

Uniwersytet Śląski w Katowicach		Wydział Nauk Ścisłych i Technicznych			
		Instytut Fizyki			
		Rok	III	Semestr	V
Kierunek	Informatyka stosowana				
Przedmiot	SiNWO - laboratorium				
Prowadzący	dr Wojciech Gurdziel				
Tytuł ćwiczenia	Wprowadzenie do make i cmake			Nr ćwiczenia	III
Sprawozdanie wykonał: (Imię i Nazwisko)	Jakub Kraus				
Data wykonania ćwiczenia	24.01.2024	Data oddania sprawozdania		31.01.2024	

# Spis treści

<b>1</b>	<b>Cel ćwiczenia</b>	<b>3</b>
<b>2</b>	<b>Przebieg ćwiczenia</b>	<b>3</b>
2.1	Wybór narzędzia . . . . .	3
2.2	Przygotowanie środowiska . . . . .	3
2.3	Przygotowanie plików źródłowych C++ . . . . .	3
2.4	Przygotowanie pliku CMakeLists.txt . . . . .	4
2.5	Generowanie plików budujących . . . . .	4
2.6	Kompilacja programu i uruchomienie . . . . .	5
<b>3</b>	<b>Wnioski</b>	<b>6</b>

# Spis rysunków

1	Zawartość pliku gcd.cpp . . . . .	3
2	Plik CMakeLists.txt . . . . .	4
3	Tworzenie folderu i plików budujących . . . . .	4
4	Utworzenie plików wykonywalnych i uruchomienie programu . . . . .	5

# 1 Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z narzędziami make oraz cmake i ich zastosowaniem w procesie kompilacji programów.

## 2 Przebieg ćwiczenia

Do wykonania ćwiczenia wykorzystałem system operacyjny openSUSE Tumbleweed[2] oraz narzędzie cmake[1].

### 2.1 Wybór narzędzia

W ramach ćwiczenia należało wybrać jedno z narzędzi, które zostaną użyte do kompilacji programów. Wybór padł na narzędzie **cmake**, ponieważ jest ono bardziej rozbudowane i pozwala na automatyzację procesu kompilacji.

### 2.2 Przygotowanie środowiska

W celu przygotowania środowiska należało zainstalować narzędzie cmake. W tym celu należało wykonać polecenie:

```
sudo zypper install cmake
```

oraz kompilator języka C++:

```
sudo zypper install gcc-c++
```

### 2.3 Przygotowanie plików źródłowych C++

W celu przetestowania działania narzędzia cmake przygotowałem pliki źródłowe programu w języku C++.

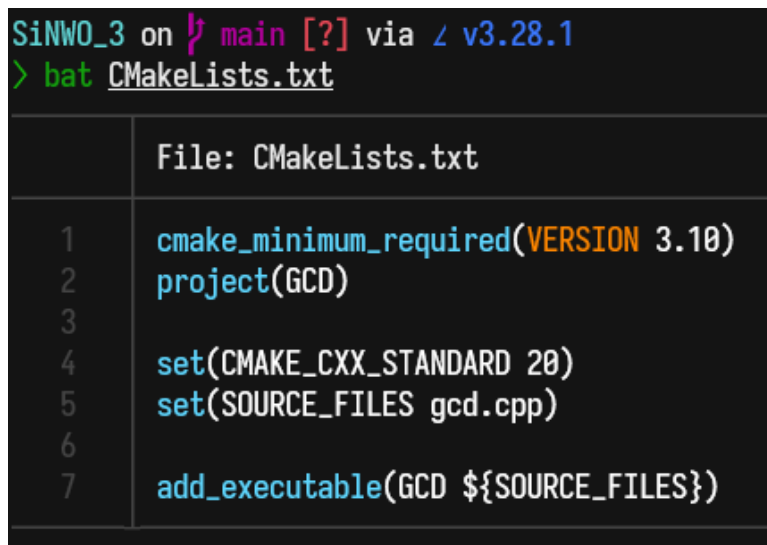


```
SiNWO_3 on main [?]  
> bat gcd.cpp  
File: gcd.cpp  
1  #include <iostream>  
2  
3  int gcd(int a, int b) {  
4      while (b != 0) {  
5          int temp = b;  
6          b = a % b;  
7          a = temp;  
8      }  
9      return a;  
10 }  
11  
12 int main() {  
13     int a, b;  
14     std::cout << "Enter two numbers: ";  
15     std::cin >> a >> b;  
16     std::cout << "The GCD of " << a << " and " << b << " is " << gcd(a, b) << std::endl;  
17     return 0;  
18 }
```

Rysunek 1: Zawartość pliku gcd.cpp

## 2.4 Przygotowanie pliku CMakeLists.txt

W celu kompilacji programu należało przygotować plik CMakeLists.txt.

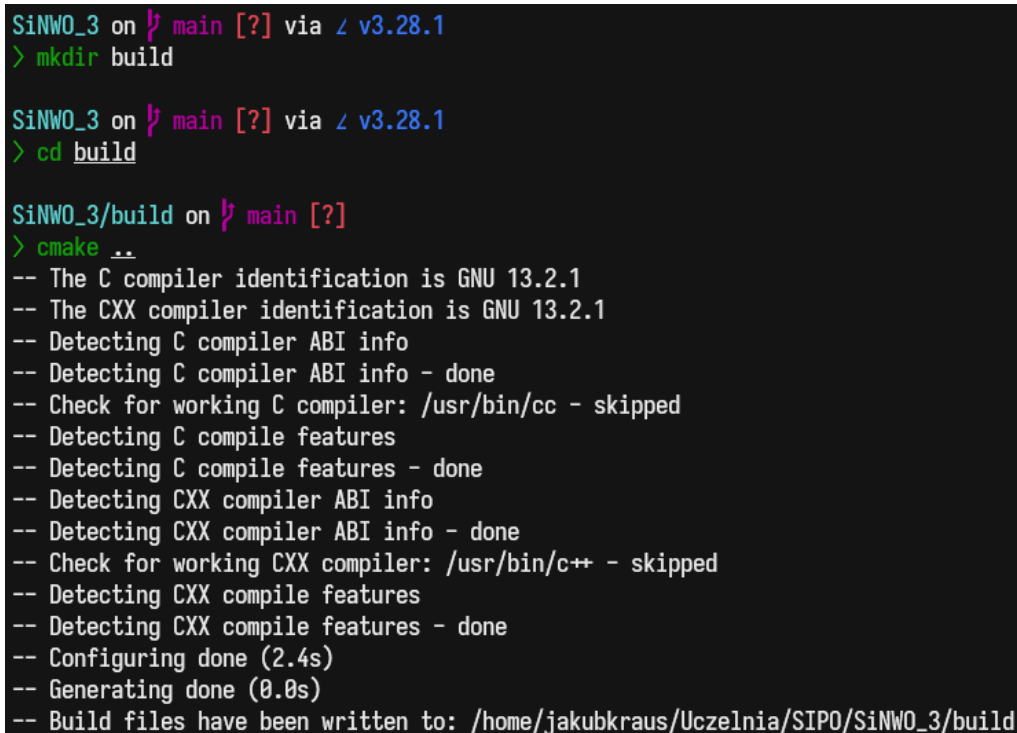


```
SiNWO_3 on main [?] via v3.28.1
> bat CMakeLists.txt
File: CMakeLists.txt
1  cmake_minimum_required(VERSION 3.10)
2  project(GCD)
3
4  set(CMAKE_CXX_STANDARD 20)
5  set(SOURCE_FILES gcd.cpp)
6
7  add_executable(GCD ${SOURCE_FILES})
```

Rysunek 2: Plik CMakeLists.txt

## 2.5 Generowanie plików budujących

W celu wygenerowania plików budujących należało stworzyć folder docelowy i wykonać polecenie:



```
SiNWO_3 on main [?] via v3.28.1
> mkdir build

SiNWO_3 on main [?] via v3.28.1
> cd build

SiNWO_3/build on main [?]
> cmake ..
-- The C compiler identification is GNU 13.2.1
-- The CXX compiler identification is GNU 13.2.1
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/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: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done (2.4s)
-- Generating done (0.0s)
-- Build files have been written to: /home/jakubkraus/Uczelnia/SIPO/SiNWO_3/build
```

Rysunek 3: Tworzenie folderu i plików budujących

## 2.6 Kompilacja programu i uruchomienie

W celu kompilacji programu należało wykonać polecenie:

```
SiNWO_3/build on main [?] via z v3.28.1  
> cmake --build .  
[ 50%] Building CXX object CMakeFiles/GCD.dir/gcd.cpp.o  
[100%] Linking CXX executable GCD  
[100%] Built target GCD  
  
SiNWO_3/build on main [?] via z v3.28.1  
> ls -l  
total 56  
-rw-r--r-- 1 jakubkraus jakubkraus 14253 Jan 31 22:22 CMakeCache.txt  
drwxr-xr-x 6 jakubkraus jakubkraus  4096 Jan 31 22:22 CMakeFiles  
-rw-r--r-- 1 jakubkraus jakubkraus  1652 Jan 31 22:22 cmake_install.cmake  
-rwxr-xr-x 1 jakubkraus jakubkraus 20752 Jan 31 22:22 GCD  
-rw-r--r-- 1 jakubkraus jakubkraus  5094 Jan 31 22:22 Makefile
```

(a) Cmake build i zawartość katalogu build

```
SiNWO_3/build on main [?] via z v3.28.1  
> ./GCD  
Enter two numbers: 33  
14  
The GCD of 33 and 14 is 1
```

(b) Uruchomienie programu

Rysunek 4: Utworzenie plików wykonywalnych i uruchomienie programu

### 3 Wnioski

CMake dostarcza narzędzie do konfiguracji, kompilacji i instalacji projektów w sposób niezależny od platformy. Możliwość definiowania projektów w sposób zależny od platformy pozwala na przenośność kodu między różnymi systemami operacyjnymi. Dzięki CMake i narzędziom kompilacyjnym, takim jak Make, możliwa jest automatyzacja procesu kompilacji. Po skonfigurowaniu projektu za pomocą CMake, polecenie

```
cmake --build .
```

pozwalą na skompilowanie projektu bez konieczności ręcznego zarządzania szczegółami kompilacji. CMake umożliwia integrację testów w proces budowy projektu za pomocą polecenia `ctest`. Ponadto, opcja

```
cmake --install .
```

pozwalą na instalację skompilowanego projektu na systemie, co może być przydatne w przypadku rozpowszechniania gotowych aplikacji.

Podsumowując, CMake stanowi skuteczne narzędzie do zarządzania procesem kompilacji, które oferuje wiele możliwości konfiguracyjnych i ułatwia tworzenie projektów w języku C++ (a także w innych językach programowania).

### Bibliografia

- [1] *CMake - Upgrade Your Software Build System*. en-US. URL: <https://cmake.org/> (term. wiz. 24.01.2024).
- [2] *openSUSE Tumbleweed*. en. URL: <https://get.opensuse.org/tumbleweed.html> (term. wiz. 24.01.2024).