Получение команд компиляций и компоновок для проектов на C/C++

Автор: Мирзазянов Глеб Романович, 18.Б11-мм
Научный руководитель: ст. преп. кафедры системного программирования Е.К. Куликов
Консультант: разработчик Huawei Technologies Ltd В.Е. Володин

Санкт-Петербургский государственный университет
Кафедра системного программирования

12 июня 2021г.

База команд компиляций и компоновок

- инструменты синтаксического анализа кода
- генерация и запуск тестов
- среды разработки
- отладка систем сборки
- команды компоновки:
  - разрешения конфликтов при наличии функций и методов с одинаковыми именами и сигнатурой; глобальными переменными

Проекты использующие базу компиляций или компоновок

- Clang Static Analyzer
- UnitTestBot
- Codechecker. Обсуждение
- Code comprehension. Обсуждение
- Android Native Development Kit

Конфликт функций

CMakeLists.txt | maina.cpp | a.cpp

```cpp
void multiply_defined_function();

int main() {
    multiply_defined_function();
    return 0;
}
```

CMakeLists.txt | maina.cpp | a.cpp

```cpp
#include <iostream>

void multiply_defined_function() {
    std::cout << "a.cpp" << std::endl;
}
```

CMakeLists.txt | mainb.cpp | b.cpp

```cpp
void multiply_defined_function();

int main() {
    multiply_defined_function();
    return 0;
}
```

CMakeLists.txt | mainb.cpp | b.cpp

```cpp
#include <iostream>

void multiply_defined_function() {
    std::cout << "b.cpp" << std::endl;
}
```

CMakeLists.txt | maina.cpp | a.cpp | mainb.cpp | b.cpp

```
cmake_minimum_required(VERSION 3.19)
project(ex1)

set(CMAKE_CXX_STANDARD 14)

add_executable(maina maina.cpp a.cpp)
add_executable(mainb mainb.cpp b.cpp)
```

Рис.: Пример конфликта. Опция CLion "показать определение" работает некорректно: в двух случаях показывает реализацию функции в файле a.cpp

Пример файла

```
[
{
  "directory": "/home/gleb/Documents/research/example/build",
  "command": "/usr/bin/c++   -Wall -Werror -Wextra -pedantic -std=gnu++14 -o CMakeFile
  "file": "/home/gleb/Documents/research/example/a.cpp"
},
{
  "directory": "/home/gleb/Documents/research/example/build",
  "command": "/usr/bin/c++  -Wall -Werror -Wextra -pedantic -std=gnu++14 -o CMakeFile
  "file": "/home/gleb/Documents/research/example/c.cpp"
},
```

```
[
{
  "directory": "/home/gleb/CLionProjects/example/build",
  "command": "/usr/bin/c++ -Wl,-z,defs CMakeFiles/mainb.dir/mainb.cpp.o CMakeFiles/mai
  "files": ["/home/gleb/CLionProjects/example/build/CMakeFiles/mainb.dir/mainb.cpp.o",
},
{
  "directory": "/home/gleb/CLionProjects/example/build",
  "command": "/usr/bin/ar qc libsay-c.a CMakeFiles/say-c.dir/c.cpp.o",
  "files": ["/home/gleb/CLionProjects/example/build/CMakeFiles/say-c.dir/c.cpp.o"]
},
```

Рис.: Пример команд компиляции и компоновок записанных в формате JSON

Цель и задачи

Цель работы — анализ существующих систем сборок и инструментов. Реализация команд компиляций и компоновок

Задачи:

- Анализ существующих систем сборок
- Обзор инструментов генерации команд компоновок и компиляции
- Реализация возможности экспорта команд компиляций или компоновок при необходимости
- Постановка и проведения экспериментов с получившимися результатами
- Внедрение реализованной функциональности внутри компании Huawei
- По возможности внедрение результатов в проект с открытым исходным кодом *CMake*

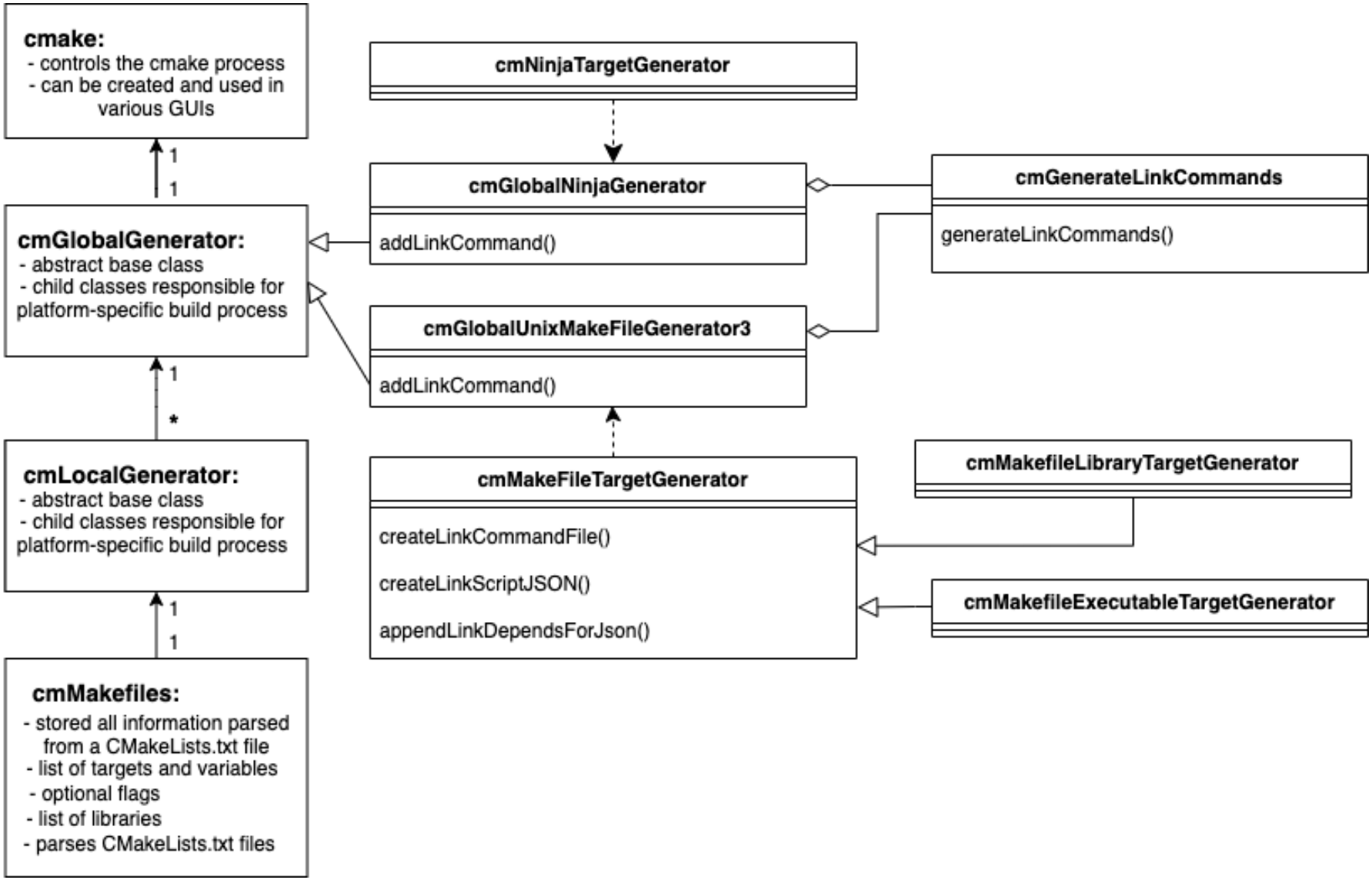
Инструменты и системы сборки

- Инструменты: Bear, Compilation database generator, Scan-build
  - Для использования требуется полная сборка проекта
  - Нет возможности экспорта команд компоновки
- Системы сборки: Bazel, SCons, QMake, Autotools, Visual Studio Project
  - Непопулярные относительно CMake
  - Нет возможности экспорта команд компоновки
- CMake
  - Реализован экспорт команд компиляции
  - Кроссплатформенность
  - Открытый исходный код
  - Нет необходимости собирать проект для получение команд компоновок и компиляции

Детали реализации

- Добавлен новый класс LinkDatabaseGenerator, позволяющий абстрагироваться от внутренней системы сборки
- С помощью существующих функций CMake реализован кроссплафторфменный инструмент создания базы компоновки
- Создана возможность использовать инструмент, указав опцию в CMakeLists.txt или прописав флаг при запуске через консоль
- Допускается получение команд компоновок в зависимости от версии формата

CMake: Архитектура



Эксперименты: база команд компоновок

Название проекта	Строчек кода	Исходных файлов	Успешная сборка
googletest	112,559	261	Да
CMake	151,950	18117	Да
corecvs	529,341	2449	Да
xgboost	292,401	1244	Да
openmp	171,876	447	Да
spdlog	41,207	149	Да
cmocka	18,295	177	Да
tiny-dnn	264863	782	Да
seasocks	25398	127	Да
Bear	19910	247	Да
llvm	1,589,859	46376	Да*
opencv	1,723,774	6677	Да*

Таблица: Результаты тестирования

Под успешной сборкой подразумевается создание всех таргетов и библиотек с помощью команд компоновок

Результаты

- Проведен обзор существующих популярных систем сборок для языка C/C++, позволяющих получить команды компиляции
- Совместно с консультантом разработал стандарт формата команд компоновки и отправил его на рассмотрение в сообщество LLVM
- Реализована возможность экспорта команд компоновок для *CMake*
- Разработал тесты для оценки качества инструмента и проверил корректность на 12 проектах, в частности OpenCV и LLVM
- Добился внедрения реализованной функциональности внутри компании Huawei
- Сделан Pull Request в *CMake*. После обсуждения с разработчиками CMake исправил все глобальные проблемы (версионирование и поддержка сборки с Ninja). На данный момент *PullRequest* в процессе правок после ревью