gauze Documentation

Release 0.1

Ruslan Baratov, David Hirvonen

May 30, 2019

Prerequirements

Gauze framework provides unified C++/CMake interface for testing on iOS/Android/Linux/OSX/Windows platforms. CMake function `gauze_add_tests` wraps standard add_test and in case of testing on host (Linux on Linux, OSX on OSX, etc.) just forwards `NAME` and `COMMAND` arguments. Main functionality of the framework is uploading/starting binaries for iOS/Android testing. CMake style generator expressions `$<GAUZE_RESOURCE_FILE:...>` and `$<GAUZE_RESOURCE_DIR:...>` can be used for managing resources.

Hunter

- gauze

CHAPTER 1

iOS

`ios-deploy` tool will be used for uploading and running executables.

Building from source:

```
> git clone https://github.com/phonegap/ios-deploy
> cd ios-deploy
[ios-deploy]> xcodebuild
[ios-deploy]> ls build/Release/ios-deploy
build/Release/ios-deploy
[ios-deploy]> export PATH=`pwd`/build/Release:$PATH
[ios-deploy]> which ios-deploy
/.../ios-deploy/build/Release/ios-deploy
```

Note:

- https://github.com/phonegap/ios-deploy

Warning: Currently only testing on device is supported (no simulator testing)

CHAPTER 2

Android

By default Gauze will run tests on real device connected to host. To check that device is connected successfully run `adb devices`:

```
> adb devices
List of devices attached
<device-name> device
```

To use emulator add `GAUZE_ANDROID_USE_EMULATOR=ON`. In this case AVD will be created automatically and emulator will be launched:

```
Creating AVD with name 'gauze_android-19_armeabi-v7a'
Starting emulator
Waiting for emulator
Emulator is ready!
```

Created AVD can be found in list of available AVDs:

```
> /.../android-sdk/tools/android list avd
    Name: gauze_android-19_armeabi-v7a
    Path: /.../.android/avd/gauze_android-19_armeabi-v7a.avd
  Target: Android 4.4.2 (API level 19)
 Tag/ABI: default/armeabi-v7a
    Skin: WVGA800
```

Emulator is visible by the same `adb device` command:

```
> /.../android-sdk/platform-tools/adb devices
List of devices attached
emulator-5678   device
<device-name>   device
```

Deleting AVD:

```
> /.../android-sdk/tools/android delete avd --name gauze_android-19_armeabi-v7a
Deleting file /.../.android/avd/gauze_android-19_armeabi-v7a.ini
Deleting folder /.../.android/avd/gauze_android-19_armeabi-v7a.avd

AVD 'gauze_android-19_armeabi-v7a' deleted.
```

Emulator can be stopped by `kill -9` command:

```
> ps aux | grep gauze_android
<username> 9160 ... /.../android-sdk/tools/emulator64-arm -avd gauze_android-19_
↪armeabi-v7a -no-window -port 5678 -gpu host
> kill -9 9160
```

Warning: Pushed resource file and executable are not cleaned up automatically. You have to remove files explicitly from `GAUZE_ANDROID_DEVICE_TESTING_ROOT` directory (defaults to `/data/local/tmp`):

```
> adb shell
> cd /data/local/tmp
> ls -la
> rm -rf somefile somedir
```

2.1 Other options

2.1.1 GAUZE_ANDROID_EMULATOR_GPU

You can set `GAUZE_ANDROID_EMULATOR_GPU` to control what GPU type will be using while creating Android emulator:

- https://developer.android.com/studio/run/emulator-acceleration.html#command-gpu

---

Hint: For Travis CI:

- use `host` on macOS machines
- use `off` on Linux machines

2.1.2 GAUZE_ANDROID_EMULATOR_PARTITION_SIZE

You can set `GAUZE_ANDROID_EMULATOR_PARTITION_SIZE` to specify the system data partition size in MBs.

2.1.3 GAUZE_ANDROID_PUSH_QUIET

Set `GAUZE_ANDROID_PUSH_QUIET` to `ON` to suppress output from `adb push` commands.

2.1.4 GAUZE_ANDROID_PUSH_RESOURCES_ONLY

Do not run tests but only push resources.

2.1.5 GAUZE_ANDROID_START_EMULATOR

Gauze will create suitable Android emulator image and start emulator automatically. In case if you want to reuse existing emulator instead, e.g. one created manually in Android Studio, you can set `GAUZE_ANDROID_START_EMULATOR` option to `OFF`. `GAUZE_ANDROID_START_EMULATOR` is set to `ON` by default.

	Real device	Emulator (automatically)	Emulator (external)
adb shell [1]	`adb -d shell`	`adb -e shell`	`adb -e shell`
GAUZE_ANDROID_USE_EMULATOR	`OFF` [2]	`ON`	`ON`
GAUZE_ANDROID_START_EMULATOR	–	`ON` [3]	`OFF`

[1] https://developer.android.com/studio/command-line/adb#issuingcommands
[2] By default real device used
[3] By default Gauze will start emulator automatically

CHAPTER 3

Simple

Simple example, without arguments on launch:

```
add_executable(gauze_simple main.cpp)

gauze_add_test(NAME gauze_simple COMMAND gauze_simple)
```

User should define `int gauze_main(int argc,char** argv)` function instead of `main`:

```
#include <iostream> // std::cout
#include <cstdlib> // EXIT_SUCCESS

int gauze_main(int argc, char** argv) {
  std::cout << "Hello, Gauze!" << std::endl;
  return EXIT_SUCCESS;
}
```

Warning: While migrating to Gauze framework note that `gauze_main` unlike `main` doesn't have default return value - you should return explicit `int` from function.

CHAPTER 4

With arguments

Launch with arguments:

```
add_executable(gauze_args main.cpp)

gauze_add_test(NAME gauze_args COMMAND gauze_args arg1 arg2 arg3)
```

```
#include <iostream> // std::cout
#include <cstdlib> // EXIT_SUCCESS

int gauze_main(int argc, char** argv) {
  std::cout << "argc = " << argc << std::endl;
  for (int i=0; i<argc; ++i) {
    std::cout << "argv[" << i << "] = " << argv[i] << std::endl;
  }

  return EXIT_SUCCESS;
}
```

CHAPTER 5

With resources

If we need some resource file while testing it can be uploaded by adding `$<GAUZE_RESOURCE_FILE:...>` directive:

```
add_executable(gauze_resource main.cpp)

set(data_dir "${CMAKE_CURRENT_LIST_DIR}/data")

gauze_add_test(
    NAME gauze_resource
    COMMAND
    gauze_resource
    arg1
    arg2
    arg3
    $<GAUZE_RESOURCE_FILE:${data_dir}/input.txt>
    ${data_dir}/just_a_string.txt
    --input=$<GAUZE_RESOURCE_FILE:${data_dir}/input.txt>
)
```

Files specified with `GAUZE_RESOURCE_FILE` will be uploaded to device and path will be substituted with real path **on device**. Note that similar string without `GAUZE_RESOURCE_FILE` will be used as is:

```
add_executable(gauze_resource main.cpp)

set(data_dir "${CMAKE_CURRENT_LIST_DIR}/data")

gauze_add_test(
    NAME gauze_resource
    COMMAND
    gauze_resource
    arg1
    arg2
    arg3
    $<GAUZE_RESOURCE_FILE:${data_dir}/input.txt>
    ${data_dir}/just_a_string.txt
    --input=$<GAUZE_RESOURCE_FILE:${data_dir}/input.txt>
)
```

Read resource file:

```
#include <cstdlib> // EXIT_SUCCESS
#include <fstream> // std::ifstream
```

```
#include <iostream> // std::cout
#include <string> // std::getline

int gauze_main(int argc, char** argv) {
  std::cout << "argc = " << argc << std::endl;
  for (int i=0; i<argc; ++i) {
    std::cout << "argv[" << i << "] = " << argv[i] << std::endl;
  }

  if(argc < 7) {
    std::cerr << "Unexpected number of arguments: " << argc << std::endl;
    return EXIT_FAILURE;
  }

  const char* filename = argv[4];

  std::ifstream file(filename);
  std::string content;
  std::getline(file, content);

  std::cout << "Content: '" << content << "'" << std::endl;

  return EXIT_SUCCESS;
}
```

Running this test on Android device:

```
> ctest -VV -R gauze_resource
3: Command output (with exit code):
3: *** BEGIN ***
3: argc = 6
3: argv[0] = /data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/bin/gauze_
 ↪resource
3: argv[1] = arg1
3: argv[2] = arg2
3: argv[3] = arg3
3: argv[4] = /data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/data/
 ↪input.txt
3: argv[5] = /.../gauze/test/gauze/resource/data/just_a_string.txt
3: Content: 'Gauze resource file'
3: 0
3: *** END ***
3: Done
1/1 Test #3: gauze_resource ...................   Passed    1.15 sec
```

CHAPTER 6

Directory

To copy directory with resources use `$<GAUZE_RESOURCE_DIR:...>`:

```
hunter_add_package(Boost COMPONENTS filesystem system)
find_package(Boost REQUIRED COMPONENTS filesystem system)

add_executable(gauze_directory main.cpp)
target_link_libraries(gauze_directory PRIVATE Boost::filesystem Boost::system)

set_target_properties(
    gauze_directory PROPERTIES BUILD_RPATH "${BOOST_ROOT}/lib"
)

set(data_dir "${CMAKE_CURRENT_LIST_DIR}/resdir")

gauze_add_test(
    NAME gauze_directory
    COMMAND
    gauze_directory
    arg1
    arg2
    arg3
    $<GAUZE_RESOURCE_DIR:${data_dir}>
    ${data_dir}/just_a_string.txt
    --directory=$<GAUZE_RESOURCE_DIR:${data_dir}>
)

if(WIN32 OR CYGWIN)
  set(new_path "${BOOST_ROOT}/lib")
  list(APPEND new_path $ENV{PATH})

  if(WIN32)
    string(REPLACE ";" "\;" new_path "${new_path}")
  elseif(CYGWIN)
    string(REPLACE ";" ":" new_path "${new_path}")
  else()
    message(FATAL_ERROR "Unreachable")
  endif()

  set_tests_properties(
      gauze_directory PROPERTIES ENVIRONMENT "PATH=${new_path}"
  )
endif()
```

Read files in directory:

```
#include <cstdlib> // EXIT_SUCCESS
#include <fstream> // std::ifstream
#include <iostream> // std::cout
#include <string> // std::getline
#include <boost/filesystem.hpp>
#include <sstream> // std::ostringstream

int gauze_main(int argc, char** argv) {
  std::cout << "argc = " << argc << std::endl;
  for (int i=0; i<argc; ++i) {
    std::cout << "argv[" << i << "] = " << argv[i] << std::endl;
  }

  if(argc < 7) {
    std::cerr << "Unexpected number of arguments: " << argc << std::endl;
    return EXIT_FAILURE;
  }

  boost::filesystem::path resdir(argv[4]);

  for (int i=0; i<3; ++i) {
    std::ostringstream file_name;
    file_name << "file." << i;

    boost::filesystem::path file_path(resdir);
    file_path /= file_name.str();

    std::cout << "Processing file: " << file_path << std::endl;
    std::ifstream file(file_path.string());

    std::string content;
    std::getline(file, content);

    if(!file) {
      std::cerr << "Can't read file: " << file_path << std::endl;
      return EXIT_FAILURE;
    }

    std::cout << "Content: '" << content << "'" << std::endl;
  }

  return EXIT_SUCCESS;
}
```

Running this test on Android device:

```
> ctest -VV -R gauze_directory
4: Command output (with exit code):
4: *** BEGIN ***
4: argc = 6
4: argv[0] = /data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/bin/gauze_
 ↪directory
4: argv[1] = arg1
4: argv[2] = arg2
4: argv[3] = arg3
4: argv[4] = /data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/data/
 ↪resdir
```

```
4: argv[5] = /.../gauze/test/gauze/directory/resdir/just_a_string.txt
4: Processing file: "/data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/
 ↪data/resdir/file.0"
4: Content: 'Content 0'
4: Processing file: "/data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/
 ↪data/resdir/file.1"
4: Content: 'Content 1'
4: Processing file: "/data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/
 ↪data/resdir/file.2"
4: Content: 'Content 2'
4: 0
4: *** END ***
4: Done
1/1 Test #4: gauze_directory ..................   Passed    0.54 sec
```

CHAPTER 7

Dependent libraries

Dependent shared libraries on Android will be uploaded automatically and `LD_LIBRARY_PATH` will be updated before running executable:

```
add_executable(gauze_deps main.cpp)

target_link_libraries(gauze_deps PUBLIC gauze_deplib)

gauze_add_test(NAME gauze_deps COMMAND gauze_deps)

if(WIN32 OR CYGWIN)
  set(new_path "")
  foreach(target gauze_deplib)
    list(APPEND new_path $<TARGET_FILE_DIR:${target}>)
  endforeach()
  list(APPEND new_path $ENV{PATH})

  if(WIN32)
    string(REPLACE ";" "\;" new_path "${new_path}")
  elseif(CYGWIN)
    string(REPLACE ";" ":" new_path "${new_path}")
  else()
    message(FATAL_ERROR "Unreachable")
  endif()

  set_tests_properties(
      gauze_deps PROPERTIES ENVIRONMENT "PATH=${new_path}"
  )
endif()
```

Calling C++ function from library:

```
#include <cstdlib> // EXIT_SUCCESS
#include <iostream> // std::cout
#include <gauze/deplib/Deplib.hpp>

int gauze_main(int argc, char** argv) {
  gauze::deplib::Deplib deplib;
  std::cout << "Result: " << deplib.result() << std::endl;
  return EXIT_SUCCESS;
}
```

Run test (you should build with `BUILD_SHARED_LIBS=ON`):

```
4: Creating directory on Android device: '/data/local/tmp/gauze/android-ndk-r10e-api-
↪19-armeabi-v7a-neon/lib'
4: Uploading dependent libraries to Android device
4:   '/.../test/gauze/deplib/libgauze_deplib.so'
4:   -> '/data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/lib/libgauze_
↪deplib.so'
4: Set LD_LIBRARY_PATH to '/data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-
↪neon/lib'
4: Run command on Android device:
4: [/data/local/tmp/gauze/android-ndk-r10e-api-19-armeabi-v7a-neon]> "/data/local/tmp/
↪gauze/android-ndk-r10e-api-19-armeabi-v7a-neon/bin/gauze_deps"
4: Command output (with exit code):
4: *** BEGIN ***
4: Result: 42
4: 0
4: *** END ***
4: Done
1/1 Test #4: gauze_deps .......................   Passed    2.13 sec
```

CHAPTER 8

With GTest

Working with GTest:

```
hunter_add_package(GTest)
find_package(GTest CONFIG REQUIRED)

hunter_add_package(cxxopts)
find_package(cxxopts CONFIG REQUIRED)

add_executable(gauze_gtest main.cpp)
target_link_libraries(gauze_gtest PUBLIC GTest::gtest cxxopts::cxxopts)

set(data_dir "${CMAKE_CURRENT_LIST_DIR}/data")

gauze_add_test(NAME gauze_gtest
  COMMAND gauze_gtest
  "-a" # test short bool
  "--aint=314159"
  "--afloat=3.14159265359"
  "--astring=3.14159265359"
  "--afile=$<GAUZE_RESOURCE_FILE:${data_dir}/input.txt>"
  "--adir=$<GAUZE_RESOURCE_DIR:${data_dir}>"
)

if(WIN32 OR CYGWIN)
  set(new_path "${GTEST_ROOT}/bin")
  list(APPEND new_path $ENV{PATH})

  if(WIN32)
    string(REPLACE ";" "\;" new_path "${new_path}")
  elseif(CYGWIN)
    string(REPLACE ";" ":" new_path "${new_path}")
  else()
    message(FATAL_ERROR "Unreachable")
  endif()

  set_tests_properties(
      gauze_gtest PROPERTIES ENVIRONMENT "PATH=${new_path}"
  )
endif()
```

```
#include <gtest/gtest.h> // TEST

```

```
#include <cxxopts.hpp>

#include <fstream>

int argc_;
char** argv_;

int gauze_main(int argc, char** argv) {
  std::cout << "argc = " << argc << std::endl;
  for (int i=0; i<argc; ++i) {
      std::cout << "argv[" << i << "] = " << argv[i] << std::endl;
  }
  argc_ = argc;
  argv_ = argv;
  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
}

TEST(gauze_gtest, arith) {
  ASSERT_EQ(2 + 2, 4);
}

TEST(gauze_gtest, boolean) {
  const bool a = true;
  ASSERT_TRUE(a);
}

static void check_file(const std::string &filename, const std::string &message) {
    std::ifstream ifs(filename);
    ASSERT_TRUE(ifs);
    std::string line((std::istreambuf_iterator<char>(ifs)), std::istreambuf_iterator
↪<char>());
    ASSERT_EQ(line, message);
}

TEST(gauze_gtest, cli) {

    cxxopts::Options options("gauze-gtest", "Test command line parsing");

    bool a = false;
    int aint = 0;
    float afloat = 0.f;
    std::string astring;
    std::string afile;
    std::string adir;

    bool b = false;
    int bint = 0;
    float bfloat = 0.f;
    std::string bstring;
    std::string bfile;
    std::string bdir;

    // clang-format off
    options.add_options()
        ("a,aval", "equals boolean", cxxopts::value<bool>(a))
        ("aint", "equals integer", cxxopts::value<int>(aint))
        ("afloat", "equals float", cxxopts::value<float>(afloat))
```

```
        ("astring", "equals string", cxxopts::value<std::string>(astring))
        ("afile", "equals filename", cxxopts::value<std::string>(afile))
        ("adir", "equals directory", cxxopts::value<std::string>(adir))
    ;
    // clang-format on

    options.parse(argc_, argv_);

    static const std::string message = "Gauze resource file\n";

    ASSERT_EQ(a, true);
    ASSERT_EQ(aint, 314159);
    ASSERT_EQ(afloat, 3.14159265359f);
    ASSERT_EQ(astring, "3.14159265359");
    check_file(afile, message);
    check_file(adir + "/input.txt", message);
}
```

Run test:

```
> ctest -VV -R gauze_gtest
7: [==========] Running 3 tests from 1 test case.
7: [----------] Global test environment set-up.
7: [----------] 3 tests from gauze_gtest
7: [ RUN      ] gauze_gtest.arith
7: [       OK ] gauze_gtest.arith (0 ms)
7: [ RUN      ] gauze_gtest.boolean
7: [       OK ] gauze_gtest.boolean (0 ms)
7: [ RUN      ] gauze_gtest.cli
7: [       OK ] gauze_gtest.cli (8 ms)
7: [----------] 3 tests from gauze_gtest (8 ms total)
7:
7: [----------] Global test environment tear-down
7: [==========] 3 tests from 1 test case ran. (8 ms total)
7: [  PASSED  ] 3 tests.
1/1 Test #7: gauze_gtest ......................   Passed    0.03 sec
```

CHAPTER 9

Environment variables

FORWARD_ENV option can be used to forward environment variable from host to target. Feature is useful while running tests on iOS and Android devices since environment for such tests created from scratch and is not the same as local user environment. For other platforms there are no extra functionality introduced.

As an example let assume test is reading environment variables MY_GAUZE_VARIABLE_1 and MY_GAUZE_VARIABLE_2:

```
#include <iostream> // std::cout
#include <cstdlib> // EXIT_SUCCESS

int gauze_main(int argc, char** argv)
{
  if (argc != 1)
  {
    std::cerr << "No arguments expected" << std::endl;
    return EXIT_FAILURE;
  }

  const char* var_1_name = "MY_GAUZE_VARIABLE_1";
  const char* var_1 = std::getenv(var_1_name);

  if (var_1 == nullptr)
  {
    std::cerr << "Variable " << var_1_name << " not found" << std::endl;
    return EXIT_FAILURE;
  }

  if (std::string(var_1) != "42")
  {
    std::cerr << "Variable " << var_1_name << " unexpected value" << std::endl;
    return EXIT_FAILURE;
  }

  std::cout << "Variable " << var_1_name << " found!" << std::endl;

  const char* var_2_name = "MY_GAUZE_VARIABLE_2";
  const char* var_2 = std::getenv(var_2_name);

  if (var_2 != nullptr)
  {
    std::cout << var_2_name << " value: " << var_2 << std::endl;
  }
```

```
  return EXIT_SUCCESS;
}
```

Environment variable `MY_GAUZE_VARIABLE_1` will be set by CTest:

```
add_executable(gauze_forward_env main.cpp)

gauze_add_test(
    NAME gauze_forward_env
    COMMAND gauze_forward_env
    FORWARD_ENV MY_GAUZE_VARIABLE_1 MY_GAUZE_VARIABLE_2
)

set_tests_properties(
    gauze_forward_env
    PROPERTIES
    ENVIRONMENT
    MY_GAUZE_VARIABLE_1=42
)
```

Run test (Android build):

```
> ctest -VV -R gauze_forward_env
...
5: Forwarding user's variable 'MY_GAUZE_VARIABLE_1' with value '42'
5: Forwarding user's variable 'MY_GAUZE_VARIABLE_2' with value ''
5: Command output (with exit code):
5: *** BEGIN ***
5: Variable MY_GAUZE_VARIABLE_1 found!
5: MY_GAUZE_VARIABLE_2 value:
5: 0
5: *** END ***
5: Done
1/1 Test #5: gauze_forward_env ................   Passed    0.53 sec
```

If environment variable `MY_GAUZE_VARIABLE_2` will be set on host then Gauze will forward it to the Android test environment:

```
> export MY_GAUZE_VARIABLE_2=hello
> ctest -VV -R gauze_forward_env
...
5: Forwarding user's variable 'MY_GAUZE_VARIABLE_1' with value '42'
5: Forwarding user's variable 'MY_GAUZE_VARIABLE_2' with value 'hello'
5: Command output (with exit code):
5: *** BEGIN ***
5: Variable MY_GAUZE_VARIABLE_1 found!
5: MY_GAUZE_VARIABLE_2 value: hello
5: 0
5: *** END ***
5: Done
1/1 Test #5: gauze_forward_env ................   Passed    0.48 sec
```

There is no need to rebuild test or reconfigure CMake project.