Современные языки и платформы программирования

Лекторы:
Аспирант МФТИ, Шер Артём Владимирович
Аспирант МФТИ, Зингеренко Михаил Владимирович

10 сентября 2024

План курса

- CMake
- GTest / DocTest
- C++ Basics (Templates, Classes, Virtualization)
- Parallel Computing
- Docker
- CI/CD

Как наш курс будет работать

Мы начнём с игрушечного примера и маленькой функции и будем постепенно наращивать вокруг неё то, что мы вам будем рассказывать.

Каждый студент выберет для себя функцию из диплома или функцию обработки изображений или функцию из линейной алгебры.

Оценка за курс будет формироваться по результатам работы над этим проектом.

Группа Курса

Компиляция простого кода

```
1 g++ main.cpp functions.cpp special_functions.cpp -o
      awsome_program
```

Make: когда файлов побольше

```
1 all: awsome_program
2 awsome_program: useful_functions.o
3     g++ main.cpp
4 useful_functions.o:
5     g++ -c utilities.cpp stuff_that_should_be_in_opencv.cpp
      -L/usr/lib/opencv_core
6 clean:
7     rm *.o awsome_program
```

Больше Make

```
1  # target to preprocess a source file
2  dependencies/include/TinyXML2/tinyxml2.cpp.i:
3    $(MAKE) -f CMakeFiles/main.dir/build.make CMakeFiles/main.
       dir/dependencies/include/TinyXML2/tinyxml2.cpp.i
4  .PHONY : dependencies/include/TinyXML2/tinyxml2.cpp.i
5
6  dependencies/include/TinyXML2/tinyxml2.s: dependencies/
       include/TinyXML2/tinyxml2.cpp.s
7
8  .PHONY : dependencies/include/TinyXML2/tinyxml2.s
9
10 # target to generate assembly for a file
11 dependencies/include/TinyXML2/tinyxml2.cpp.s:
12   $(MAKE) -f CMakeFiles/main.dir/build.make CMakeFiles/main.
       dir/dependencies/include/TinyXML2/tinyxml2.cpp.s
13 .PHONY : dependencies/include/TinyXML2/tinyxml2.cpp.s
14
15 drawcall.o: drawcall.cpp.o
16
17 .PHONY : drawcall.o
18
19 # target to build an object file
```

Cmake: зачем он нужен и с чем его едят

Cmake это кросс-платформенная утилита предназначенная для сборки и автоматизации проекта и позволяет автоматически генерировать make. Cmake осуществляет контроль за зависимостями, другими библиотеками, исходном кодом, тестированием и др.

Cmake использует файл конфигурации CMakeLists.txt (лежащий в директории проекта) для генерации MakeFile.

Этапы сборки

- Конфигурация: выбор платформы, опции, пакеты. Проверка кэша в CMakeCache.txt
- Генерация
- Сборка
- Установка
- Тестирование

Типы сборок

Debug	-O0 -g
RelWithDebInfo	-O2 -g -DNDEBUG
Release	-O3 -DNDEBUG
MinSizeRel	-Os -DNDEBUG
Собственный	Ваши ключи сборки

Создание проекта с CMake

```
1 # CMakeLists.txt
2 # Set the minimum required version of CMake for this project
      .
3 # Error will be raised if version is too low.
4 cmake_minimum_required(VERSION 3.10 FATAL_ERROR)
5 # Give the project a name.
6 project(hello_cmake)
```

Базовая конфигурация

```
1 cmake_minimum_required(VERSION 3.10 FATAL_ERROR)
2 project(hello_cmake)
3 # Define an executable
4 add_executable(
5   # Executable name
6   hello_cmake
7   # Files required to compile
8   main.cpp other.cpp
9 )
```

Базовая конфигурация: добавление библиотеки

```
1  cmake_minimum_required(VERSION 3.10 FATAL_ERROR)
2  project(hello_cmake)
3  # Define a library
4  add_library(
5    # Library name
6    hello_cmake_library
7    # Can be SHARED (.dll/.so) or STATIC (.lib/.a)
8    SHARED
9    # Files required to compile
10   hello_cmake_lib.cpp other.cpp
11 )
```

Переменные

```
1  cmake_minimum_required(VERSION 3.10 FATAL_ERROR)
2  project(hello_cmake)
3  file(
4    # GLOB to match a regular expression
5    GLOB
6    # Name of output variable
7    LIBRARY_SOURCE_FILES
8    # Regular expression pattern
9    lib/*.cpp
10 )
11 # Variables values are accessed by putting their names in ${}
12 add_library(hello_cmake_library SHARED ${LIBRARY_SOURCE_FILES}
```

Линковка

```
1  file(GLOB LIBRARY_SOURCE_FILES lib/*.cpp)
2  add_library(hello_cmake_library SHARED ${
       LIBRARY_SOURCE_FILES}
3
4  file(GLOB APP_SOURCE_FILES app/*.cpp)
5  add_executable(hello_cmake ${APP_SOURCE_FILES})
6
7  # Declare hello_cmake_library as a dependency and link with
       it
8  target_link_libraries(
9    # Target in question: the executable
10   hello_cmake
11   # Inheritance (PRIVATE, PUBLIC or INTERFACE)
12   PRIVATE # targets depending on this will not inherit
13   hello_cmake_library
14 )
```

Наследование

```
1  add_library(hello_cmake_library SHARED ${
       LIBRARY_SOURCE_FILES})
2  # Set different include directories for library and clients
3  target_include_directories (
4    hello_cmake_library
5    PRIVATE # Applies to this target only
6    lib/include/hello_cmake_library
7    INTERFACE # Applies to clients but not target
8    lib/include # clients forced to #include <
       hello_cmake_library>
9    PUBLIC # Applies to both target and clients
10   ${OTHER_LIB_INCLUDE_DIRECTORIES}
11 )
12 add_executable(hello_cmake ${APP_SOURCE_FILES})
13
14 # Linking with library inherits include directories
15 target_link_libraries(hello_cmake PRIVATE
       hello_cmake_library)
```

До следующей лекции!