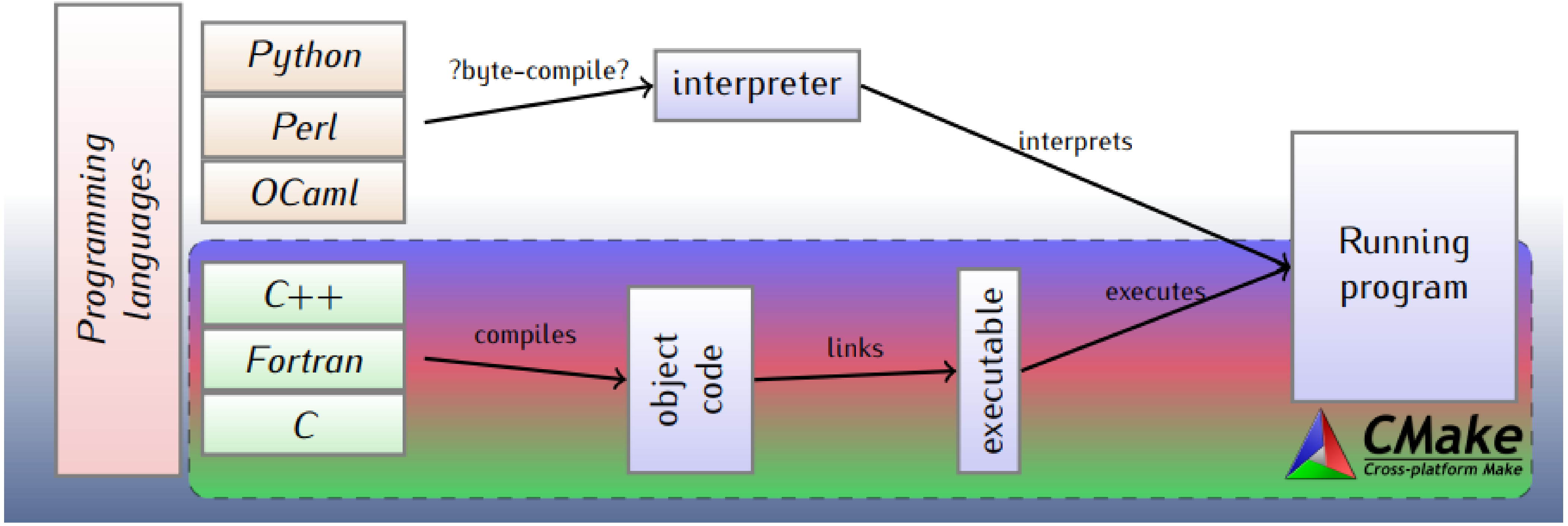
CMAKE


Programming languages
Python
Perl
OCaml
?byte-compile?
interpreter
interprets
Running program
C++
Fortran
C
compiles
object code
links
executable
executes
CMake
Cross-platform Make

Składnia plików CmakeLists.txt

- Komentarz: # to jest komentarz
- wywołanie polecenia: COMMAND(arg1 arg2 ...)
- listy wartości: A; B; C
- zmienne ${VAR}
- instrukcje warunkowe:
  - IF() ... ELSE()/ELSEIF() ... ENDIF()
  - WHILE() ... ENDWHILE()
  - FOREACH() ... ENDFOREACH()
- I wyrażenia regularne

Polecenia

- Prosta składnia
- Każde polecenie zaczyna się od nowej linii
- Argumenty oddzielone są spacją

```
COMMAND(ARG “ARG WITH SPACES”
        ${A_LIST} “${A_STRING}”)
```

```
TARGET_LINK_LIBRARIES(myTarget lib1 lib2)
FIND_LIBRARY(MY_LIB NAMES my1 my2
                    PATHS /foo /bar)
```

Zmienne

- Konwencja nazw jak w C
- Wartość są zawsze literałem łańcuchowym
- Ustawiane poprzez polecenie SET
- Odnosimy się zawsze ${VAR}

```
SET(A_LIST ${A_LIST} foo)
SET(A_STRING “${A_STRING} bar”)
```

Struktury kontrolne

- IF

```
IF(CONDITION)
  MESSAGE(“Yes”)
ELSE(CONDITION)
  MESSAGE(“No”)
ENDIF(CONDITION)
```

- FOREACH

```
FOREACH(c A B C)
  MESSAGE(“${c}: ${${c}}”)
ENDFOREACH(c)
```

- MACRO

```
MACRO(MY_MACRO arg1 arg2)
  SET(${arg1} “${${arg2}}”)
ENDMACRO(MY_MACRO)
MY_MACRO(A B)
```

Przykład projektu

CMakeLists.txt

```
PROJECT(FOO)
SUBDIRS(Foo Bar Executable)
```

Foo/CMakeLists.txt

```
ADD_LIBRARY(foo foo1.cxx foo2.cxx)
```

Bar/CMakeLists.txt

```
ADD_LIBRARY(bar bar1.cxx bar2.cxx)
TARGET_LINK_LIBRARIES(bar foo)
```

Executable/CMakeLists.txt

```
ADD_EXECUTABLE(zot zot1.cxx zot2.cxx)
TARGET_LINK_LIBRARIES(zot bar)
```

Przykład 1

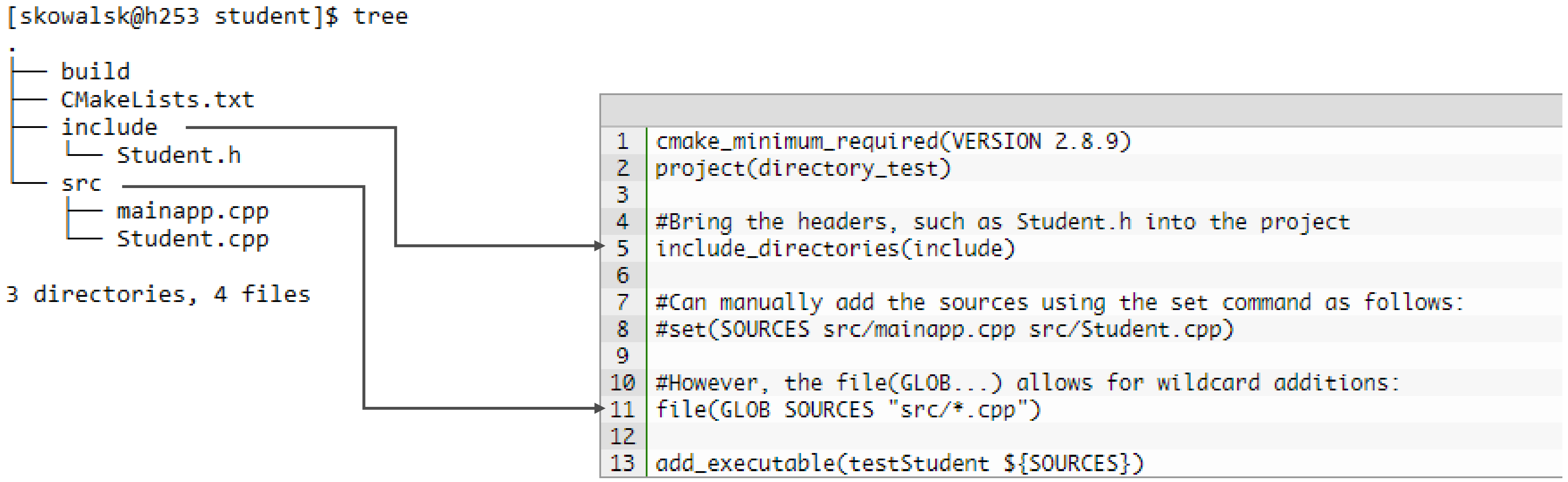
tutorial.cxx

```
// A simple program that computes the square root of a number
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main (int argc, char *argv[])
{
  if (argc < 2)
    {
    fprintf(stdout,"Usage: %s number\n",argv[0]);
    return 1;
    }
  double inputValue = atof(argv[1]);
  double outputValue = sqrt(inputValue);
  fprintf(stdout,"The square root of %g is %g\n",
          inputValue, outputValue);
  return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required (VERSION 2.6)
project (Tutorial)
add_executable(Tutorial tutorial.cxx)
```

Przykład 2

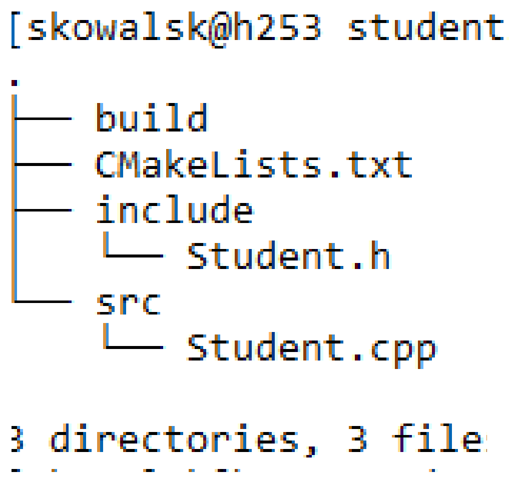


```
[skowalsk@h253 student]$ tree
.
├── build
├── CMakeLists.txt
├── include
│   └── Student.h
└── src
    ├── mainapp.cpp
    └── Student.cpp

3 directories, 4 files
```

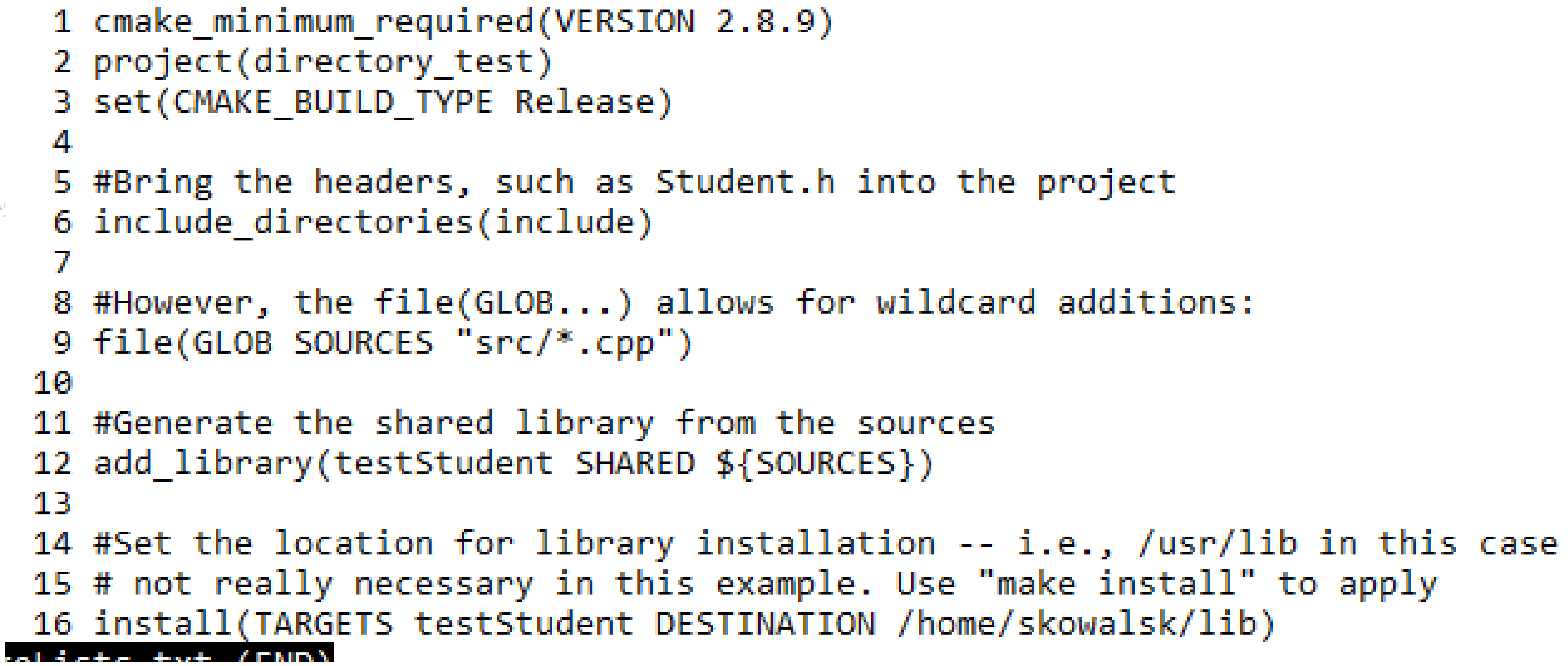
```
cmake_minimum_required(VERSION 2.8.9)
project(directory_test)

#Bring the headers, such as Student.h into the project
include_directories(include)

#Can manually add the sources using the set command as follows:
#set(SOURCES src/mainapp.cpp src/Student.cpp)

#However, the file(GLOB...) allows for wildcard additions:
file(GLOB SOURCES "src/*.cpp")

add_executable(testStudent ${SOURCES})
```

Uwaga – dodanie nowego (kolejnego) pliku źródłowego zawsze wymaga uruchomienia cmake

Przykład 3 – biblioteka dynamiczna

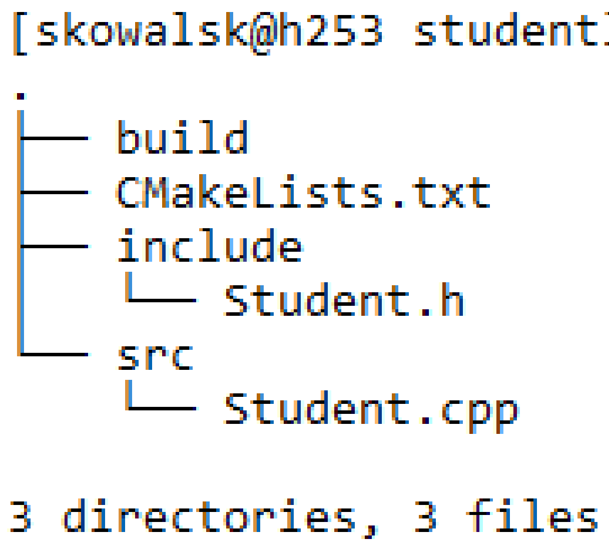


```
[skowalsk@h253 studentlib_shared]$ tree
.
├── build
├── CMakeLists.txt
├── include
│   └── Student.h
└── src
    └── Student.cpp

3 directories, 3 file
```

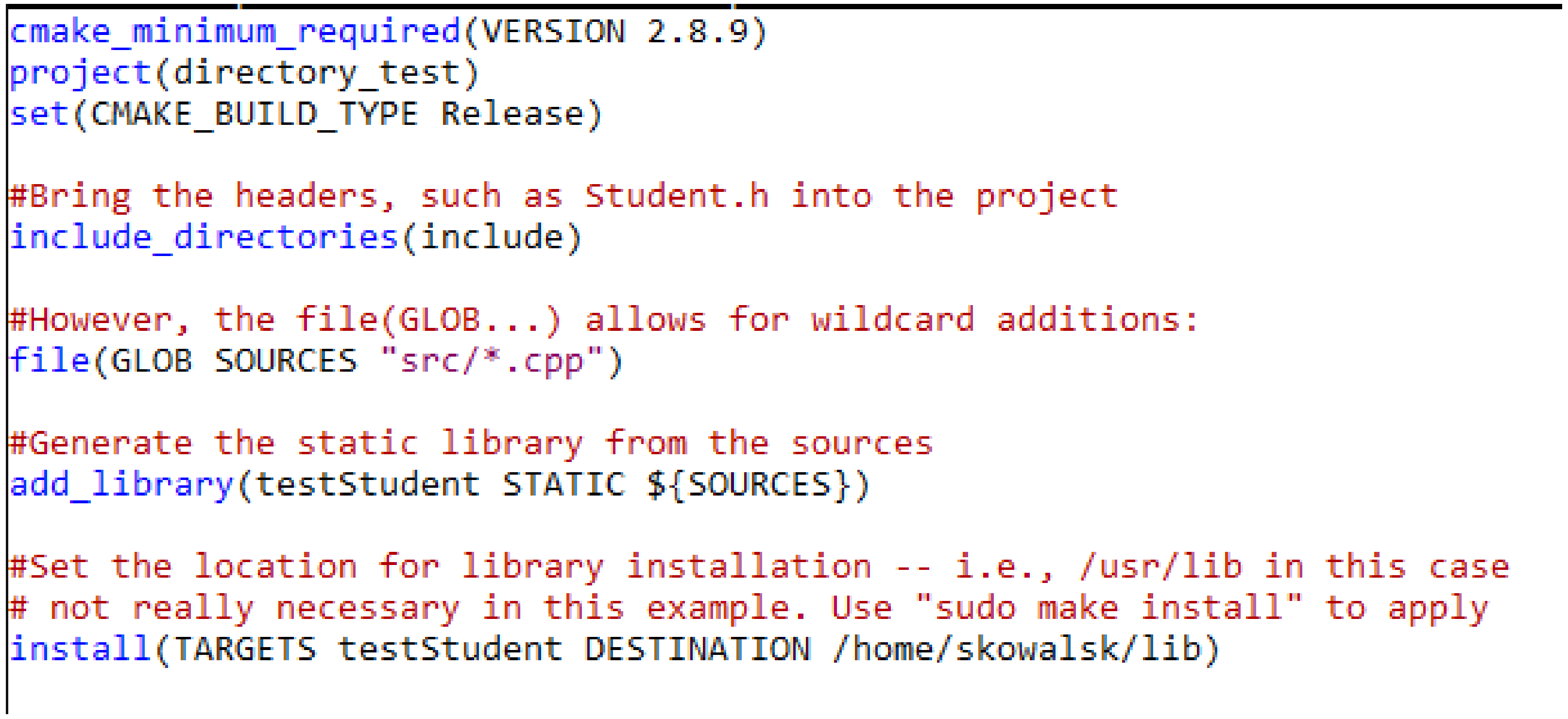


```
cmake_minimum_required(VERSION 2.8.9)
project(directory_test)
set(CMAKE_BUILD_TYPE Release)

#Bring the headers, such as Student.h into the project
include_directories(include)

#However, the file(GLOB...) allows for wildcard additions:
file(GLOB SOURCES "src/*.cpp")

#Generate the shared library from the sources
add_library(testStudent SHARED ${SOURCES})

#Set the location for library installation -- i.e., /usr/lib in this case
# not really necessary in this example. Use "make install" to apply
install(TARGETS testStudent DESTINATION /home/skowalsk/lib)
eLists.txt (END)
```

Przykład 4 – biblioteka statyczna

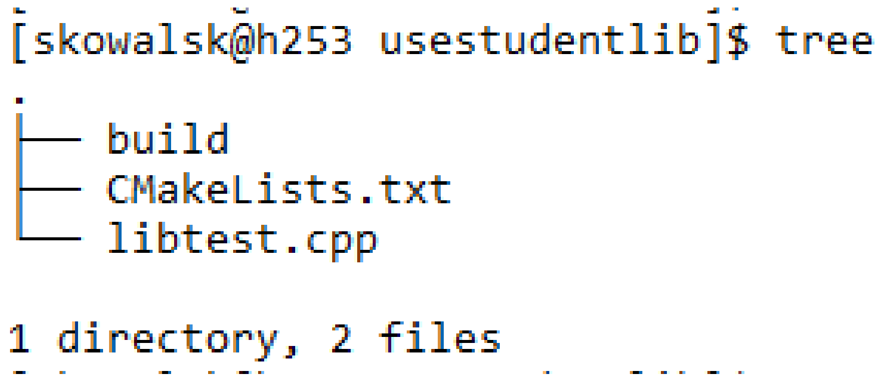


```
[skowalsk@h253 studentlib_static]$ tree
.
├── build
├── CMakeLists.txt
├── include
│   └── Student.h
└── src
    └── Student.cpp

3 directories, 3 files
```

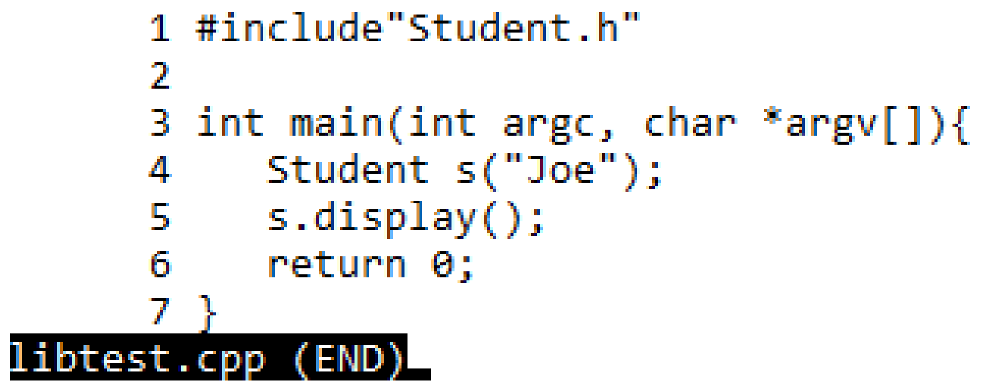


```
cmake_minimum_required(VERSION 2.8.9)
project(directory_test)
set(CMAKE_BUILD_TYPE Release)

#Bring the headers, such as Student.h into the project
include_directories(include)

#However, the file(GLOB...) allows for wildcard additions:
file(GLOB SOURCES "src/*.cpp")

#Generate the static library from the sources
add_library(testStudent STATIC ${SOURCES})

#Set the location for library installation -- i.e., /usr/lib in this case
# not really necessary in this example. Use "sudo make install" to apply
install(TARGETS testStudent DESTINATION /home/skowalsk/lib)
```

Przykład 5 – użycie bibliotek

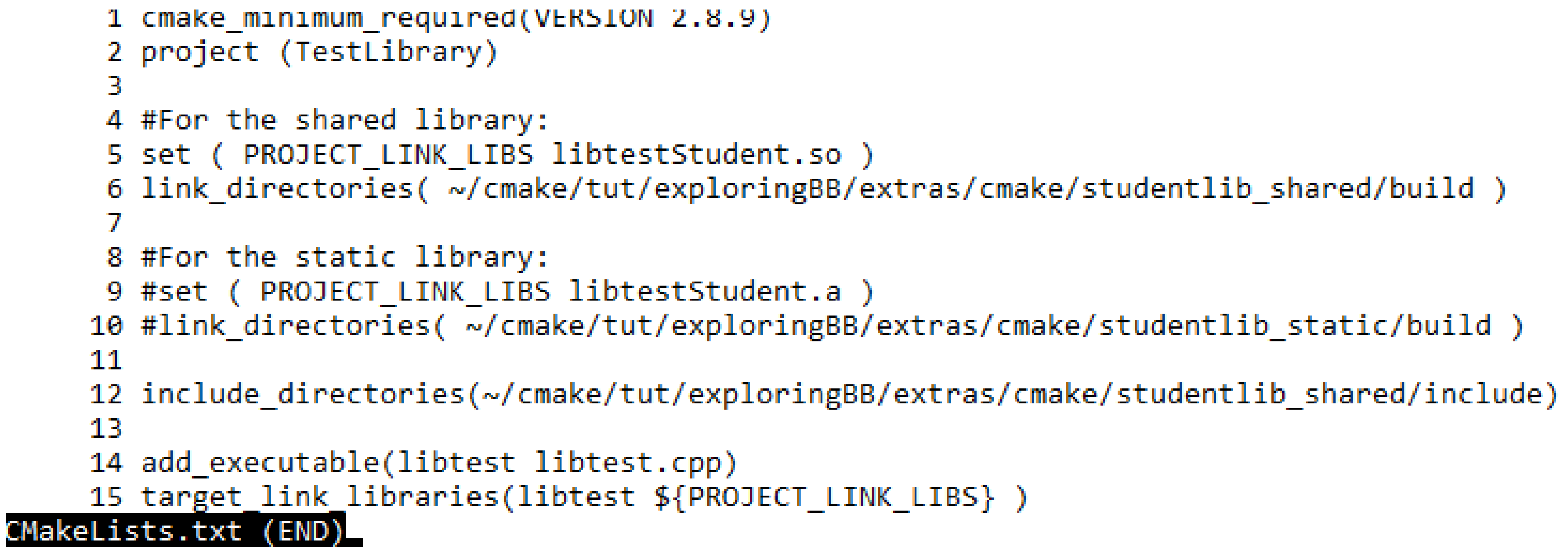


```
[skowalsk@h253 usestudentlib]$ tree
.
├── build
├── CMakeLists.txt
└── libtest.cpp

1 directory, 2 files
```

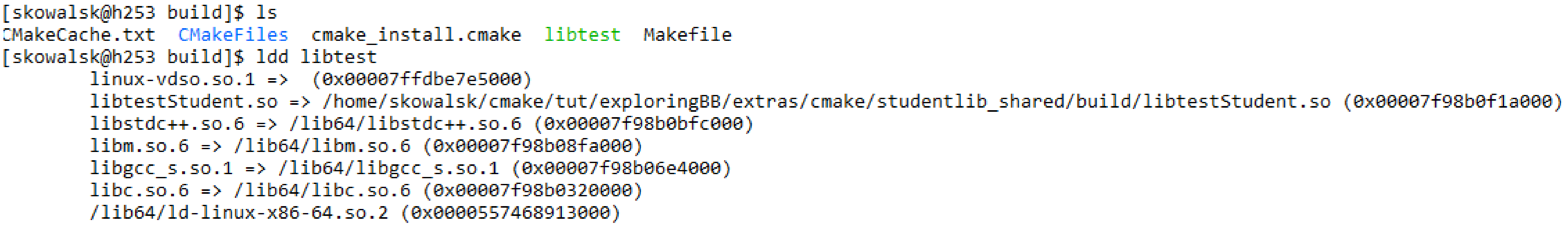


```
#include"Student.h"

int main(int argc, char *argv[]){
   Student s("Joe");
   s.display();
   return 0;
}
libtest.cpp (END)
```

Przykład 5 – użycie bibliotek



```
cmake_minimum_required(VERSION 2.8.9)
project (TestLibrary)

#For the shared library:
set ( PROJECT_LINK_LIBS libtestStudent.so )
link_directories( ~/cmake/tut/exploringBB/extras/cmake/studentlib_shared/build )

#For the static library:
#set ( PROJECT_LINK_LIBS libtestStudent.a )
#link_directories( ~/cmake/tut/exploringBB/extras/cmake/studentlib_static/build )

include_directories(~/cmake/tut/exploringBB/extras/cmake/studentlib_shared/include)

add_executable(libtest libtest.cpp)
target_link_libraries(libtest ${PROJECT_LINK_LIBS} )
CMakeLists.txt (END)
```



```
[skowalsk@h253 build]$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  libtest  Makefile
[skowalsk@h253 build]$ ldd libtest
        linux-vdso.so.1 =>  (0x00007ffdbe7e5000)
        libtestStudent.so => /home/skowalsk/cmake/tut/exploringBB/extras/cmake/studentlib_shared/build/libtestStudent.so (0x00007f98b0f1a000)
        libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f98b0bfc000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f98b08fa000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f98b06e4000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f98b0320000)
        /lib64/ld-linux-x86-64.so.2 (0x0000557468913000)
```