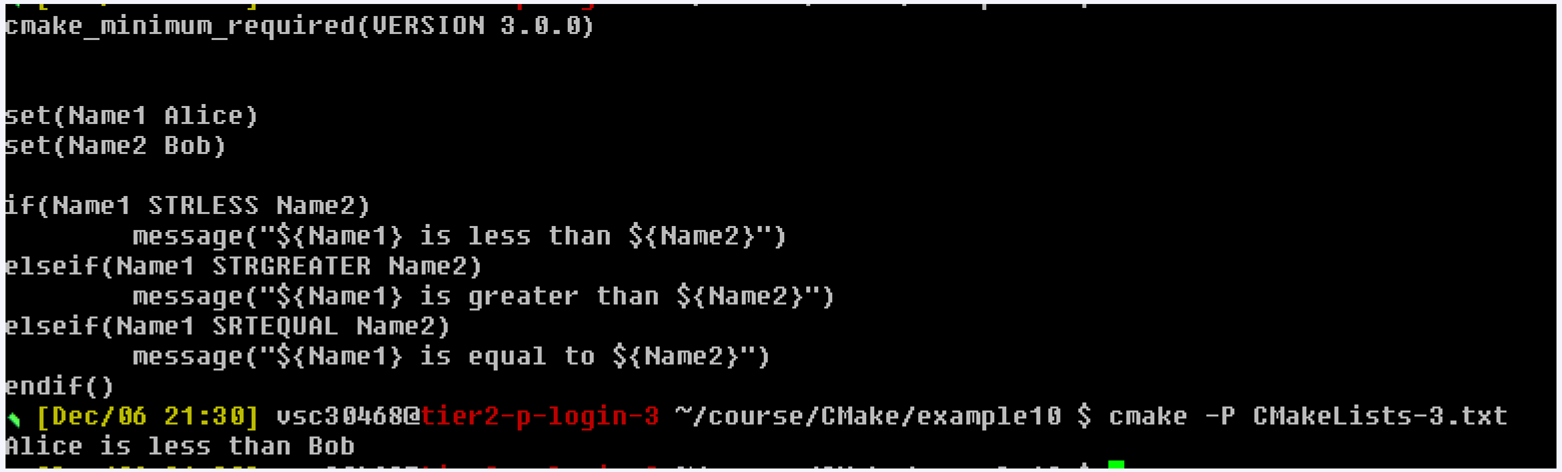


```
cmake_minimum_required(VERSION 3.0.0)



if(COMMAND target_link_library)
        message("target_link_library is a command")
else()
        message("target_link_library is NOT a command")
endif()

if(COMMAND target_link_libraries)
        message("target_link_libraries is a command")
else()
        message("target_link_libraries is NOT a command")
endif()

if(EXISTS /user/leuven/304/vsc30468/course/Cmake/exercise/CMakeLists.txt)
        message("Given file exists")
else()
        message("File not found")
endif()
 [Dec/06 21:28] vsc30468@tier2-p-login-3 ~/course/CMake/example10 $ cmake -P CMakeLists-2.txt
target_link_library is NOT a command
target_link_libraries is a command
File not found
```

VLAAMS
SUPERCOMPUTER
CENTRUM

Binary tests

- Binary tests – if 2 strings
- string or variable are = < >
- strings compared lexically (according to alphabet – upper case before lower case)



```
cmake_minimum_required(VERSION 3.0.0)


set(Name1 Alice)
set(Name2 Bob)

if(Name1 STRLESS Name2)
        message("${Name1} is less than ${Name2}")
elseif(Name1 STRGREATER Name2)
        message("${Name1} is greater than ${Name2}")
elseif(Name1 SRTEQUAL Name2)
        message("${Name1} is equal to ${Name2}")
endif()
 [Dec/06 21:30] vsc30468@tier2-p-login-3 ~/course/CMake/example10 $ cmake -P CMakeLists-3.txt
Alice is less than Bob
```

VLAAMS SUPERCOMPUTER CENTRUM

Boolean operators

- If () condition | Boolean operators

- `if(NOT DEFINED VAR)`
- `if(NOT(VAR STREQUAL “test” OR VAR2 STREQUAL “test2”))`
- `if(NOT(VAR STREQUAL “test” AND VAR2 STREQUAL “test2”))`

VLAAMS SUPERCOMPUTER CENTRUM

While loop

- Loops:
  - `while`
  - `foreach`

```
while(<condition>)
        <commands>
endwhile()
```

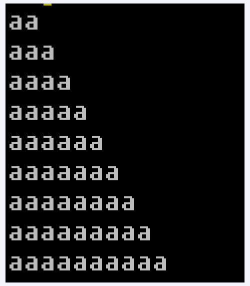
Foreach

- Dereference in the message not in the condition
- foreach – iterate over list of items or range of numbers

```
foreach(<loop_variable> <items>)
      <commands>
endforeach()
```

```
while(NOT VAR STREQUAL “aaaaaaaaaa”)
        set(VAR ${VAR}a)
        message(${VAR})
endwhile()
```

example11



```
aa
aaa
aaaa
aaaaa
aaaaaa
aaaaaaa
aaaaaaaa
aaaaaaaaa
aaaaaaaaaa
```

Foreach

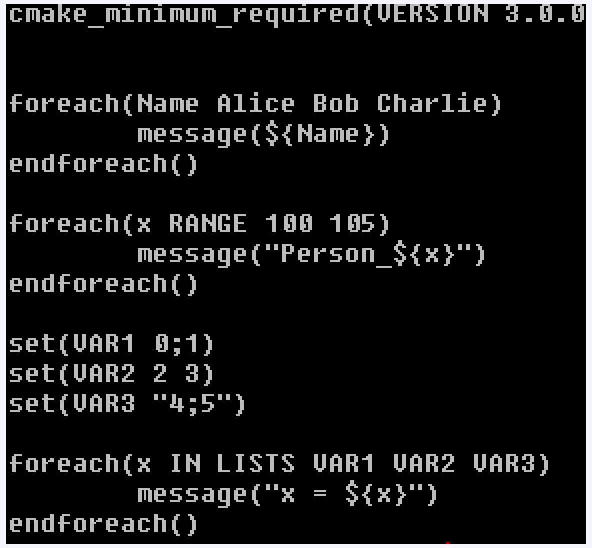
- Space or semicolon separated- output the same
- foreach() command
  - `foreach(Name Alice Bob Charlie)`
  - `foreach(Name Alice;Bob;Charlie)`

Foreach

- X in
  - 0-10: `foreach(x RANGE 10)`
  - 10-20: `foreach(x RANGE 10 20)`
  - 10-20 steps of 3: `foreach(x RANGE 10 20 3)`

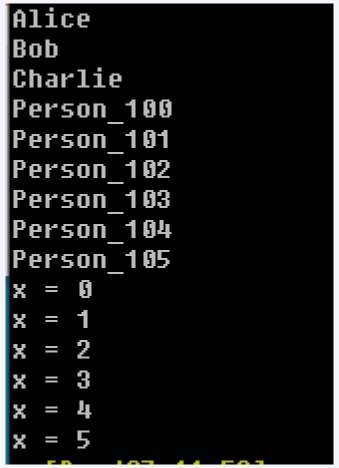
Foreach

- foreach() command
  - `foreach(x IN LISTS <list1> <list2> <list3>)`
  - `foreach(Name Alice;Bob;Charlie)`



```
cmake_minimum_required(VERSION 3.0.0

foreach(Name Alice Bob Charlie)
        message(${Name})
endforeach()

foreach(x RANGE 100 105)
        message("Person_${x}")
endforeach()

set(VAR1 0;1)
set(VAR2 2 3)
set(VAR3 "4;5")

foreach(x IN LISTS VAR1 VAR2 VAR3)
        message("x = ${x}")
endforeach()
```

example11



```
Alice
Bob
Charlie
Person_100
Person_101
Person_102
Person_103
Person_104
Person_105
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
```

With ; loop executed 6 times

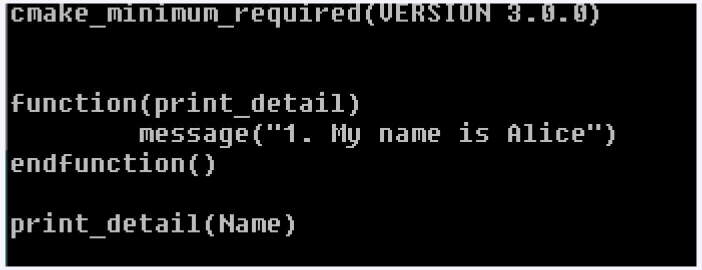
VLAAMS SUPERCOMPUTER CENTRUM

Function

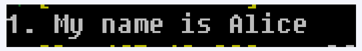
```
function(<function_name> <function_agrs>)
        <commands>
endfuction()
```

Function

example12



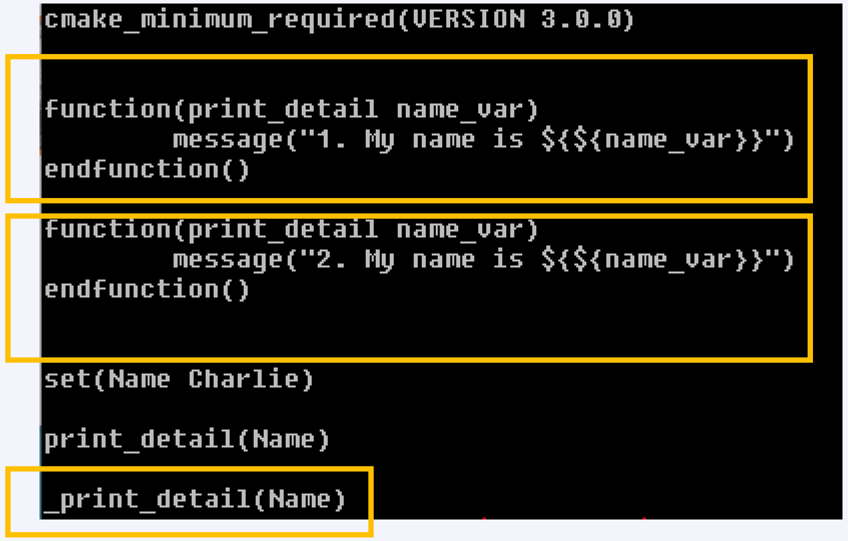
```
cmake_minimum_required(VERSION 3.0.0)


function(print_detail)
        message("1. My name is Alice")
endfunction()

print_detail(Name)
```



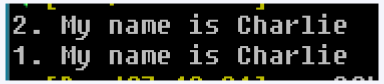
```
1. My name is Alice
```

Functions

- If we define twice - CMake renames original to _function-name
- Prepending twice _ does not work

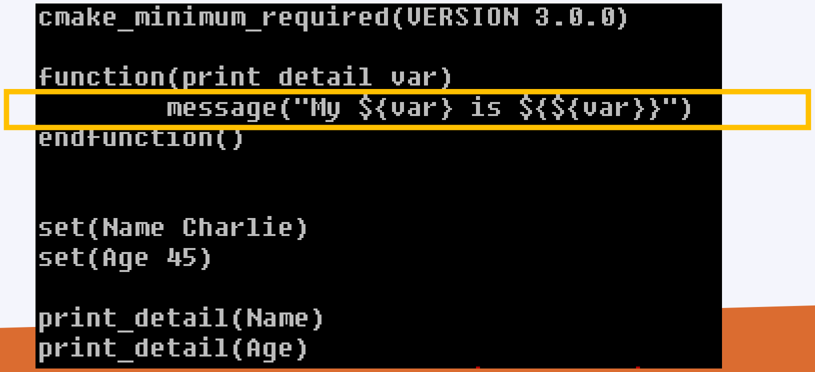


```
cmake_minimum_required(VERSION 3.0.0)

function(print_detail name_var)
        message("1. My name is ${${name_var}}")
endfunction()
function(print_detail name_var)
        message("2. My name is ${${name_var}}")
endfunction()

set(Name Charlie)

print_detail(Name)

_print_detail(Name)
```

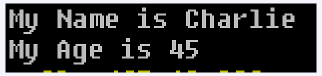


```
2. My name is Charlie
1. My name is Charlie
```

example12



```
cmake_minimum_required(VERSION 3.0.0)

function(print_detail var)
        message("My ${var} is ${${var}}")
endfunction()


set(Name Charlie)
set(Age 45)

print_detail(Name)
print_detail(Age)
```



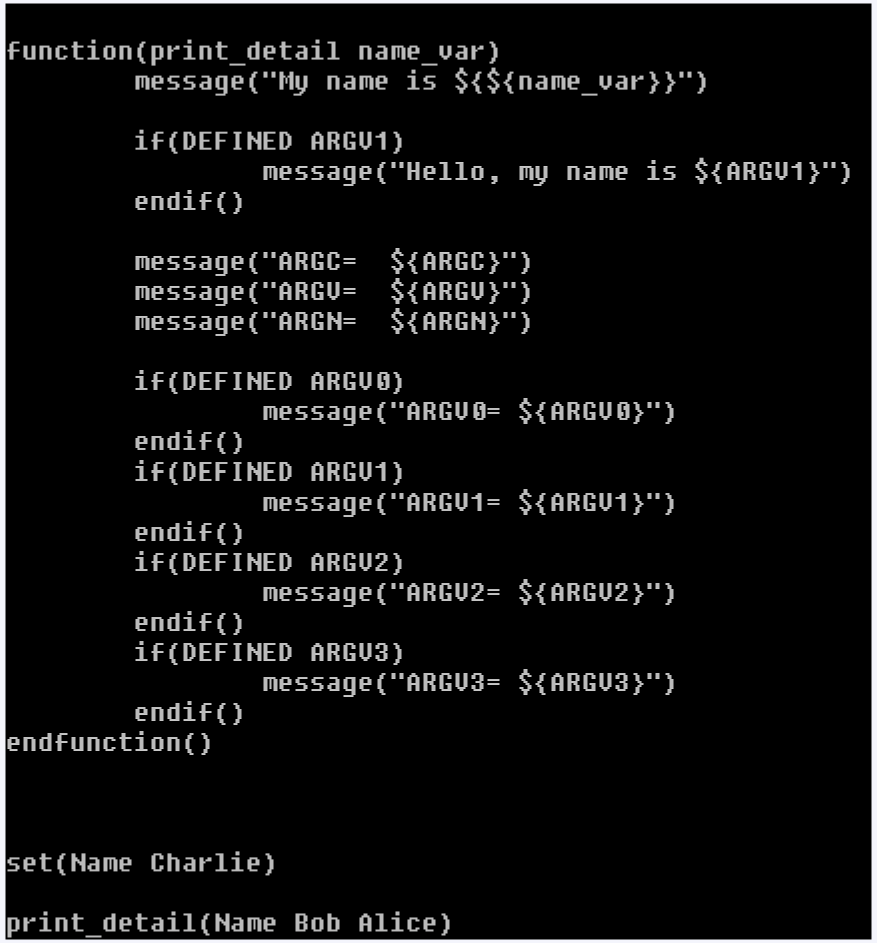
```
My Name is Charlie
My Age is 45
```

Functions

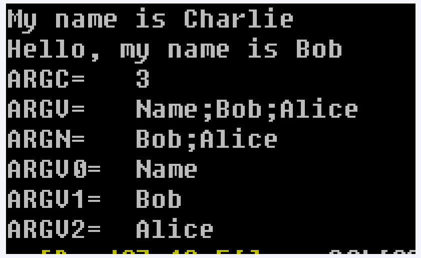
- Optional arguments

Special variables	Description
ARGC	Total count of arguments (named+optional)
ARGV	List of all arguments(named+optional)
ARGN	List of optional arguments
ARGV0	First argument
ARGV1	Second argument
ARGV2	Third argument

example12



```
function(print_detail name_var)
        message("My name is ${${name_var}}")

        if(DEFINED ARGV1)
                message("Hello, my name is ${ARGV1}")
        endif()

        message("ARGC=  ${ARGC}")
        message("ARGV=  ${ARGV}")
        message("ARGN=  ${ARGN}")

        if(DEFINED ARGV0)
                message("ARGV0= ${ARGV0}")
        endif()
        if(DEFINED ARGV1)
                message("ARGV1= ${ARGV1}")
        endif()
        if(DEFINED ARGV2)
                message("ARGV2= ${ARGV2}")
        endif()
        if(DEFINED ARGV3)
                message("ARGV3= ${ARGV3}")
        endif()
endfunction()



set(Name Charlie)

print_detail(Name Bob Alice)
```

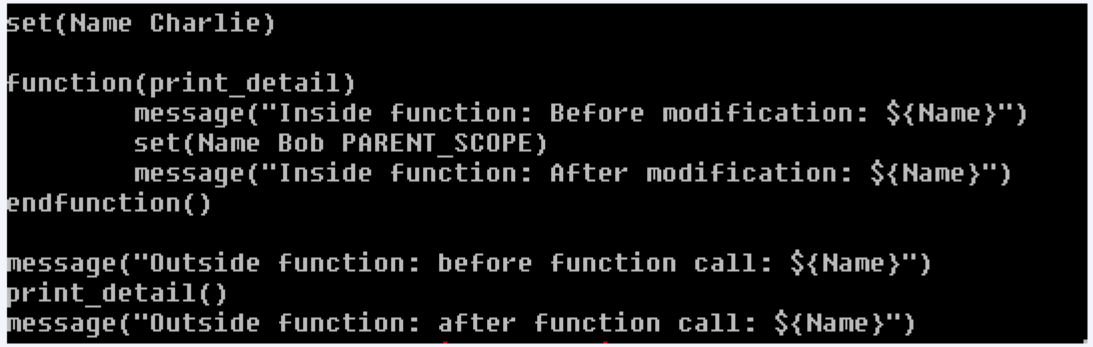


```
My name is Charlie
Hello, my name is Bob
ARGC=   3
ARGV=   Name;Bob;Alice
ARGN=   Bob;Alice
ARGV0=  Name
ARGV1=  Bob
ARGV2=  Alice
```

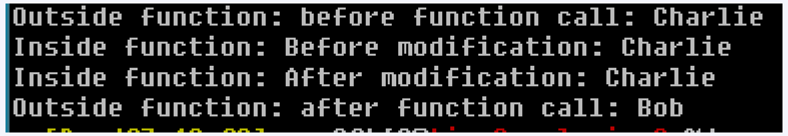
Scopes

- Whenever function is called in CMake a new scope created inside a current scope
- All the changes to variables inside function are **local**
- Functions do not return value in CMake

example13



```
set(Name Charlie)

function(print_detail)
        message("Inside function: Before modification: ${Name}")
        set(Name Bob PARENT_SCOPE)
        message("Inside function: After modification: ${Name}")
endfunction()

message("Outside function: before function call: ${Name}")
print_detail()
message("Outside function: after function call: ${Name}")
```



```
Outside function: before function call: Charlie
Inside function: Before modification: Charlie
Inside function: After modification: Charlie
Outside function: after function call: Bob
```

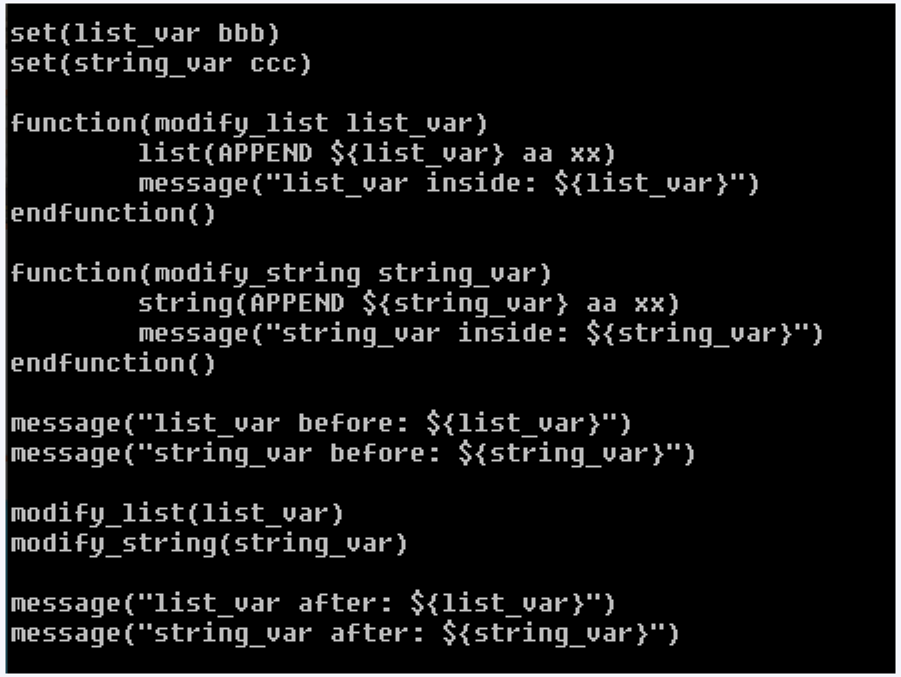
Variable 'Name' modified			
Parent Scope Name = Charlie	Parent Scope Name = Charlie	Parent Scope Name = Bob	Parent Scope Name = Bob
	Function Scope Name = Charlie	Function Scope Name = Charlie	
Function called 'Name' variable copied		Function execution ended	

VLAAMS SUPERCOMPUTER CENTRUM

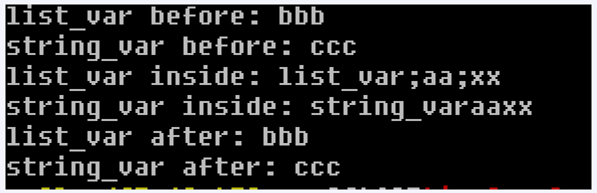
Scopes

- Modification is local
- Modifying lists and strings

```
function(modify_list list_var)
       list(APPEND ${list_var} aa xx)
endfuction()


function(modify_string string_var)
       string(APPEND ${string_var} aa xx)
endfuction()
```



```
set(list_var bbb)
set(string_var ccc)

function(modify_list list_var)
        list(APPEND ${list_var} aa xx)
        message("list_var inside: ${list_var}")
endfunction()

function(modify_string string_var)
        string(APPEND ${string_var} aa xx)
        message("string_var inside: ${string_var}")
endfunction()

message("list_var before: ${list_var}")
message("string_var before: ${string_var}")

modify_list(list_var)
modify_string(string_var)

message("list_var after: ${list_var}")
message("string_var after: ${string_var}")
```



```
list_var before: bbb
string_var before: ccc
list_var inside: list_var;aa;xx
string_var inside: string_varaaxx
list_var after: bbb
string_var after: ccc
```

VLAAMS SUPERCOMPUTER CENTRUM

Scopes

- add_subdirectory not scriptable, we need to build
- Adds a subdirectory to the build. The dir1, … specifies the directory in which the source CMakeLists.txt and code files are located. If it is a relative path, it will be evaluated with respect to the current directory (the typical usage), but it may also be an absolute path.
- The CMakeLists.txt file in the specified source directory will be processed immediately by CMake before processing in the current input file continues beyond this command.

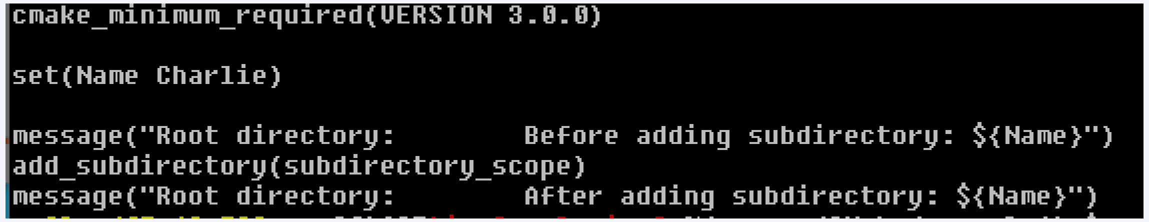
```
add_subdirectory(<dir1> <dir2> <dir3>…)
```

VLAAMS SUPERCOMPUTER CENTRUM

Scopes

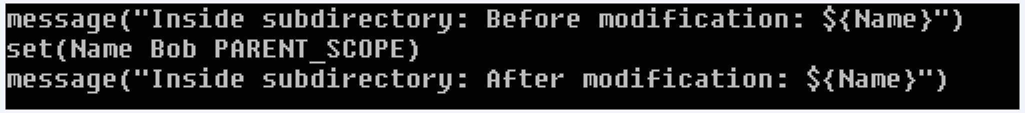
example14

CMakeLists.txt

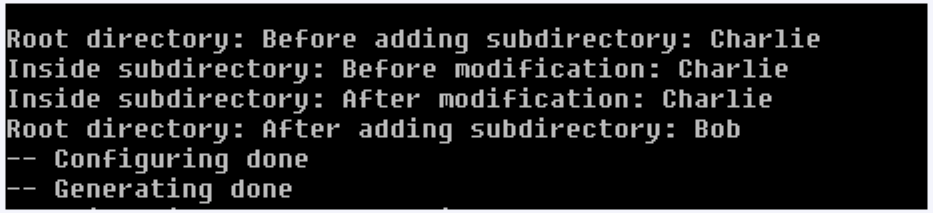


```
cmake_minimum_required(VERSION 3.0.0)

set(Name Charlie)

message("Root directory:        Before adding subdirectory: ${Name}")
add_subdirectory(subdirectory_scope)
message("Root directory:        After adding subdirectory: ${Name}")
```

subdirectory_scope/CMakeLists.txt



```
message("Inside subdirectory: Before modification: ${Name}")
set(Name Bob PARENT_SCOPE)
message("Inside subdirectory: After modification: ${Name}")
```



```
Root directory: Before adding subdirectory: Charlie
Inside subdirectory: Before modification: Charlie
Inside subdirectory: After modification: Charlie
Root directory: After adding subdirectory: Bob
-- Configuring done
-- Generating done
```

VLAAMS SUPERCOMPUTER CENTRUM

Macros

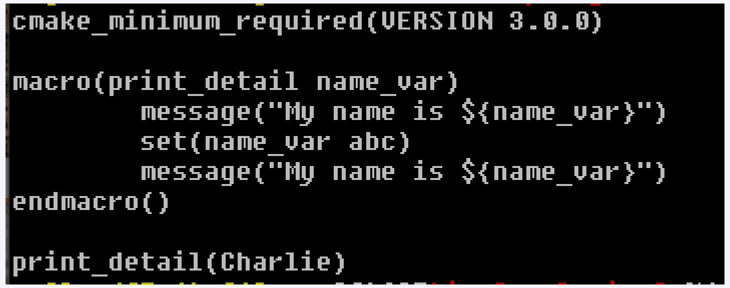
- Macros look like functions, accept named and optional arguments
- Marcos do not introduce a new scope
- Commands are executed in parent scope

```
macro(<function_name> <function_agrs>)
	<commands>
endmacro()
```

VLAAMS SUPERCOMPUTER CENTRUM

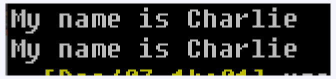
Macros

- Function vs macro

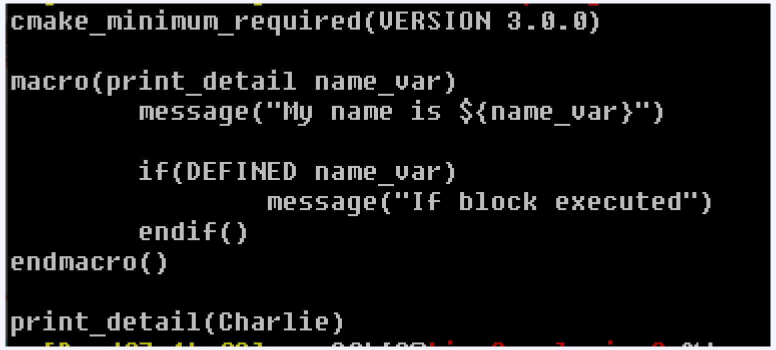


```
cmake_minimum_required(VERSION 3.0.0)

macro(print_detail name_var)
        message("My name is ${name_var}")
        set(name_var abc)
        message("My name is ${name_var}")
endmacro()

print_detail(Charlie)
```

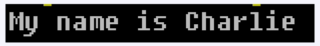
example15



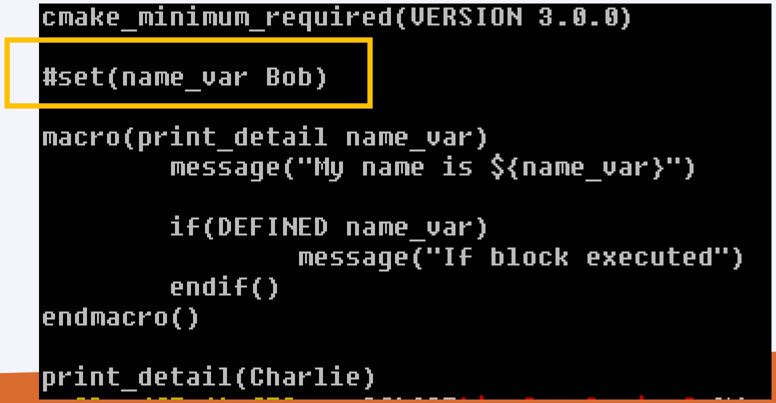
```
My name is Charlie
My name is Charlie
```



```
cmake_minimum_required(VERSION 3.0.0)

macro(print_detail name_var)
        message("My name is ${name_var}")

        if(DEFINED name_var)
                message("If block executed")
        endif()
endmacro()

print_detail(Charlie)
```



```
My name is Charlie
```



```
cmake_minimum_required(VERSION 3.0.0)

#set(name_var Bob)

macro(print_detail name_var)
        message("My name is ${name_var}")

        if(DEFINED name_var)
                message("If block executed")
        endif()
endmacro()

print_detail(Charlie)
```

VLAAMS SUPERCOMPUTER CENTRUM

Macros

- CMake commands, function names and macro names are **case insensitive**

```
function(FOO)
   <commands>
endfuction()
```

- `foo()`
- `fOo()`
- `FoO()`

- `target_link_library()`
- `TARGET_LINK_LIBRARY()`
- `target_Link_library()`
- `target_link_LIBrary()`

Modules

- So far, we have been writing our CMake codes in CMakelists.txt files. These files are collectively called the ListFiles.
- Apart from the listfiles, we also have the concept of modules, where the CMake codes are written. These modules have .cmake extension.

- CMake provides some standard modules containing the CMake codes so that we can directly use those in any project. You can find those in the `$EBROOTCMAKE/share/cmake-3.20/Modules` directory.
- These modules can be used with the include() command. If you want to use this module, you need to write these 2 lines of code and then the variable VAR will contain the number of processors.

```
include(ProcessorCount)
ProcessorCount(VAR)
message("Number of processors are: ${VAR}")
```

VLAAMS SUPERCOMPUTER CENTRUM

Modules

- You can use this VAR variable in your project if you want to run parallel jobs of any process.
- Apart from the standard modules, you can also make your own module.

Modules

- We can create this directory structure with 2 files CMakeLists.txt and my_module.cmake
- The CMakeLists.txt file will contain the following lines:

```
cmake_minimum_required(VERSION 3.0.0)
project(Calculator_Project VERSION 1.0.0)
include(my_module)
```

- and the my_module.cmake file will contain just the following line:

```
message("Hello from the my_module.cmake file!")
```

- At this time running the CMake command will give an error, because we also need to specify the path which contains the my_module.cmake file.

Modules

- To specify that path, we have a variable called CMAKE_MODULE_PATH which contains the lists of paths to search the module. This variable is a cache variable

```
cmake_minimum_required(VERSION 3.0.0)
project(Calculator_Project VERSION 1.0.0)
list(APPEND  CMAKE_MODULE_PATH  <path-to-example16-directory>)
include(my_module)
```

- Now we can run the cmake .. command from the build directory and the output will be:

```
Hello from the my_module.cmake file!
```

example16

VLAAMS
SUPERCOMPUTER
CENTRUM

Modules

- Now you might think that this command is similar to add_subdirectory() command, but it's not! When we use the include() command, we do not introduce a new scope. This means that if we set or modify a variable inside the my_module.cmake file, that modification is going to reflect inside the CMakeLists.txt file.

- The modules are often used if we want to have reusable code in our project. Also, if your CMakeLists.txt file is too long, some part of it can be written inside another .cmake file; to improve the readability of the code.

VLAAMS SUPERCOMPUTER CENTRUM

Normal variables

- Normal variables

```
set(Name Charlie)
set Age (50)
function(print_detail var)
   message (“My ${var} is ${${var}}”)
endfuction()
print_detail(Name)       ←——————        New scope
print_detail(Age)        ←——————        Another new scope
```

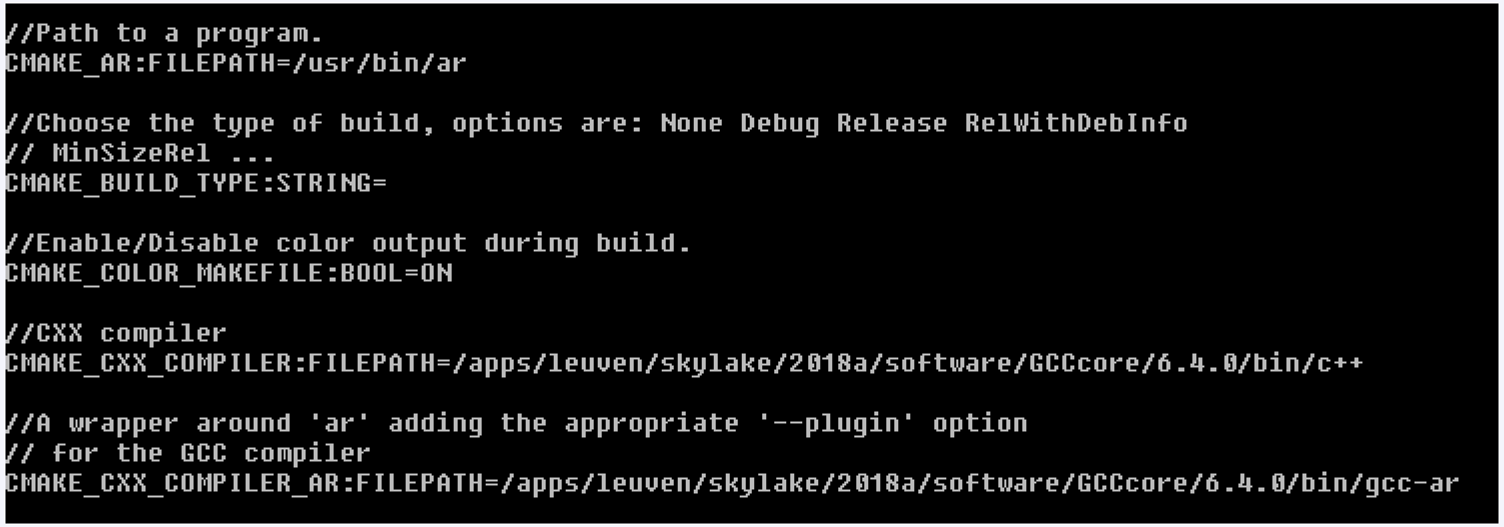
Global variables

- 2 types of variables with global scope
  - Persistent cache variables
  - Environment variables

Cache variables

- All variables written inside CMake cache file are cache variables

- Cache variables:
  - Set by CMake, depending on the Development environment
  - Set by commands inside CMakeList.txt



```
//Path to a program.
CMAKE_AR:FILEPATH=/usr/bin/ar

//Choose the type of build, options are: None Debug Release RelWithDebInfo
// MinSizeRel ...
CMAKE_BUILD_TYPE:STRING=

//Enable/Disable color output during build.
CMAKE_COLOR_MAKEFILE:BOOL=ON

//CXX compiler
CMAKE_CXX_COMPILER:FILEPATH=/apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/c++

//A wrapper around 'ar' adding the appropriate '--plugin' option
// for the GCC compiler
CMAKE_CXX_COMPILER_AR:FILEPATH=/apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/gcc-ar
```

Cache variables

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)


add_subdirectory(my_math)

add_subdirectory(my_print)


add_executable(calculator
main.cpp)


target_link_libraries(calculator PRIVATE my_math  my_print)


set(A "123" CACHE STRING "This command sets variable A in persistent cache")
message($CACHE{A})
```

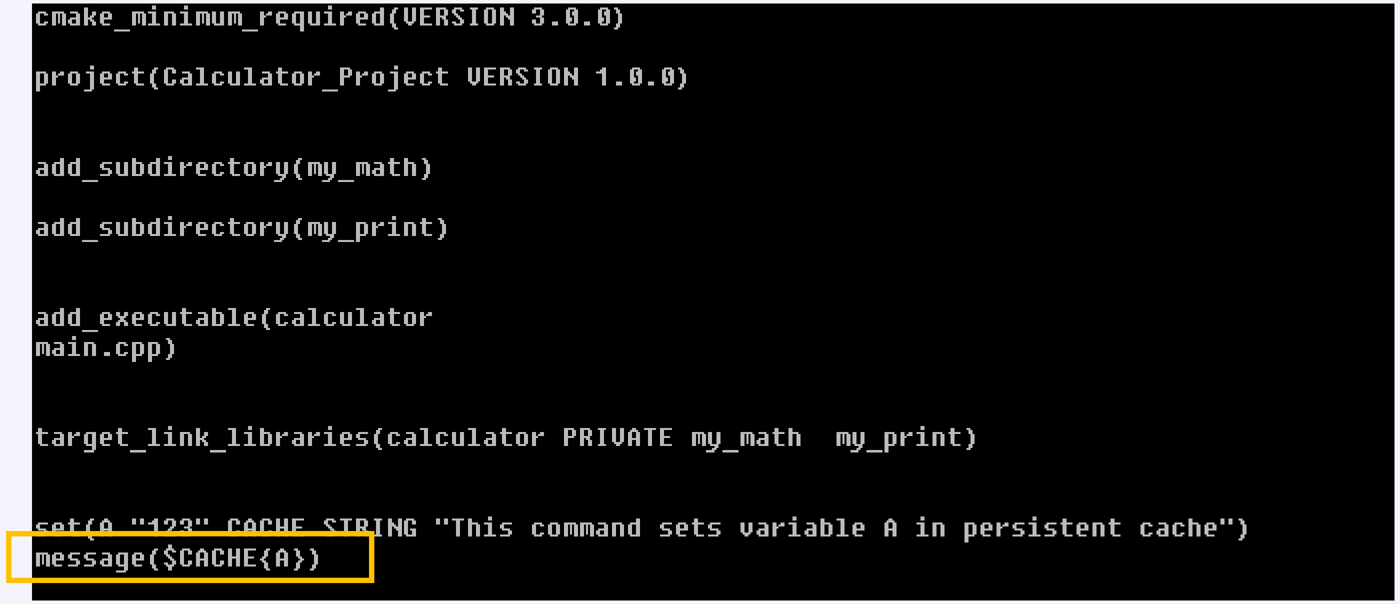
CMakeCache.txt

```
########################
# EXTERNAL cache entries
########################

//This command sets variable A in persistent cache
A:STRING=123

//Path to a program.
CMAKE_ADDR2LINE:FILEPATH=/usr/bin/addr2line

//Path to a program.
CMAKE_AR:FILEPATH=/usr/bin/ar
```

example17

Cache variables

- Cache variable dereferencing
- `$CACHE(variable_name)`



```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)


add_subdirectory(my_math)

add_subdirectory(my_print)


add_executable(calculator
main.cpp)


target_link_libraries(calculator PRIVATE my_math  my_print)


set(A "123" CACHE STRING "This command sets variable A in persistent cache")
message($CACHE{A})
```

Cache variables

- First searched in local scope, if not found – in global scope (cache)

```
set(A "000")
set(A "123" CACHE STRING "A is a cache variable")
message(${A})                Output: 000
```

```
set(A "000")
set(A "123" CACHE STRING "A is a cache variable")
message($CACHE{A})      Output: 123
```

Direct lookup in cache, ignoring existing normal variable

Environment variables

- Environment variables
  - Global scope
  - Not stored in CMakeCache.txt

- Setting env variables:

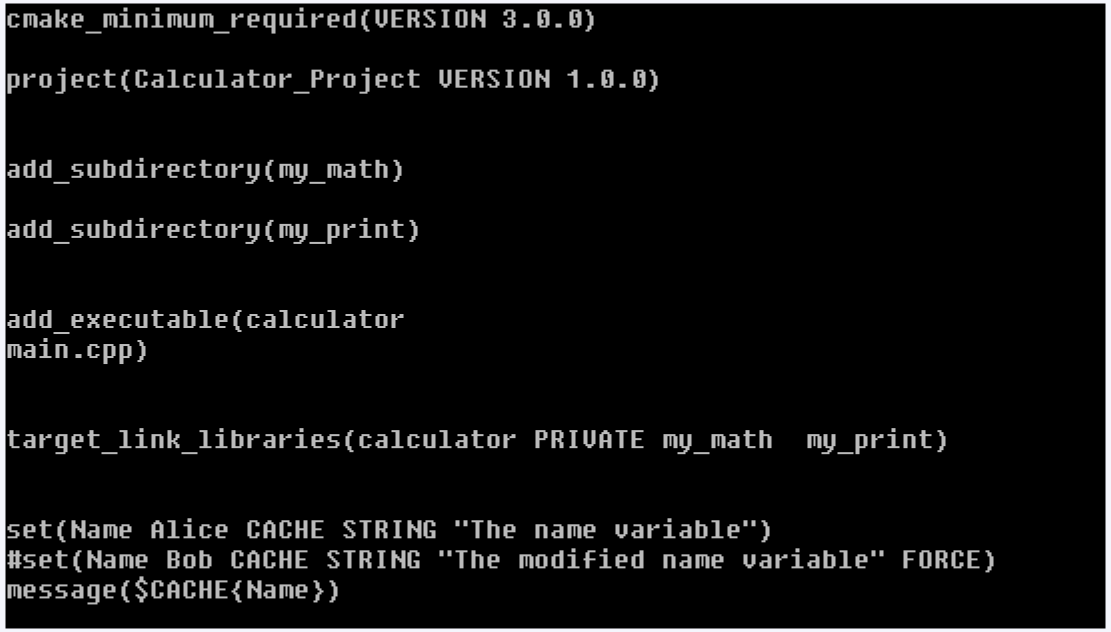
```
set(ENV <variable_name> <variable_value>)
```

- Dereferencing env variables

```
$ENV{variable_name}
```

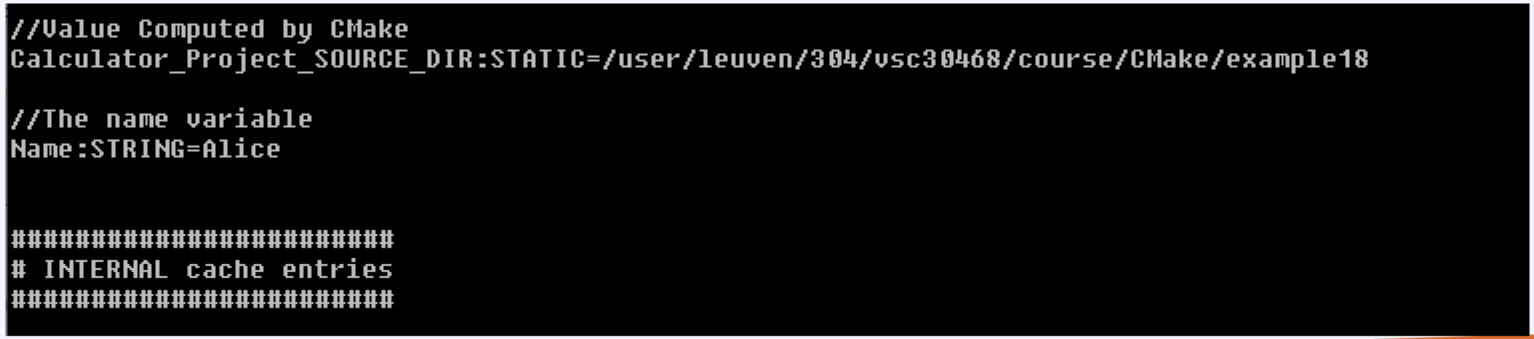
Cache variables

- Cache entry not modified



```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)


add_subdirectory(my_math)

add_subdirectory(my_print)


add_executable(calculator
main.cpp)


target_link_libraries(calculator PRIVATE my_math  my_print)


set(Name Alice CACHE STRING "The name variable")
#set(Name Bob CACHE STRING "The modified name variable" FORCE)
message($CACHE{Name})
```

example18



```
//Value Computed by CMake
Calculator_Project_SOURCE_DIR:STATIC=/user/leuven/304/vsc30468/course/CMake/example18

//The name variable
Name:STRING=Alice


########################
# INTERNAL cache entries
########################
```

VLAAMS SUPERCOMPUTER CENTRUM

Cache variables

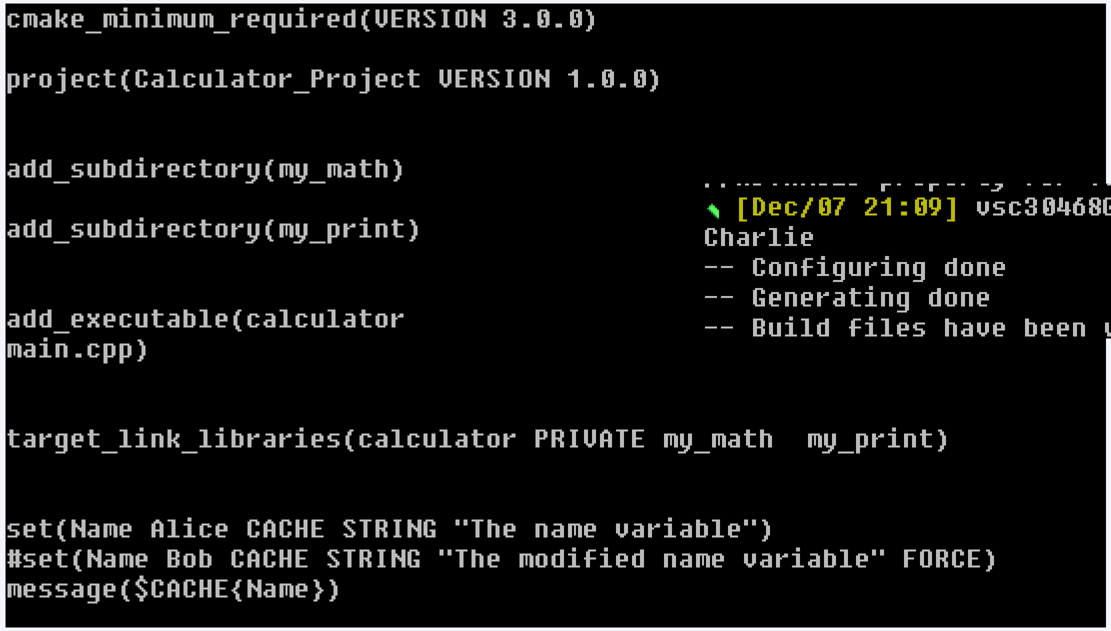
- Running CMake first time –> CMakeCache.txt is created, modify is going to be rejected

- Modifying cache variables:
  - Edit CMakeCache.txt file
  - Use FORCE keyword
  - Use –D flag

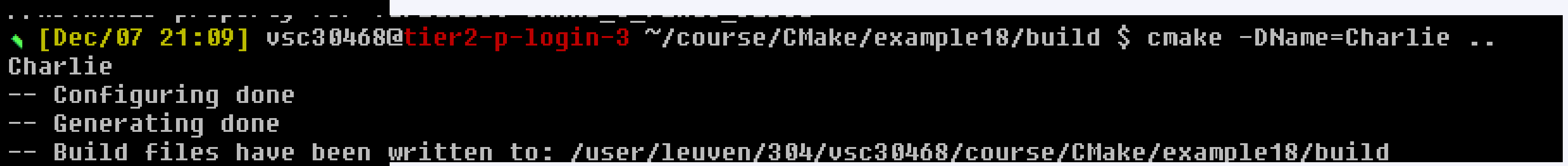
- Recommended not to use force, using -D recommended

- Using both force and –D -> Force has higher priority

Cache variables



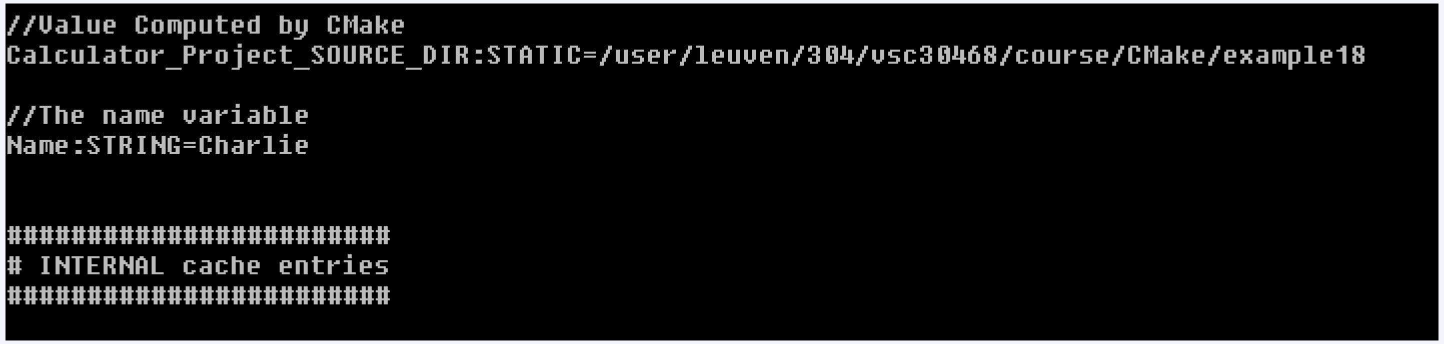
```
cmake_minimum_required(VERSION 3.0.0)

project(Calculator_Project VERSION 1.0.0)


add_subdirectory(my_math)

add_subdirectory(my_print)


add_executable(calculator
main.cpp)


target_link_libraries(calculator PRIVATE my_math  my_print)


set(Name Alice CACHE STRING "The name variable")
#set(Name Bob CACHE STRING "The modified name variable" FORCE)
message($CACHE{Name})
```



```
[Dec/07 21:09] vsc30468@tier2-p-login-3 ~/course/CMake/example18/build $ cmake -DName=Charlie ..
Charlie
-- Configuring done
-- Generating done
-- Build files have been written to: /user/leuven/304/vsc30468/course/CMake/example18/build
```

example18

CMakeCache.txt



```
//Value Computed by CMake
Calculator_Project_SOURCE_DIR:STATIC=/user/leuven/304/vsc30468/course/CMake/example18

//The name variable
Name:STRING=Charlie


########################
# INTERNAL cache entries
########################
```

VLAAMS
SUPERCOMPUTER
CENTRUM

Cache variables

- Most frequently used cache variables

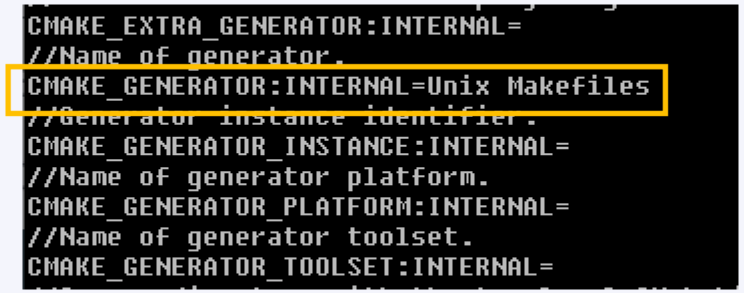
For CMAKE_VERSION 3.20.1

`CMAKE_MAJOR_VERSION`	3
`CMAKE_MINOR_VERSION`	20
`CMAKE_PATCH_VERSION`	1

- `CMAKE_PROJECT_NAME`: set to the name of the project (PROJECT_NAME) when we execute project command, can be specified in any directory, always refers to top project name

Generators

- About the build system, currently make – can be verified from cache variable entry `CMAKE_GENERATOR`
- `cmake -help` shows available generators (should be installed first)
- `cmake -DCMAKE_GENERATOR=Ninja ..; ninja (for build.ninja file generated)`
- We can also change generator with `-G` flag: `cmake -GNinja ..`

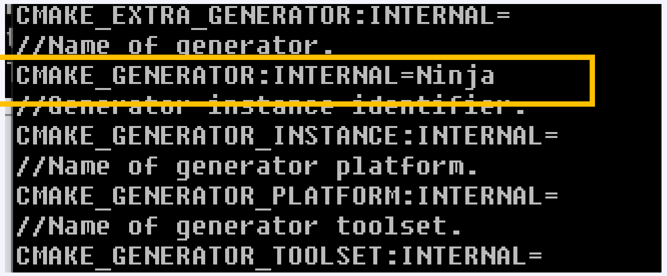


```
CMAKE_EXTRA_GENERATOR:INTERNAL=
//Name of generator.
CMAKE_GENERATOR:INTERNAL=Unix Makefiles
//Generator instance identifier.
CMAKE_GENERATOR_INSTANCE:INTERNAL=
//Name of generator platform.
CMAKE_GENERATOR_PLATFORM:INTERNAL=
//Name of generator toolset.
CMAKE_GENERATOR_TOOLSET:INTERNAL=
```

Tip: Clean the build dir before changing generator

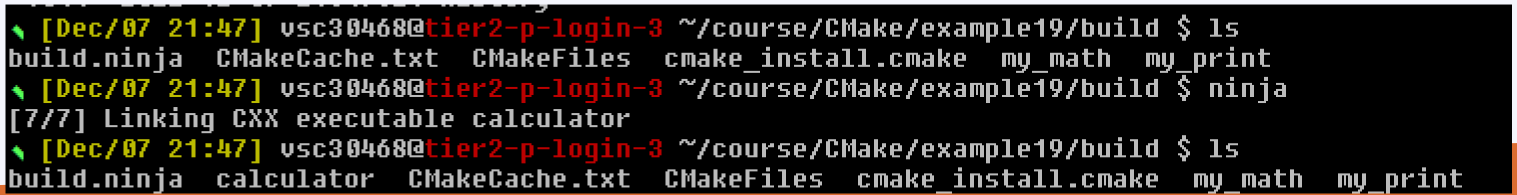
```
$ rm -r *
```

```
$ module load Ninja/1.10.2-GCCcore-10.3.0
$ cmake -GNinja ..
```



```
CMAKE_EXTRA_GENERATOR:INTERNAL=
//Name of generator.
CMAKE_GENERATOR:INTERNAL=Ninja
//Generator instance identifier.
CMAKE_GENERATOR_INSTANCE:INTERNAL=
//Name of generator platform.
CMAKE_GENERATOR_PLATFORM:INTERNAL=
//Name of generator toolset.
CMAKE_GENERATOR_TOOLSET:INTERNAL=
```

example19



```
[Dec/07 21:47] vsc30468@tier2-p-login-3 ~/course/CMake/example19/build $ ls
build.ninja  CMakeCache.txt  CMakeFiles  cmake_install.cmake  my_math  my_print
[Dec/07 21:47] vsc30468@tier2-p-login-3 ~/course/CMake/example19/build $ ninja
[7/7] Linking CXX executable calculator
[Dec/07 21:47] vsc30468@tier2-p-login-3 ~/course/CMake/example19/build $ ls
build.ninja  calculator  CMakeCache.txt  CMakeFiles  cmake_install.cmake  my_math  my_print
```

VLAAMS SUPERCOMPUTER CENTRUM

Installation

- Library on the system is usually part of package (download-compile-install process)
- Install – is copying the files – items to copy and the destination
- How to check: where - print `message({CMAKE_INSTALL_PREFIX})`

```
install(FILES<file_name> DESTINATION<dir>)

or

install(TARGETS<tgt_name> DESTINATION<dir>)
```

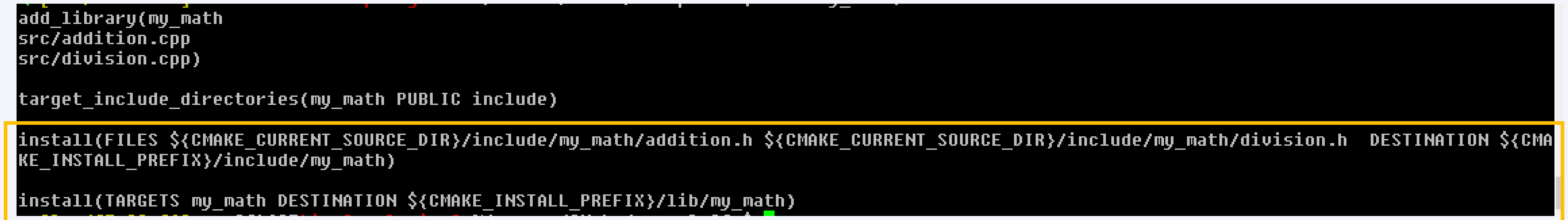
Installation

- Recommendation:
  - install() command -> destination
  - Header files: -> /usr/local/include/<package_name>
  - Targets: -> /usr/local/lib/<package_name>

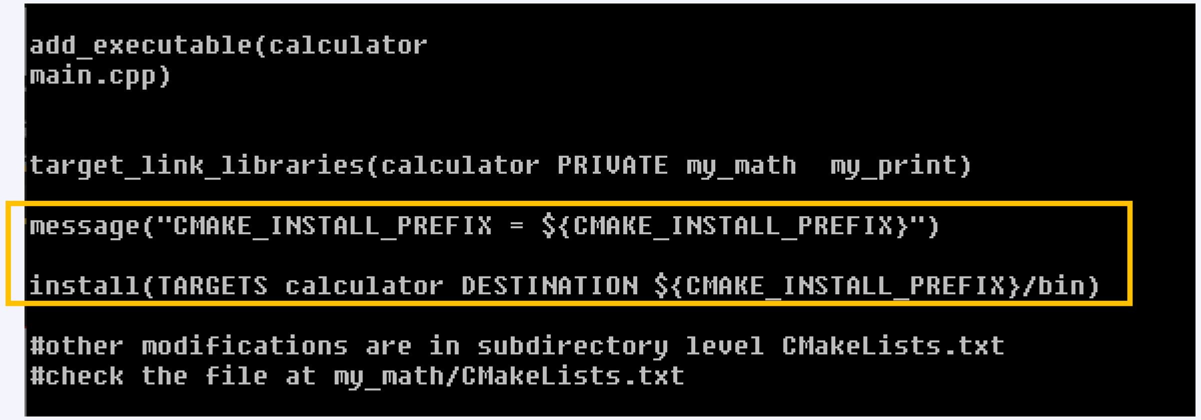
- Nothing is installed in include or lib folder after `make` command, only after `(sudo) make install`

- If we edited CMakefile and forgot to run cmake the maketool will run cmake for us.
- BUT: not available when first time building project – there is no Makefile yet.

Installation

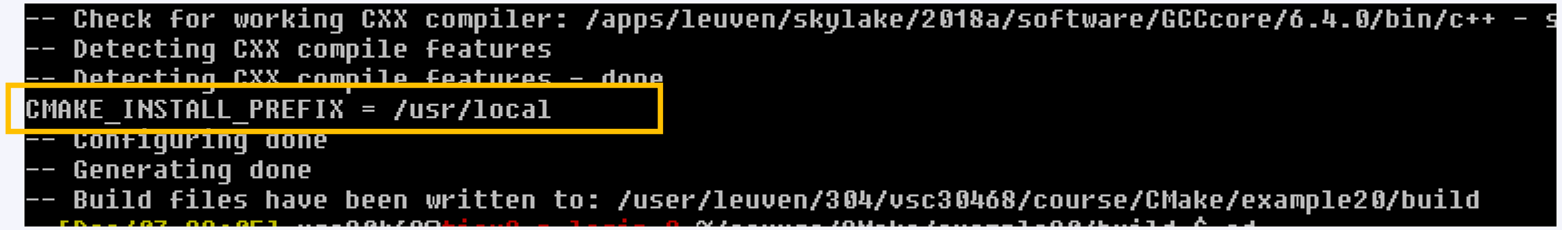


```
add_library(my_math
src/addition.cpp
src/division.cpp)

target_include_directories(my_math PUBLIC include)

install(FILES ${CMAKE_CURRENT_SOURCE_DIR}/include/my_math/addition.h ${CMAKE_CURRENT_SOURCE_DIR}/include/my_math/division.h  DESTINATION ${CMA
KE_INSTALL_PREFIX}/include/my_math)

install(TARGETS my_math DESTINATION ${CMAKE_INSTALL_PREFIX}/lib/my_math)
```



```
add_executable(calculator
main.cpp)


target_link_libraries(calculator PRIVATE my_math  my_print)

message("CMAKE_INSTALL_PREFIX = ${CMAKE_INSTALL_PREFIX}")

install(TARGETS calculator DESTINATION ${CMAKE_INSTALL_PREFIX}/bin)

#other modifications are in subdirectory level CMakeLists.txt
#check the file at my_math/CMakeLists.txt
```

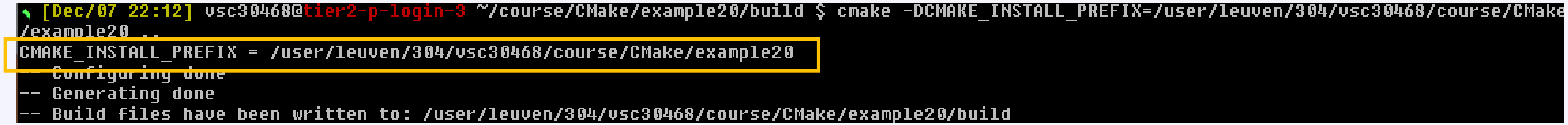
example20



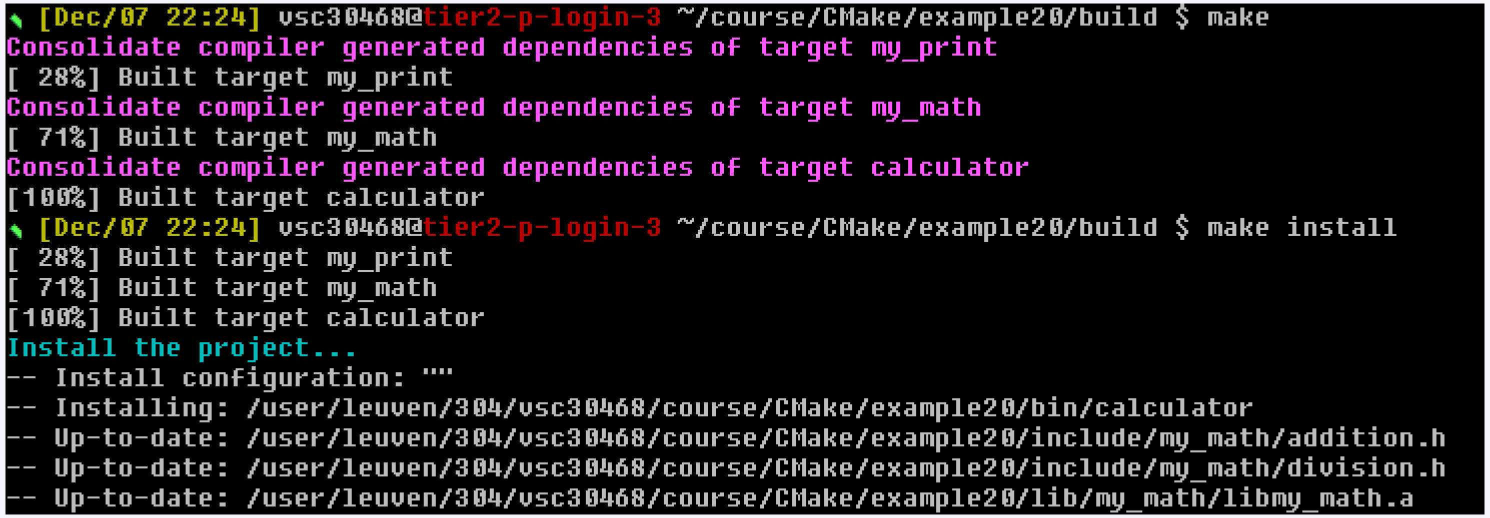
```
-- Check for working CXX compiler: /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/c++ - s
-- Detecting CXX compile features
-- Detecting CXX compile features - done
CMAKE_INSTALL_PREFIX = /usr/local
-- Configuring done
-- Generating done
-- Build files have been written to: /user/leuven/304/vsc30468/course/CMake/example20/build
```

VLAAMS SUPERCOMPUTER CENTRUM

Installation



```
[Dec/07 22:12] vsc30468@tier2-p-login-3 ~/course/CMake/example20/build $ cmake -DCMAKE_INSTALL_PREFIX=/user/leuven/304/vsc30468/course/CMake
/example20 ..
CMAKE_INSTALL_PREFIX = /user/leuven/304/vsc30468/course/CMake/example20
-- Configuring done
-- Generating done
-- Build files have been written to: /user/leuven/304/vsc30468/course/CMake/example20/build
```



```
[Dec/07 22:24] vsc30468@tier2-p-login-3 ~/course/CMake/example20/build $ make
Consolidate compiler generated dependencies of target my_print
[ 28%] Built target my_print
Consolidate compiler generated dependencies of target my_math
[ 71%] Built target my_math
Consolidate compiler generated dependencies of target calculator
[100%] Built target calculator
[Dec/07 22:24] vsc30468@tier2-p-login-3 ~/course/CMake/example20/build $ make install
[ 28%] Built target my_print
[ 71%] Built target my_math
[100%] Built target calculator
Install the project...
-- Install configuration: ""
-- Installing: /user/leuven/304/vsc30468/course/CMake/example20/bin/calculator
-- Up-to-date: /user/leuven/304/vsc30468/course/CMake/example20/include/my_math/addition.h
-- Up-to-date: /user/leuven/304/vsc30468/course/CMake/example20/include/my_math/division.h
-- Up-to-date: /user/leuven/304/vsc30468/course/CMake/example20/lib/my_math/libmy_math.a
```

VLAAMS
SUPERCOMPUTER
CENTRUM

Comments in CMake

- We can have single line comment by prepending a line with # , for example:

```
# This is a comment
```

- We can also have multi-line comment by wrapping the lines between #[[and #]] , for example:

```
#[[ This is comment line 1
    This is comment line 2
    This is comment line 3
#]]
```

- We can also un-comment the multi-line comments by prepending #[[and #]] with # , for example:

```
##[[ This is NOT comment line 1
     This is NOT comment line 2
     This is NOT comment line 3
##]]
```

Comments in CMake

- We can also have nested comments, with the help of =, like this:

```
#[==[ This is comment line 1
      This is comment line 2
      #[=[This is comment line 3
          This is comment line 4
          #[[This is comment line 5
          #]]This is comment line 6
          This is comment line 7
      #]=]This is comment line 8
      This is comment line 9
#]==]
```

CMake scripting

example21

- Running CMakeLists.txt in Script mode using -P option:

```
cmake -P CMakeLists.txt
```

- There is another fancy way of doing this in the Linux terminal, which is similar to executing a bash or a python script.
- Run the command `which cmake` in your terminal. It will give you the location of the cmake executable. In my case, it is `/apps/leuven/rocky8/skylake/2021a/software/CMake/3.22.1-GCCcore-10.3.0/bin/cmake`
- Open the CMakeLists.txt file and paste the following code in its first line

```
#! /apps/leuven/rocky8/skylake/2021a/software/CMake/3.22.1-GCCcore-10.3.0/bin/cmake -P
```

- Give the execution permission to the CMakeLists.txt file by running the command: `chmod +x CMakeLists.txt`
- Now, you can either use the command `./CMakeLists.txt` to run the CMakeLists.txt file in the script mode.

Debug vs Release

- How can we handle different build configurations like debug, release etc. using CMakeLists.txt file?

Make a project, which does not dependent on any external library. In this case, you can directly set a normal variable CMAKE_BUILD_TYPE to Debug or Release, while generating build system files. To do that, simply execute

`cmake -DCMAKE_BUILD_TYPE=Debug ..` or
`cmake -DCMAKE_BUILD_TYPE=Release ..` commands.

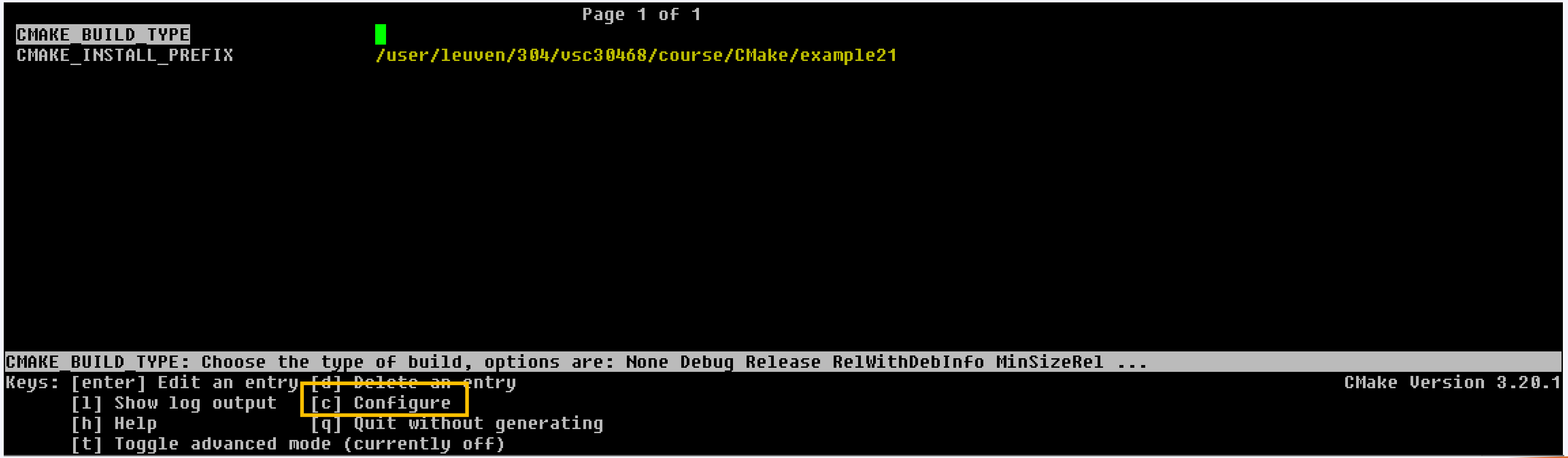
Release build is faster and also has less file size compared to the debug build. When you set the CMAKE_BUILD_TYPE variable, the compiler flags are automatically modified to offer you the desired optimization levels.

Command line usage

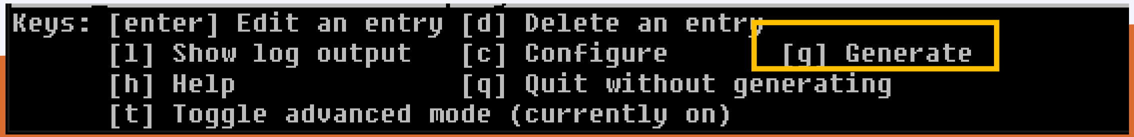
- Can be used from scripts
- Go to the build directory
- Pass path to source tree
- Use -G to select build system generator
- Use -D to set cache variables

GUI usage

- `$ccmake ..` (Linux)
- CMakeSetup (Windows)
- Editing cache entries to configure the build
- Use configure (c) button after the change
- Use generate (g) button after config is done



```
                                                     Page 1 of 1
 CMAKE_BUILD_TYPE
 CMAKE_INSTALL_PREFIX             /user/leuven/304/vsc30468/course/CMake/example21











CMAKE_BUILD_TYPE: Choose the type of build, options are: None Debug Release RelWithDebInfo MinSizeRel ...
Keys: [enter] Edit an entry [d] Delete an entry                                          CMake Version 3.20.1
      [l] Show log output   [c] Configure
      [h] Help              [q] Quit without generating
      [t] Toggle advanced mode (currently off)
```

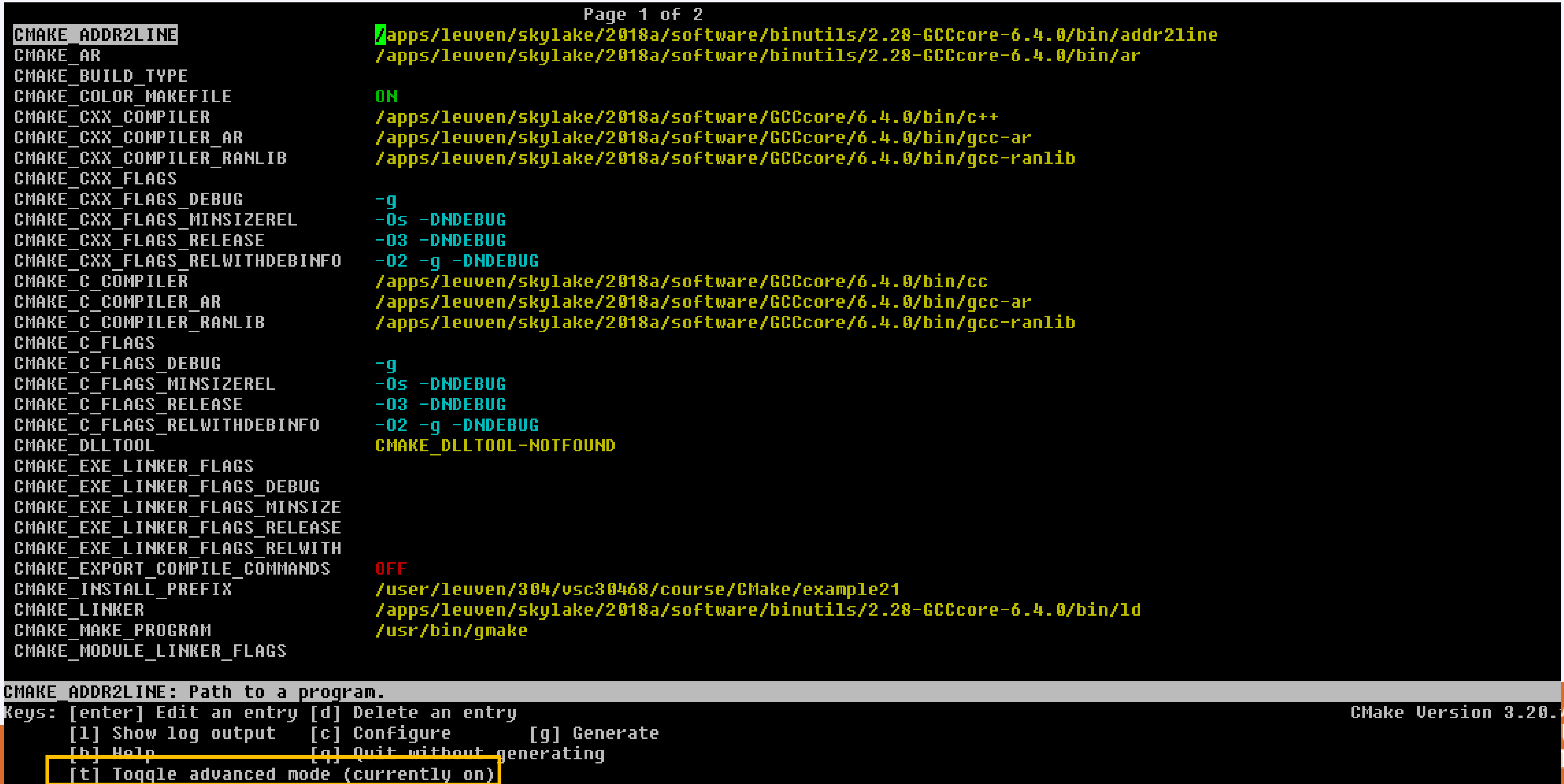


```
Keys: [enter] Edit an entry [d] Delete an entry
      [l] Show log output   [c] Configure         [g] Generate
      [h] Help              [q] Quit without generating
      [t] Toggle advanced mode (currently on)
```

VLAAMS SUPERCOMPUTER CENTRUM

GUI usage

- t – advanced mode (more settings)



```
                                                              Page 1 of 2
CMAKE_ADDR2LINE                  /apps/leuven/skylake/2018a/software/binutils/2.28-GCCcore-6.4.0/bin/addr2line
CMAKE_AR                         /apps/leuven/skylake/2018a/software/binutils/2.28-GCCcore-6.4.0/bin/ar
CMAKE_BUILD_TYPE
CMAKE_COLOR_MAKEFILE             ON
CMAKE_CXX_COMPILER               /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/c++
CMAKE_CXX_COMPILER_AR            /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/gcc-ar
CMAKE_CXX_COMPILER_RANLIB        /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/gcc-ranlib
CMAKE_CXX_FLAGS
CMAKE_CXX_FLAGS_DEBUG            -g
CMAKE_CXX_FLAGS_MINSIZEREL       -Os -DNDEBUG
CMAKE_CXX_FLAGS_RELEASE          -O3 -DNDEBUG
CMAKE_CXX_FLAGS_RELWITHDEBINFO   -O2 -g -DNDEBUG
CMAKE_C_COMPILER                 /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/cc
CMAKE_C_COMPILER_AR              /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/gcc-ar
CMAKE_C_COMPILER_RANLIB          /apps/leuven/skylake/2018a/software/GCCcore/6.4.0/bin/gcc-ranlib
CMAKE_C_FLAGS
CMAKE_C_FLAGS_DEBUG              -g
CMAKE_C_FLAGS_MINSIZEREL         -Os -DNDEBUG
CMAKE_C_FLAGS_RELEASE            -O3 -DNDEBUG
CMAKE_C_FLAGS_RELWITHDEBINFO     -O2 -g -DNDEBUG
CMAKE_DLLTOOL                    CMAKE_DLLTOOL-NOTFOUND
CMAKE_EXE_LINKER_FLAGS
CMAKE_EXE_LINKER_FLAGS_DEBUG
CMAKE_EXE_LINKER_FLAGS_MINSIZE
CMAKE_EXE_LINKER_FLAGS_RELEASE
CMAKE_EXE_LINKER_FLAGS_RELWITH
CMAKE_EXPORT_COMPILE_COMMANDS    OFF
CMAKE_INSTALL_PREFIX             /user/leuven/304/vsc30468/course/CMake/example21
CMAKE_LINKER                     /apps/leuven/skylake/2018a/software/binutils/2.28-GCCcore-6.4.0/bin/ld
CMAKE_MAKE_PROGRAM               /usr/bin/gmake
CMAKE_MODULE_LINKER_FLAGS

CMAKE_ADDR2LINE: Path to a program.
Keys: [enter] Edit an entry [d] Delete an entry                                          CMake Version 3.20.
      [l] Show log output   [c] Configure        [g] Generate
      [h] Help              [q] Quit without generating
      [t] Toggle advanced mode (currently on)
```

AAMS
UTER
TRUM

Developer Documentation

- Command-line documentation:
  - Run `cmake --help` for summary
  - Run `cmake --help COMMAND` for detailed help with a specific list/file command
  - Try "`cmake --help IF`"
- Online documentation:
  - https://cmake.org/cmake/help/latest/

Questions

Helpdesk:
hpcinfo@kuleuven.be or https://admin.kuleuven.be/icts/HPCinfo_form/HPC-info-formulier

VSC web site:
http://www.vscentrum.be/

VSC documentation: https://docs.vscentrum.be

VSC agenda: training sessions, events

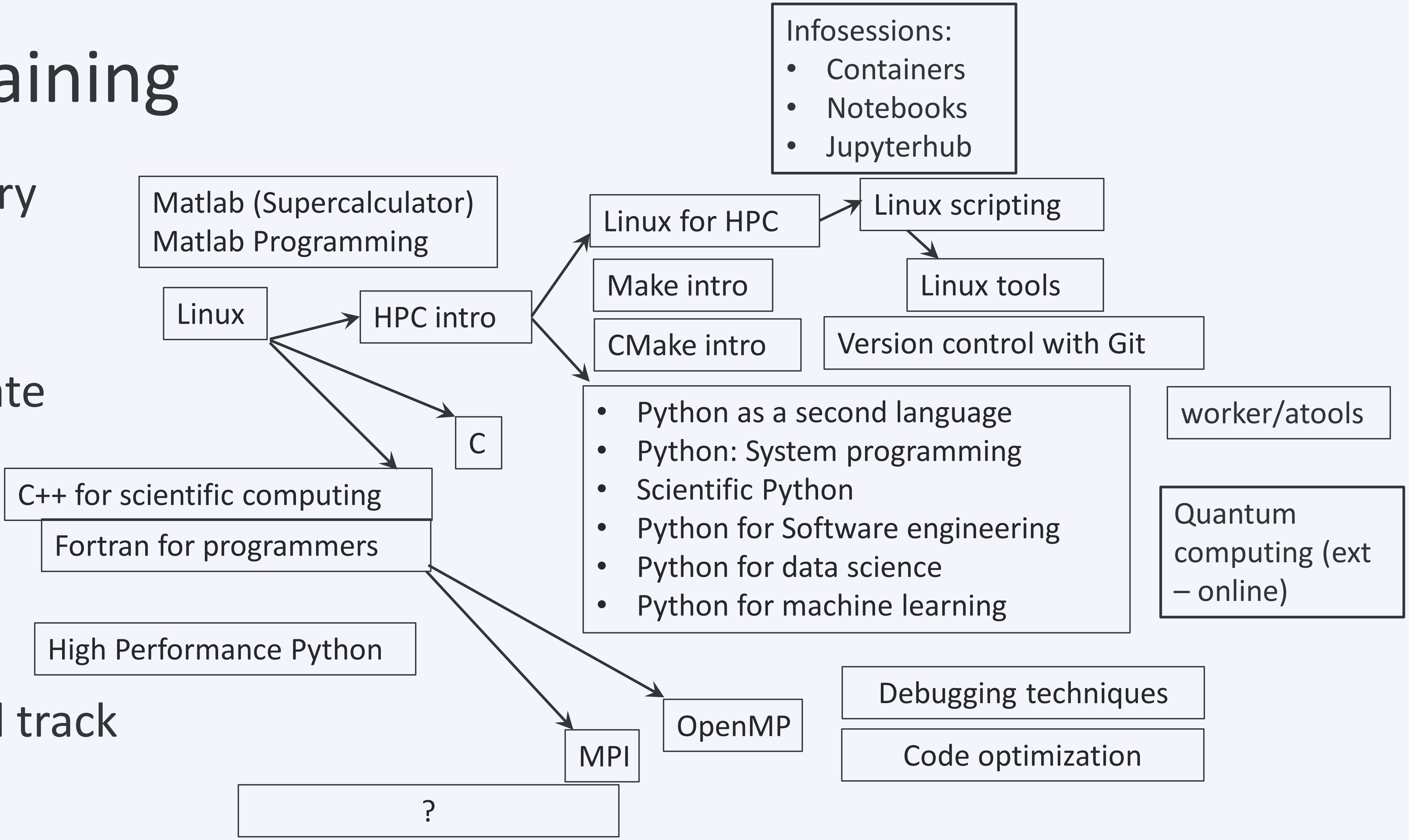
Systems status page:
http://status.vscentrum.be

Stay Connected to VSC

LinkedIn®

VSC training

- Introductory
- Intermediate
- Advanced
- Specialized track



PRACE MOOC Defensive programming and debugging and Fortran for Programmers (announced by e-mail)

Stay up-to-date https://www.vscentrum.be/en/education-and-trainings