

Описание программы: calculation_hash

1. Среда разработки.

Оборудование:

description: Computer

*-memory

size: 7859MiB

*-cpu

product: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz

width: 64 bits

*-disk

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1		1	500130303	500130303	238.5G	ee	GPT

ОС:

Ubuntu 18.04.4 LTS [linux version 4.15.0-130-generic]

IDE:

Qt Creator 4.11.0 (Based on Qt 5.14.0 (GCC 5.3.1 20160406 (Red Hat 5.3.1-6), 64 bit))

Компилятор и cmake:

gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0

cmake version 3.10.2

2. Стиль кода.

В качестве автоматического контроля стиля форматирования файлов CMakeLists.txt используется <https://github.com/polysquare/polysquare-cmake-linter>.

Практика кодирования на C++:

<https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md>

Стиль кодирования кода: <https://google.github.io/styleguide/cppguide.html>

Стандарты и библиотеки:

C++11, C++17, boost

3. О программе.

Консольная программа на C++ для генерации сигнатуры указанного файла. Сигнатура генерируется следующим образом: исходный файл делится на блоки равной (фиксированной) длины (если размер файла не кратен размеру блока, последний фрагмент считается для меньшего размера). По умолчанию размер блока 1 Мб. Для каждого блока вычисляется значение crc32 с помощью boost библиотеки и дописывается в выходной файл-сигнатуру.

Интерфейс: программы: командная строка, в которой указаны:

- Путь до входного файла
- Путь до выходной файла
- Размер блока

В программе:

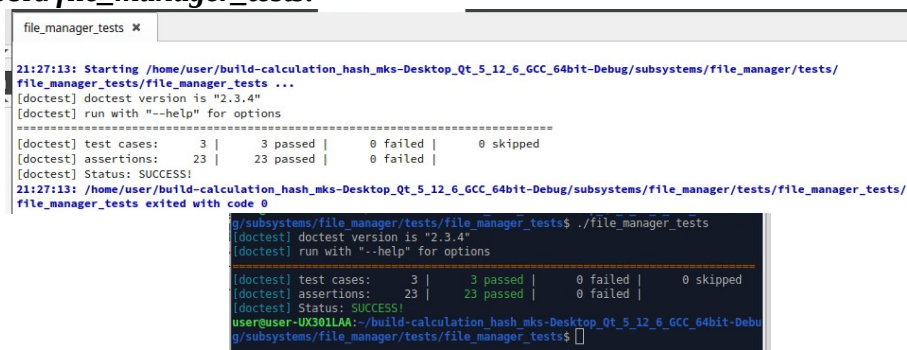
- Оптимизирована скорость работы программы с учетом работы в многопроцессорной среде (используется треды boost библиотеки).
- Оптимизирована скорость работы программы за счет использования маппинга для чтения файла через boost::interprocess (*хранится указатель на данные, для сверх больших файлов >100Гб можно дополнительно добавить чтение окнами по заданному размеру*).
- Реализован класс эксепшенов, который позволяет ловить ошибки основного функционала, также эксепшены boost::interprocess выводятся в работе по чтению файла в map.
- Добавлены assert после аллоцирования памяти, чтобы на этапе разработки уже видеть какие-то проблемы с памятью.
- Применяются при работе с ресурсами умные указатели boost библиотеки.
- Не используются сторонние библиотеки OpenMP, OpenCL, etc. (*возможно использовать Intel Threading Building Blocks в перспективе*)
- По коду программы применяются разные приемы C++: шаблоны, стандарты, лямбда функции и т.п.
- В программе написаны тесты для основного функционала классов через doctest.

4. Описание тестов.

Тесты можно запускать из Qtcreator или из исполняемого файла.

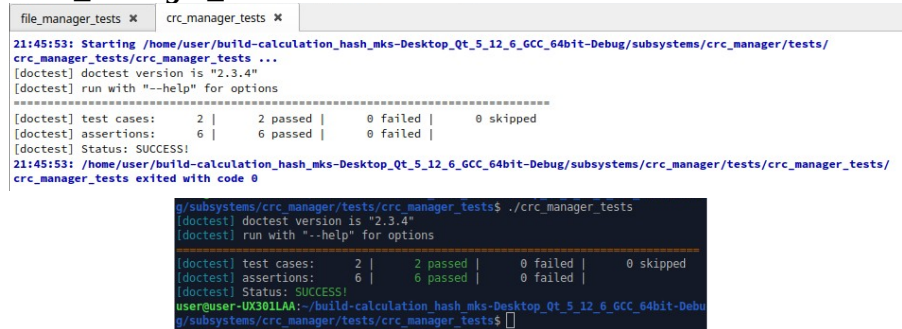
Для запуска теста нужно скопировать файл с именем **"test_file_for_read"** по пути **"/tmp/test_directory/"** (сейчас при сборке stake файл и директория добавляются автоматически). Этот файл должен быть одним по заданному пути для запуска какого-либо из тестов (скрипты очистки созданных файлов в заданной директории не разрабатывались, чтобы сгенерируемые тестами файлы можно посмотреть «вручную»). Файл на запись программа пишет в конец файла, поэтому для запуска нового теста, нужно удалить все созданные файлы, кроме исходного).

Скрин теста **file_manager_tests**:



```
file_manager_tests ✖
21:27:13: Starting /home/user/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/file_manager/tests/
file_manager_tests/file_manager_tests ...
[doctest] doctest version is "2.3.4"
[doctest] run with "--help" for options
=====
[doctest] test cases:      3 |      3 passed |      0 failed |      0 skipped
[doctest] assertions:    23 |     23 passed |      0 failed |
[doctest] Status: SUCCESS!
21:27:13: /home/user/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/file_manager/tests/file_manager_tests/
file_manager_tests exited with code 0
g:/subsystems/file_manager/tests/file_manager_tests$ ./file_manager_tests
[doctest] doctest version is "2.3.4"
[doctest] run with "--help" for options
=====
[doctest] test cases:      3 |      3 passed |      0 failed |      0 skipped
[doctest] assertions:    23 |     23 passed |      0 failed |
[doctest] Status: SUCCESS!
user@user-UX301LAA:~/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debu
n/subsystems/file_manager/tests/file_manager_tests$
```

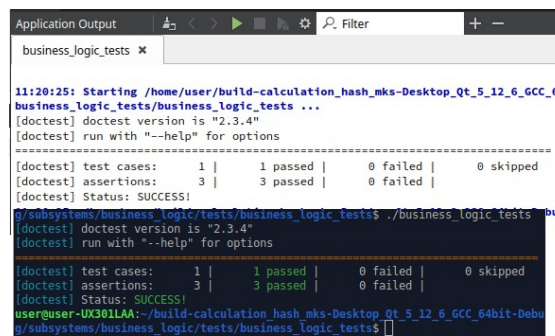
Скрин теста *crc_manager_tests*:



```
file_manager_tests x crc_manager_tests x
21:45:53: Starting /home/user/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/crc_manager/tests/
crc_manager_tests/crc_manager_tests ...
[doctest] doctest version is "2.3.4"
[doctest] run with "--help" for options
=====
[doctest] test cases: 2 | 2 passed | 0 failed | 0 skipped
[doctest] assertions: 6 | 6 passed | 0 failed |
[doctest] Status: SUCCESS!
21:45:53: /home/user/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/crc_manager/tests/crc_manager_tests/
crc_manager_tests exited with code 0

g/subsystems/crc_manager/tests/crc_manager_tests$ ./crc_manager_tests
[doctest] doctest version is "2.3.4"
[doctest] run with "--help" for options
=====
[doctest] test cases: 2 | 2 passed | 0 failed | 0 skipped
[doctest] assertions: 6 | 6 passed | 0 failed |
[doctest] Status: SUCCESS!
user@user-UX301LAA:~/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/crc_manager/tests/crc_manager_tests$
```

Скрин теста *business_logic_tests*:



```
Application Output
business_logic_tests x
11:20:25: Starting /home/user/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/business_logic/tests/business_logic_tests ...
[doctest] doctest version is "2.3.4"
[doctest] run with "--help" for options
=====
[doctest] test cases: 1 | 1 passed | 0 failed | 0 skipped
[doctest] assertions: 3 | 3 passed | 0 failed |
[doctest] Status: SUCCESS!
g/subsystems/business_logic/tests/business_logic_tests$ ./business_logic_tests
[doctest] doctest version is "2.3.4"
[doctest] run with "--help" for options
=====
[doctest] test cases: 1 | 1 passed | 0 failed | 0 skipped
[doctest] assertions: 3 | 3 passed | 0 failed |
[doctest] Status: SUCCESS!
user@user-UX301LAA:~/build-calculation_hash_mks-Desktop_Qt_5_12_6_GCC_64bit-Debug/subsystems/business_logic/tests/business_logic_tests$
```

Скрин программы *calculation_hash_mks*:

1. Ввод данных.

В тестовой директории ***"/tmp/test_directory/"*** находится файл для чтения ***"test_file_for_read"***.

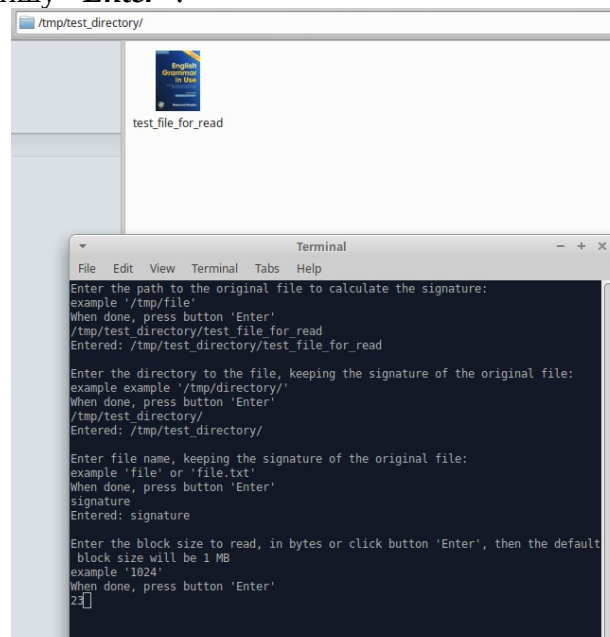
1.1 Вводим данные для пути к файлу теста: ***"/tmp/test_directory/test_file_for_read"***.

1.2 Вводим директорию для сохранения файла с подсчитанной сигнатурой: ***"/tmp/test_directory/"***.

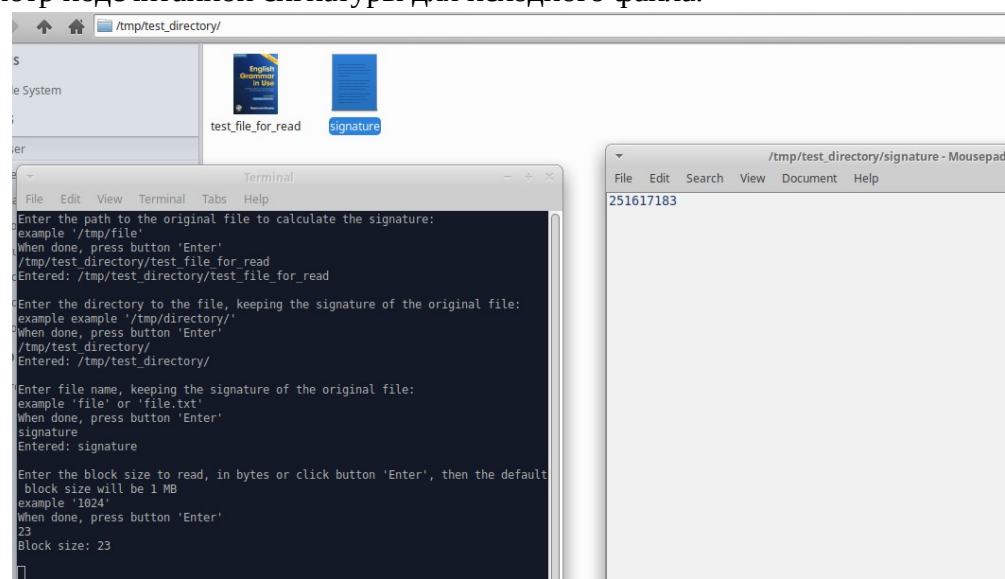
1.3 Вводим имя файла для сохранения подсчитанных сигнатур: ***"signature"***.

1.4 Вводим размер блока: ***"23"***.

1.5 Нажимаем клавишу ***"Enter"***.



2. Просмотр подсчитанной сигнатуры для исходного файла.



3. Завершение программы.

Нажатие на "x".