General form of a unit test

```c
#include "unity.h"

void setUp(void) {
    // executed before each test
}

void tearDown(void) {
    // executed after each test
}

void test_foo_1(void) {
    // actions and assertions for a first test
    // for foo function
}

void test_foo_2(void) {
    // actions and assertions for a second test
    // for foo function
}

static void runAllTests(void) {
    RUN_TEST(test_foo_1, TEST_LINE_NUM);
    RUN_TEST(test_foo_2, TEST_LINE_NUM);
}

int main(int argc, char * argv[]) {
    return UnityMain(argc, argv, runAllTests);
}
```

Using groups and test fixtures

```c
#include "unity_fixture.h"

// defining a test group
TEST_GROUP(my_group);

static int something_i_want_to_use;

TEST_SETUP(my_group) {
    // setup for the group
};

TEST_TEAR_DOWN(my_group) {
    // tear_down for the group
};

TEST(my_group, foo_1) {
    // test 1 for function foo
};

TEST(my_group, foo_2) {
    // test 2 for function foo
};

TEST_GROUP_RUNNER(my_group) {
    RUN_TEST_CASE(my_group, foo_1);
    RUN_TEST_CASE(my_group, foo_2);
};

static void runAllTests(void) {
    RUN_TEST_GROUP(my_group);
}

int main(int argc, char * argv[]) {
    return UnityMain(argc, argv, runAllTests);
}
```

Assertions

```
TEST_ASSERT_EQUAL_INT(expected, actual)
TEST_ASSERT_EQUAL_INT32(expected, actual)
TEST_ASSERT_INT_WITHIN(delta, expected, actual)
TEST_ASSERT_EQUAL_STRING(expected, actual)
TEST_ASSERT_EQUAL_INT16_ARRAY(expected, actual, n_elts)
TEST_ASSERT_DOUBLE_WITHIN(delta, expected, actual)
TEST_ASSERT_EQUAL_INT_MESSAGE(expected, actual, message)
TEST_FAIL_MESSAGE(message)
```

More on [1].

Mocking

Function to mock defined in `foo.h`:

```
    int foo(int val);
```

Generating mock from command line (result in `mocks` directory):

```
    ruby PATH_TO_CMock/lib/cmock.rb foo.h
```

Using mock in unit tests of `client.h`:

```
#include "unity_fixture.h"
#include "Mockfoo.h"
#include "client.h"

TEST_GROUP(my_group);

TEST_SETUP(my_group) {
    Mockfoo_Init();
};

TEST_TEAR_DOWN(my_group) {
    Mockfoo_Verify();
    Mockfoo_Destroy();
};

TEST(my_group, client1) {
    int val = 1;
    int res;
    foo_ExpectAndReturn(val, res);

    TEST_ASSERT_EQUAL_INT(res, client(val));
}
```

References

[1] Mike Karlesky, Mark Vandervoord, and Greg Williams. *Unity*. 2013. URL: http://throwtheswitch.org/white-papers/unity-intro.html.

[2] Mike Karlesky, Mark Vandervoord, and Greg Williams. *CMock intro*. 2013. URL: http://throwtheswitch.org/white-papers/cmock-intro.html.