Unit Testing

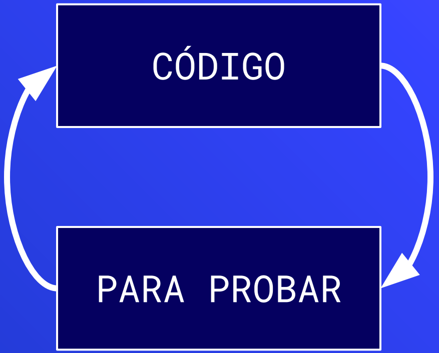
ITBA - Sistemas Operativos

Agenda

- ¿Qué son los Test Unitarios?
- Test Driven Development
- Testing en C
- CuTest
- Demo
- Estructura
- Bibliografía recomendada

1.
¿Qué son los Test Unitarios?

Idea Básica



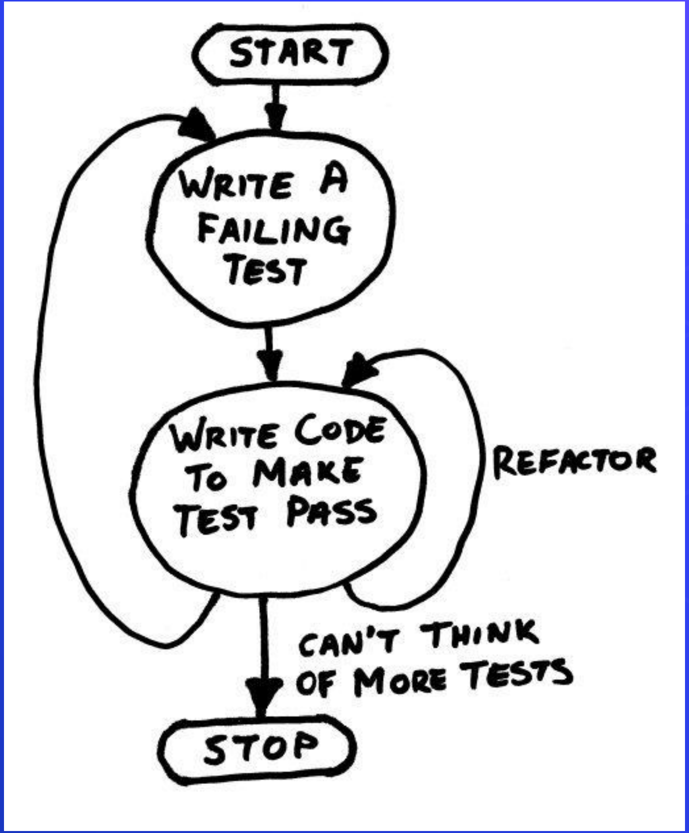
Unit Test

- Probar por Separado todas las partes
- Independiente
- Si se cambia el código → Detección de errores.

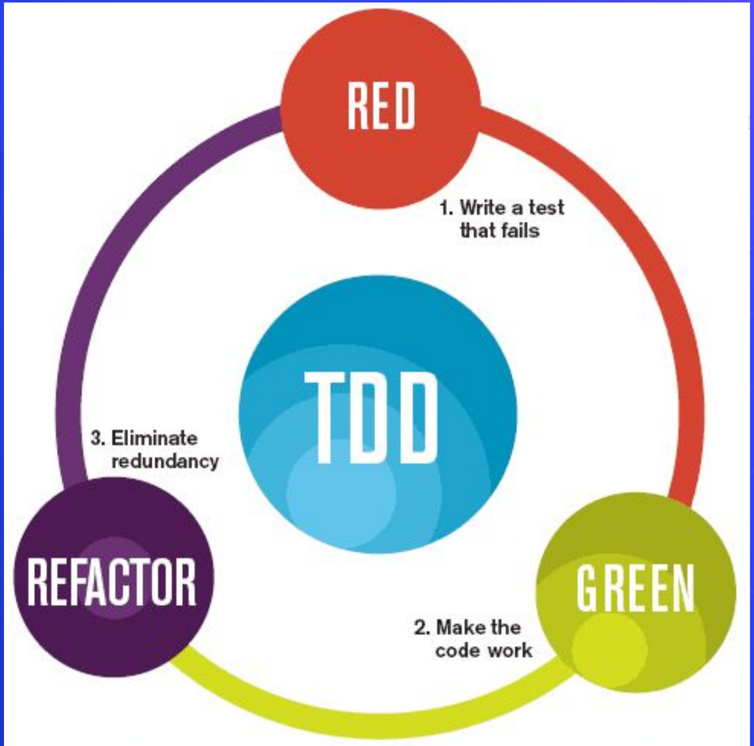
Todo junto se escribe separado ...

2. Test Driven Development

TDD (Test Driven Development)



Red, Green, Refactor ...

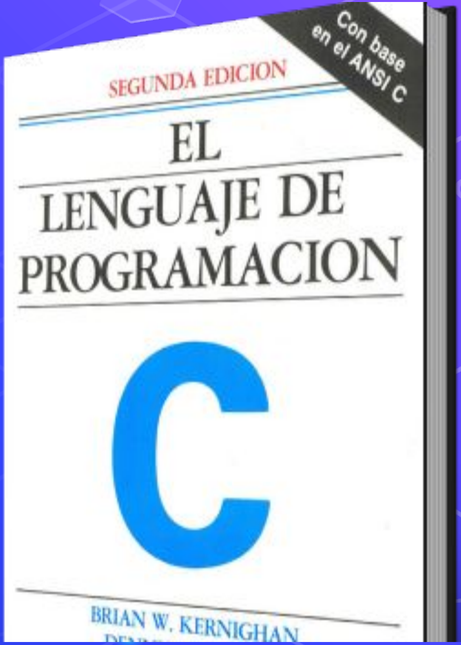


The mantra of Test-Driven Development (TDD) is “red, green, refactor.”

3.
Testing en C

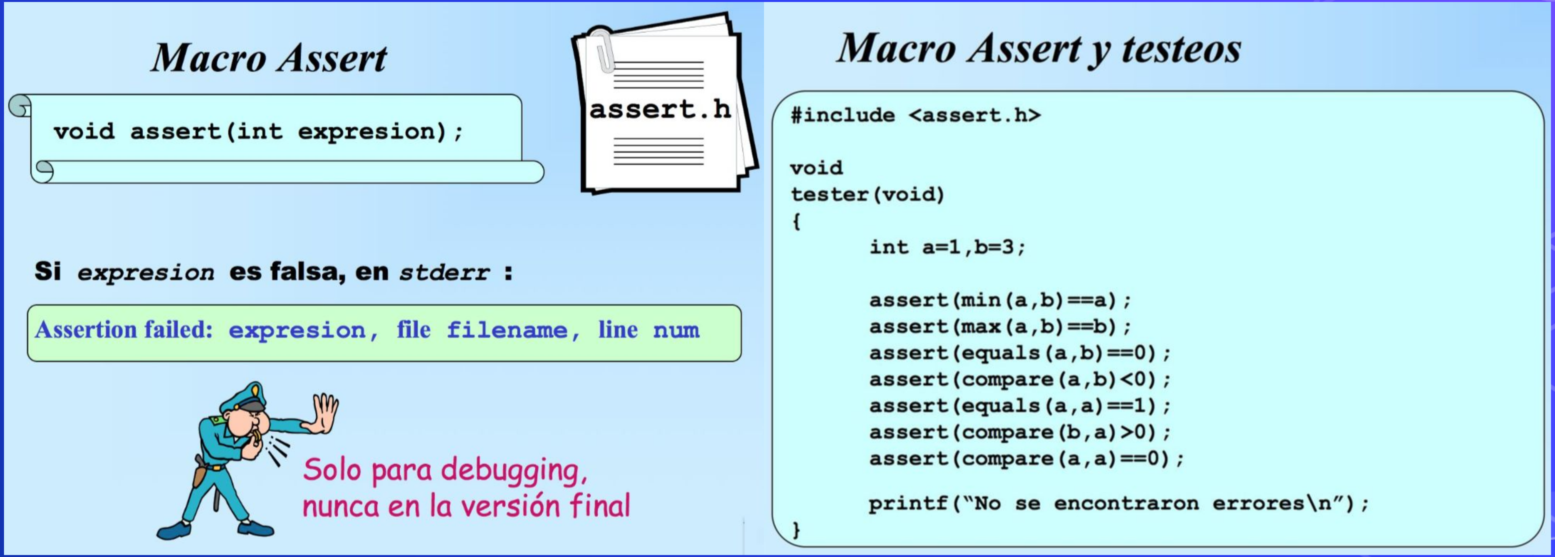
Unit Test en C

- Macro Assert
- Libreria de Testeo Propia
- Framework de Terceros



Macro Assert

¿Qué dirían en Programación Imperativa?



Macro Assert

```
void assert(int expresion);
```

Si ***expresion*** **es falsa, en** ***stderr*** **:**

Assertion failed: expresion, file filename, line num

Solo para debugging,
nunca en la versión final

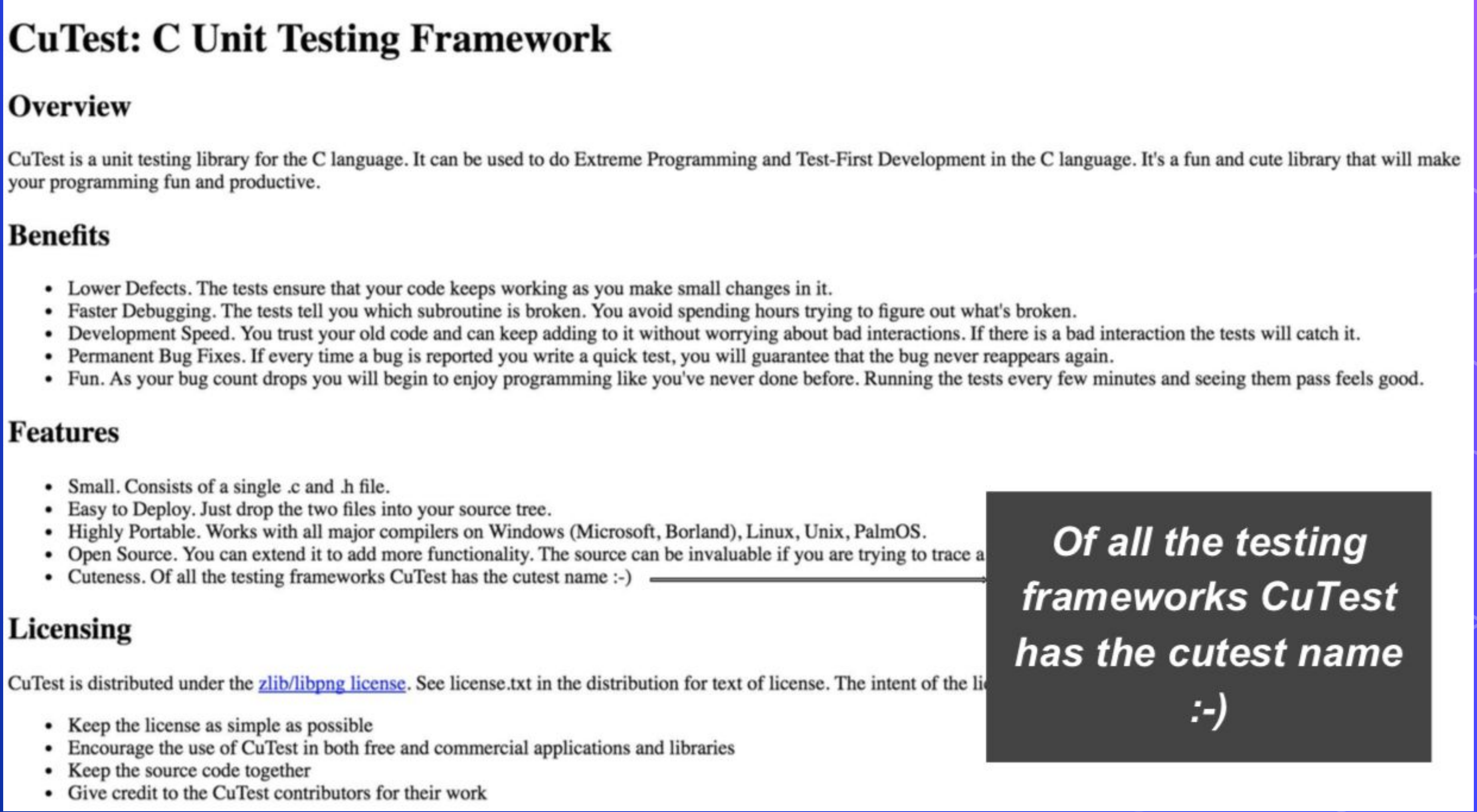
Macro Assert y testeos

```
#include <assert.h>

void
tester(void)
{
        int a=1,b=3;

        assert(min(a,b)==a);
        assert(max(a,b)==b);
        assert(equals(a,b)==0);
        assert(compare(a,b)<0);
        assert(equals(a,a)==1);
        assert(compare(b,a)>0);
        assert(compare(a,a)==0);

        printf("No se encontraron errores\n");
}
```

4.
CuTest

CuTest: C Unit Testing Framework

- http://cutest.sourceforge.net/
- https://github.com/ennorehling/cutest



CuTest: C Unit Testing Framework

Overview

CuTest is a unit testing library for the C language. It can be used to do Extreme Programming and Test-First Development in the C language. It's a fun and cute library that will make your programming fun and productive.

Benefits

- Lower Defects. The tests ensure that your code keeps working as you make small changes in it.
- Faster Debugging. The tests tell you which subroutine is broken. You avoid spending hours trying to figure out what's broken.
- Development Speed. You trust your old code and can keep adding to it without worrying about bad interactions. If there is a bad interaction the tests will catch it.
- Permanent Bug Fixes. If every time a bug is reported you write a quick test, you will guarantee that the bug never reappears again.
- Fun. As your bug count drops you will begin to enjoy programming like you've never done before. Running the tests every few minutes and seeing them pass feels good.

Features

- Small. Consists of a single .c and .h file.
- Easy to Deploy. Just drop the two files into your source tree.
- Highly Portable. Works with all major compilers on Windows (Microsoft, Borland), Linux, Unix, PalmOS.
- Open Source. You can extend it to add more functionality. The source can be invaluable if you are trying to trace a
- Cuteness. Of all the testing frameworks CuTest has the cutest name :-)

Licensing

CuTest is distributed under the zlib/libpng license. See license.txt in the distribution for text of license. The intent of the li

- Keep the license as simple as possible
- Encourage the use of CuTest in both free and commercial applications and libraries
- Keep the source code together
- Give credit to the CuTest contributors for their work

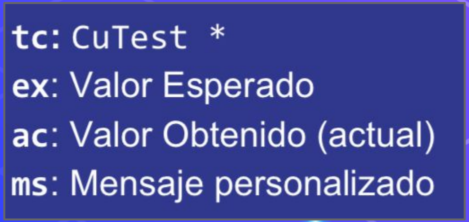
Of all the testing frameworks CuTest has the cutest name :-)

CuTest vs Assert

- Se corren **TODOS** los test !!!
- No se termina al primer “Assertion Failure”
- Mensajes más detallados (Valor Obtenido y Esperado)
- Mayor variedad de Funciones (Más Claridad)

CuTest.h

```
#define CuFail(tc, ms)
#define CuAssert(tc, ms, cond)
#define CuAssertTrue(tc, cond)
#define CuAssertStrEquals(tc,ex,ac)
#define CuAssertStrEquals_Msg(tc,ms,ex,ac)
#define CuAssertIntEquals(tc,ex,ac)
#define CuAssertIntEquals_Msg(tc,ms,ex,ac)
#define CuAssertDblEquals(tc,ex,ac,dl)
#define CuAssertDblEquals_Msg(tc,ms,ex,ac,dl)
#define CuAssertPtrEquals(tc,ex,ac)
#define CuAssertPtrEquals_Msg(tc,ms,ex,ac)
#define CuAssertPtrNotNull(tc,p)
#define CuAssertPtrNotNullMsg(tc,msg,p)
```



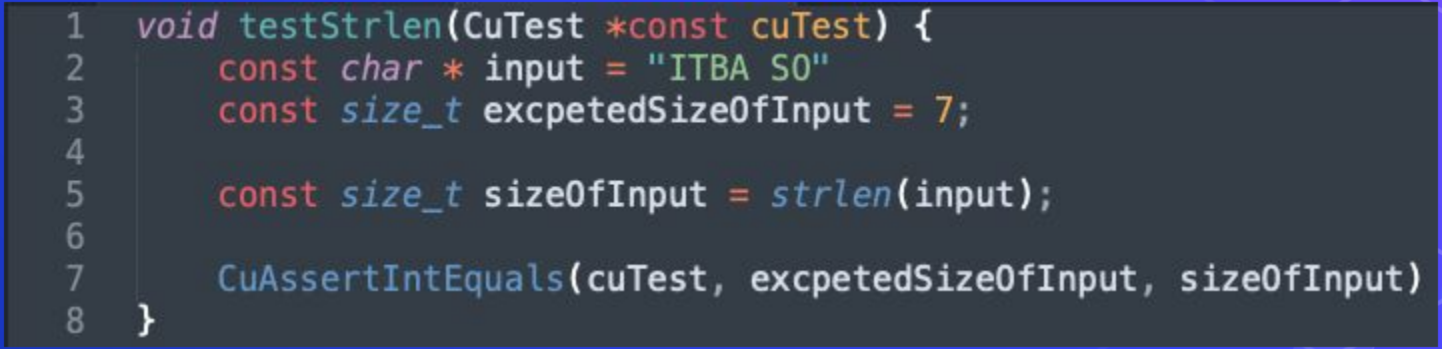
tc: `CuTest *`
ex: Valor Esperado
ac: Valor Obtenido (actual)
ms: Mensaje personalizado

CuTest - Terminos

- **Test unitario:** funciones que reciben por parámetro un **CuTest** * e invocan a algunas de las funciones de CuTest.h mencionadas.

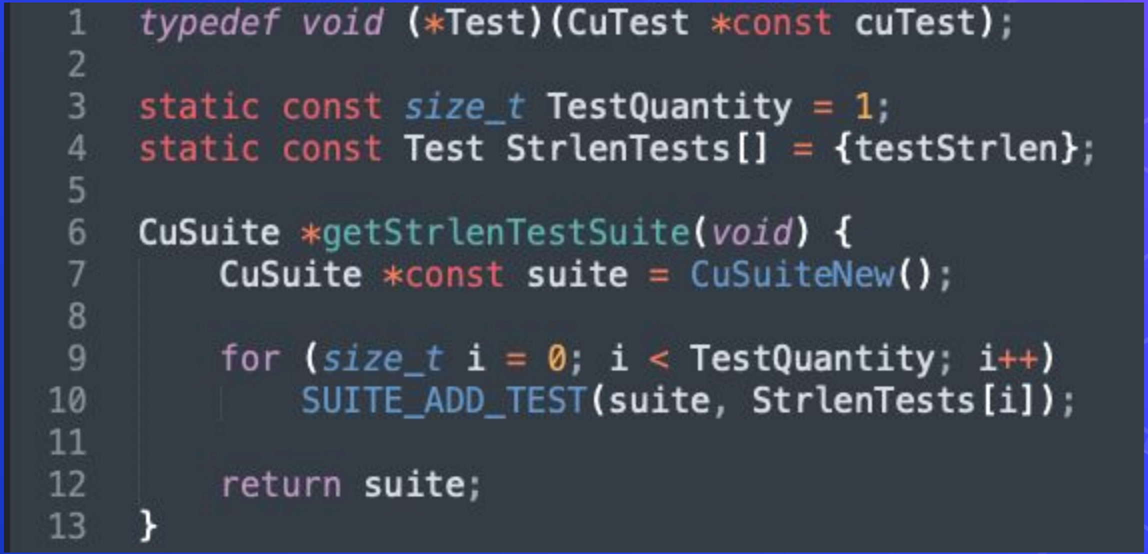
- **Suite de tests:** función que retorne un **CuSuite** * y tiene al menos un test unitario.

Ejemplo de Unit Test



```
void testStrlen(CuTest *const cuTest) {
    const char * input = "ITBA SO"
    const size_t excpetedSizeOfInput = 7;

    const size_t sizeOfInput = strlen(input);

    CuAssertIntEquals(cuTest, excpetedSizeOfInput, sizeOfInput)
}
```

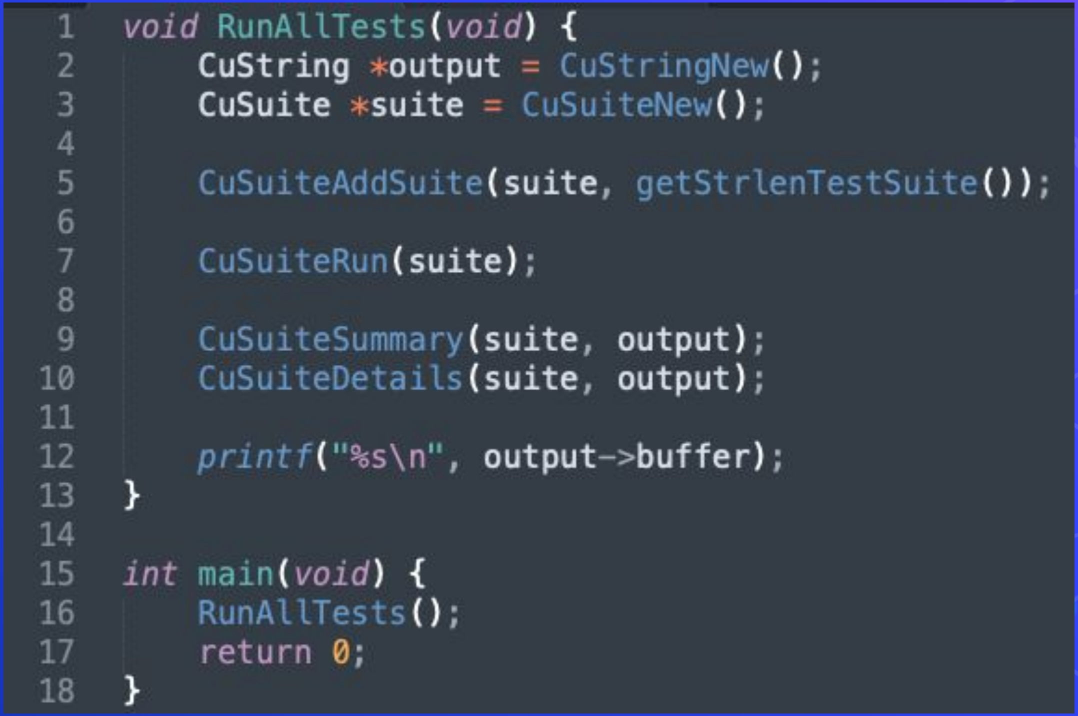
Ejemplo de Suite



```c
typedef void (*Test)(CuTest *const cuTest);

static const size_t TestQuantity = 1;
static const Test StrlenTests[] = {testStrlen};

CuSuite *getStrlenTestSuite(void) {
    CuSuite *const suite = CuSuiteNew();

    for (size_t i = 0; i < TestQuantity; i++)
        SUITE_ADD_TEST(suite, StrlenTests[i]);

    return suite;
}
```

Se recomienda un archivo .c independiente para cada suite.

Ejemplo AllTest.c



```c
void RunAllTests(void) {
    CuString *output = CuStringNew();
    CuSuite *suite = CuSuiteNew();

    CuSuiteAddSuite(suite, getStrlenTestSuite());

    CuSuiteRun(suite);

    CuSuiteSummary(suite, output);
    CuSuiteDetails(suite, output);

    printf("%s\n", output->buffer);
}

int main(void) {
    RunAllTests();
    return 0;
}
```

¿Cómo usamos CuTest en nuestros Proyectos?

En el directorio de trabajo, debemos tener los archivos:

- CuTest.c y CuTest.h
- AllTests.c
- Los .c de nuestras suites de tests.
- Otros archivos .c y .h de la funcionalidad testeada.

5. Estructura

Unit Tests Structures & Patterns

- Behavior-Driven Development (BDD)
  - Given-When-Then
  - Arrange-Act-Assert

6.
Demo

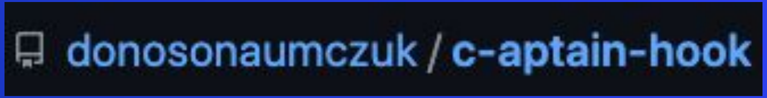
c-unit-testing-example

alejoaquili/c-unit-testing-example

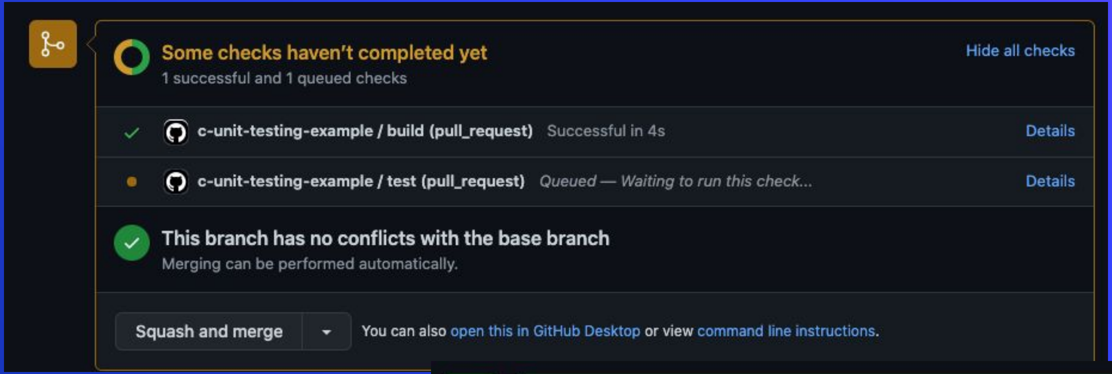
Git Hooks

Son scripts que Git ejecuta antes/después de algún evento. Estan en .git/hooks

- Pre-commit
- Post-commit
- Pre-push
- etc


donosonaumczuk / c-aptain-hook

Github Actions



Some checks haven't completed yet
1 successful and 1 queued checks

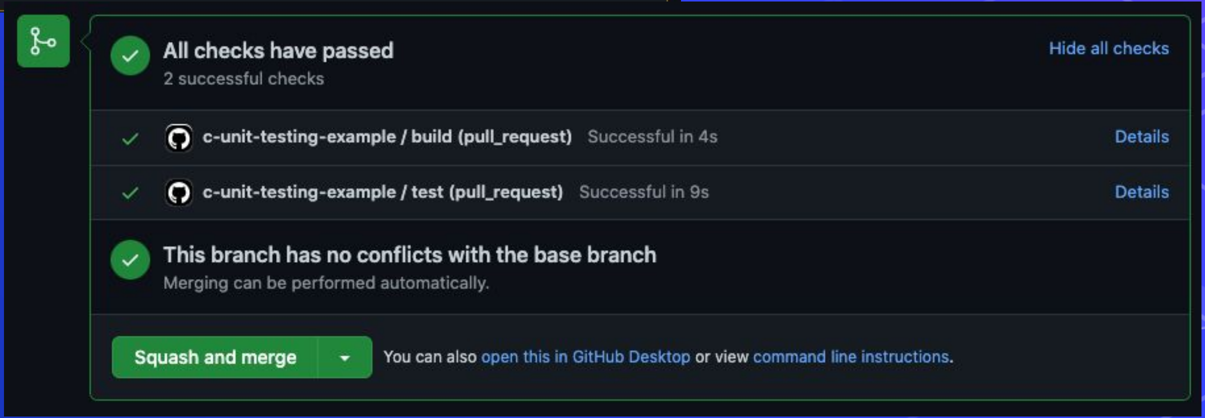
Hide all checks

✓ c-unit-testing-example / build (pull_request) Successful in 4s Details

● c-unit-testing-example / test (pull_request) *Queued — Waiting to run this check...* Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.



All checks have passed
2 successful checks

Hide all checks

✓ c-unit-testing-example / build (pull_request) Successful in 4s Details

✓ c-unit-testing-example / test (pull_request) Successful in 9s Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

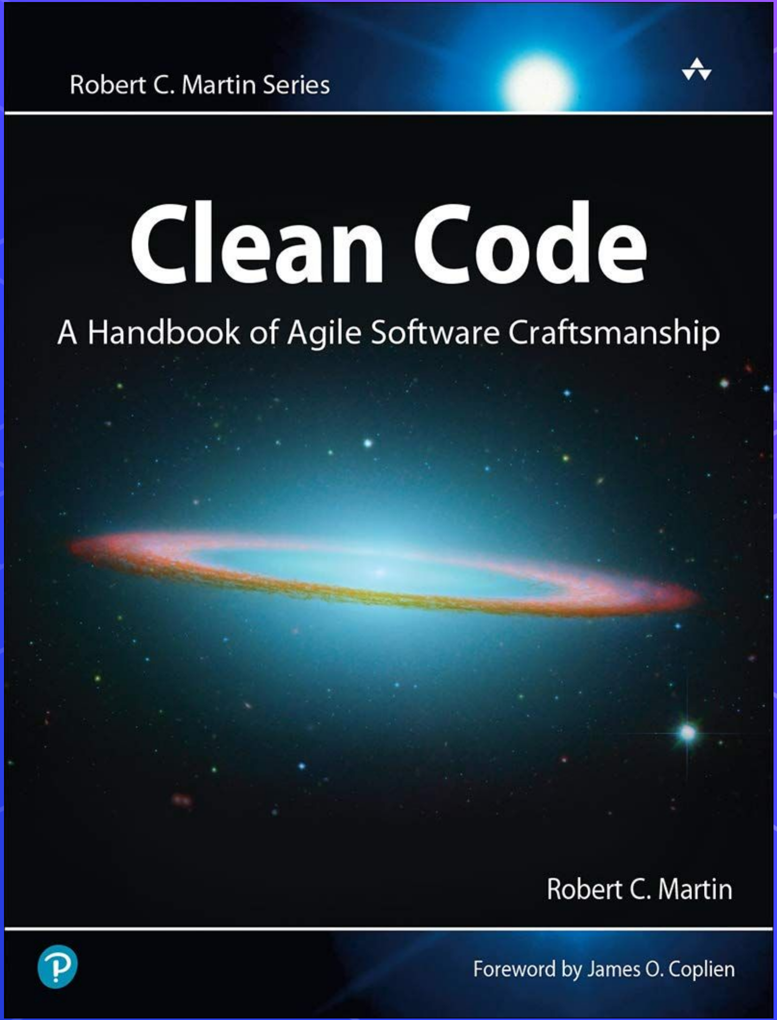
You can also open this in GitHub Desktop or view command line instructions.

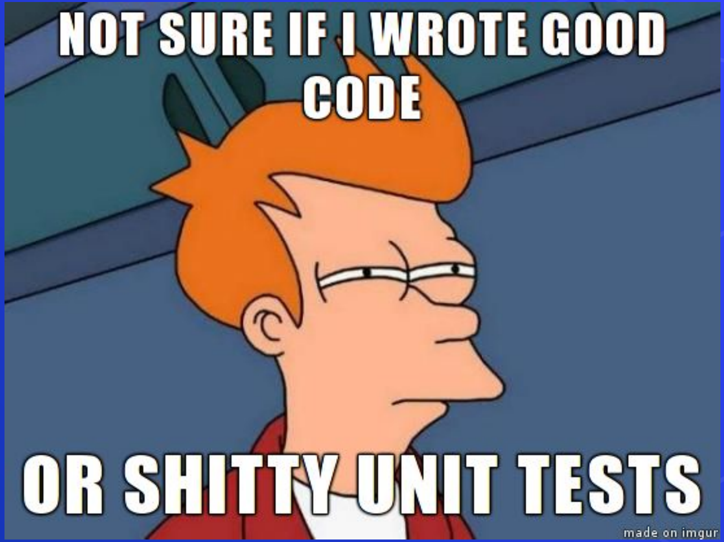
GitHub Docs

7.
Bibliografía recomendada

Clean Code: A Handbook of Agile Software Craftsmanship

Libro de Robert C. Martin


Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Robert C. Martin
Foreword by James O. Coplien



¡Muchas Gracias!
¿Preguntas?