Epitech Documentation

Unit tests for C/C++

How to write them

Unit tests rely on Criterion library.

PROJECT TREE

The files containing the unit tests must be placed in a folder named `tests` at the root of the project. The `Makefile` must be directly in the `tests` directory.

The subtree of this `tests` directory is up to you.

Here is an example of project tree:

```
Terminal
~/ > tree -F
.
|-- include/
|    |-- main.h
|    |-- utils.h
|-- Makefile
|-- src
|  |-- main.c
|  |-- utils.c
|-- tests
|  |-- tests_utils.c
|  |-- Makefile
```

COMPILING UNIT TESTS

The `Makefile` used to compile the unit tests will be this one ; you must only add your source files and your tests files where specified in the Makefile.

```
SRC_DIR=        $(realpath ..)

# Must list all project files without the main() function
# Criterion uses its own main() ;
# having a main() in any .c file will have the build fail
SRC=            $(SRC_DIR)/XXXX.c
                $(SRC_DIR)/YYYY.c

SRC_UT_DIR=     $(realpath .)

# Must list all files containing unit tests
SRC_UT=         $(SRC_UT_DIR)/tests_XXXX.c
                $(SRC_UT_DIR)/tests_YYYY.c

OBJ=            $(SRC:.c=.o) $(SRC_UT:.c=.o)

CFLAGS=         -Wall -Wextra --coverage

LDFLAGS=        -lcriterion -lgcov

NAME=           units

all:            $(NAME)

$(NAME):        $(OBJ)
                cc -o $(NAME) $(OBJ) $(LDFLAGS)

clean:
                rm -f $(OBJ)

fclean:         clean
                rm -f $(NAME)

re:             fclean all
```

TESTS FILES

The tests files only contain the tests, following this format:

```
#include <criterion/criterion.h>

Test(suite_name, test_name)
{
    ...
}
```

with *suite_name + test_name* unique.

The list of asserts is here.
The most used are:

```
// Passes if Expression is true
cr_assert(Expression, FormatString, ...);
// Passes if Expression is false
cr_assert_not(Expression, FormatString, ...);
// Passes if Actual == Expected
cr_assert_eq(Actual, Expected, FormatString, ...);
// Passes if Actual != Expected
cr_assert_neq(Actual, Expected, FormatString, ...);
```

EXAMPLES

Criterion maintainer has written many example files.
Basic usage of Criterion can be found here and here.
Here is an example of a unit test file:

```c
#include <criterion/criterion.h>

const char *str = "Hello world";
const int len = 11;

Test(utils, is_str_length_equal_to_len_v1)
{
    cr_assert(strlen(str) == len);
}

Test(utils, is_str_length_equal_to_len_v2)
{
    cr_assert_eq(strlen(str), len);
}

Test(utils, is_str_length_equal_to_len_v3)
{
    cr_assert_not(strlen(str) != len);
}
```

The 3 tests are doing the same thing with different syntaxes.
They check that the `"Hello world"` string has a length of 11 characters.

However the following test aborts:

```c
Test(utils, is_str_length_different_to_len)
{
    cr_assert_neq(strlen(str), len);
}
```