cpputest / **cpputest**

CppUTest unit testing and mocking framework for C/C++ http://cpputest.github.com

#cpputest #unit-testing #test-driven-development #c-plus-plus #very-kewl #memory-leak #mocking-framework

3,253 commits | **3** branches | **8** releases | **63** contributors | BSD-3-Clause

Branch: **master** | **New pull request** | **Find file** | **Clone or download**

basvodde committed on **GitHub** Merge pull request #1113 from pmalek/master ··· Latest commit `ce0b67b` on 6 Aug

.settings	Committing accidentally removed file	2 years ago
build	Cleaned up configure.ac and added no long long to MakefileWorked	a year ago
builds	void operator delete was trying to return something. It caused proble…	4 years ago
cmake	Add cmake cross-build support	2 months ago
cpputest_build	Add final newline	3 years ago
docs	Created VS2010 related document and template.	5 years ago

examples	Removing IEEE754 implementation from MAC	9 months ago
include	Merge pull request #1096 from dlindelof/CHECK_RELATION	2 months ago
m4	Install a generated config.h which is prefixed	a year ago
platforms	Moving the test source files to fix error introduced with `02513e8` (pu…	4 months ago
platforms_examples	Removed Prag Prog, Grenning copyright.	a year ago
scripts	Check for clang fixed.	6 months ago
src	Fixing cmake_minimum_version to 3.1 as target_sources requires it	a month ago
tests	Merge pull request #1096 from dlindelof/CHECK_RELATION	2 months ago
.cproject	Suggested improvements - hopefully "kosher"	3 years ago
.gitattributes	.dep and .mak should be crlf as well	3 years ago
.gitignore	Removed and ignoring IAR automatically generated file	10 months ago
.project	fixed the broken eclipse project files	6 years ago
.travis.yml	Secure keys added.	6 months ago
.travis_github_deployer.yml	Hopefully this format is final	3 years ago
AUTHORS	Files needed for GNU standards	5 years ago
CMakeLists.txt	Fixing cmake_minimum_version to 3.1 as target_sources requires it	a month ago
COPYING	Files needed for GNU standards	5 years ago
ChangeLog	Fixed a very weird ChangeLog :)	2 years ago
CppUTest.dep	Re-generate .mak and .dep	3 years ago
CppUTest.dsp	Add build option for longlong	a year ago
CppUTest.dsw	Revert "Removed non-existing AllTests.dsp from CppUTest.dsw"	3 years ago
CppUTest.mak	Add build option for longlong	a year ago

CppUTest.sln	Regenerated and test solution (VS2005)	3 years ago
CppUTest.vcproj	Add/Remove headers in project file	a year ago
CppUTest.vcxproj	Add a missing header file	a year ago
CppUTestConfig.cmake.build.in	do not include config if target is already defined	2 years ago
CppUTestConfig.cmake.install.in	export targets	2 years ago
CppUTest_VS201x.sln	Add converted VS201x solution for examples; rename	3 years ago
Doxyfile	Small changes to the Doxgen generation. Lots more to do related to that	4 years ago
Makefile.am	Updated location of tests in autotools file	10 months ago
Makefile_CppUTestExt	Delayed merge from @**simshi**. Merge request 48	5 years ago
Makefile_using_MakefileWorker	Fix path in old-style makefile for new test folder	10 months ago
NEWS	Files needed for GNU standards	5 years ago
README	Removed unneeded whitesapce	3 years ago
README.md	Readme updated.	a year ago
README_CppUTest_for_C.txt	spelling fixes	2 years ago
README_InstallCppUTest.txt	Update READMEs to make InstallScript.sh working	2 years ago
README_UsersOfPriorVersions.txt	Added README for prior users	10 years ago
appveyor.yml	Wildcard doesn't work for cache	a year ago
autogen.sh	removing the configure call	3 years ago
config.h.cmake	Add build option for longlong	a year ago
configure.ac	Added support for strdup macro replacement on tests. Using auto gener…	10 months ago
cpputest.pc.in	- Adding the required linker flags to the pkgcfg.	5 years ago

cpputest_doxy_gen.conf	First doxygen file	5 years ago
makeAndRun.bat	Add makeAndRun.bat again	3 years ago
makeVS2008.bat	Fix the line endings	3 years ago
makeVS201x.bat	...and again	3 years ago
makeVc6.bat	Fix the line endings	3 years ago
valgrind.suppressions	Hope this now also works when the compiler optimizes	4 years ago

README.md

CppUTest

CppUTest unit testing and mocking framework for C/C++

More information on the project page

Travis Linux build status: build passing

AppVeyor Windows build status: build passing

Coverage: coverage 100%

Getting Started

You'll need to do the following to get started:

Building from source (unix-based, cygwin, MacOSX):

- Download latest version
- autogen.sh
- configure
- make
- make check
- You can use "make install" if you want to install CppUTest system-wide

You can also use CMake, which also works for Windows Visual Studio.

- Download latest version
- cmake CMakeList.txt
- make

Then to get started, you'll need to do the following:

- Add the include path to the Makefile. Something like:
  - CPPFLAGS += -I(CPPUTEST_HOME)/include
- Add the memory leak macros to your Makefile (needed for additional debug info!). Something like:
  - CXXFLAGS += -include $(CPPUTEST_HOME)/include/CppUTest/MemoryLeakDetectorNewMacros.h
  - CFLAGS += -include $(CPPUTEST_HOME)/include/CppUTest/MemoryLeakDetectorMallocMacros.h
- Add the library linking to your Makefile. Something like:
  - LD_LIBRARIES = -L$(CPPUTEST_HOME)/lib -lCppUTest -lCppUTestExt

After this, you can write your first test:

```
TEST_GROUP(FirstTestGroup)
{
};

TEST(FirstTestGroup, FirstTest)
```

```
{
   FAIL("Fail me!");
}
```

Command line switches

- -v verbose, print each test name as it runs
- -r# repeat the tests some number of times, default is one, default if # is not specified is 2. This is handy if you are experiencing memory leaks related to statics and caches.
- -g group only run test whose group contains the substring group
- -n name only run test whose name contains the substring name

Test Macros

- TEST(group, name) - define a test
- IGNORE_TEST(group, name) - turn off the execution of a test
- TEST_GROUP(group) - Declare a test group to which certain tests belong. This will also create the link needed from another library.
- TEST_GROUP_BASE(group, base) - Same as TEST_GROUP, just use a different base class than Utest
- TEST_SETUP() - Declare a void setup method in a TEST_GROUP - this is the same as declaring void setup()
- TEST_TEARDOWN() - Declare a void setup method in a TEST_GROUP
- IMPORT_TEST_GROUP(group) - Export the name of a test group so it can be linked in from a library. Needs to be done in main.

Set up and tear down support

- Each TEST_GROUP may contain a setup and/or a teardown method.

- setup() is called prior to each TEST body and teardown() is called after the test body.

Assertion Macros

The failure of one of these macros causes the current test to immediately exit

- CHECK(boolean condition) - checks any boolean result
- CHECK_TRUE(boolean condition) - checks for true
- CHECK_FALSE(boolean condition) - checks for false
- CHECK_EQUAL(expected, actual) - checks for equality between entities using ==. So if you have a class that supports operator==() you can use this macro to compare two instances.
- STRCMP_EQUAL(expected, actual) - check const char* strings for equality using strcmp
- LONGS_EQUAL(expected, actual) - Compares two numbers
- BYTES_EQUAL(expected, actual) - Compares two numbers, eight bits wide
- POINTERS_EQUAL(expected, actual) - Compares two const void *
- DOUBLES_EQUAL(expected, actual, tolerance) - Compares two doubles within some tolerance
- FAIL(text) - always fails
- TEST_EXIT - Exit the test without failure - useful for contract testing (implementing an assert fake)

Customize CHECK_EQUAL to work with your types that support operator==()

- Create the function: `SimpleString StringFrom(const yourType&)`

The Extensions directory has a few of these.

Building default checks with TestPlugin

- CppUTest can support extra checking functionality by inserting TestPlugins

- TestPlugin is derived from the TestPlugin class and can be inserted in the TestRegistry via the installPlugin method.
- TestPlugins can be used for, for example, system stability and resource handling like files, memory or network connection clean-up.
- In CppUTest, the memory leak detection is done via a default enabled TestPlugin

Example of a main with a TestPlugin:

```
int main(int ac, char** av)
{
   LogPlugin logPlugin;
   TestRegistry::getCurrentRegistry()->installPlugin(&logPlugin);
   int result = CommandLineTestRunner::RunAllTests(ac, av);
   TestRegistry::getCurrentRegistry()->resetPlugins();
   return result;
}
```

Memory leak detection

- A platform specific memory leak detection mechanism is provided.
- If a test fails and has allocated memory prior to the fail and that memory is not cleaned up by TearDown, a memory leak is reported. It is best to only chase memory leaks when other errors have been eliminated.
- Some code uses lazy initialization and appears to leak when it really does not (for example: gcc stringstream used to in an earlier release). One cause is that some standard library calls allocate something and do not free it until after main (or never). To find out if a memory leak is due to lazy initialization set the -r switch to run tests twice. The signature of this situation is that the first run shows leaks and the second run shows no leaks. When both runs show leaks, you have a leak to find.

How is memory leak detection implemented?

- Before setup() a memory usage checkpoint is recorded

- After teardown() another checkpoint is taken and compared to the original checkpoint
- In Visual Studio the MS debug heap capabilities are used
- For GCC a simple new/delete count is used in overridden operators new, new[], delete and delete[]

If you use some leaky code that you can't or won't fix you can tell a TEST to ignore a certain number of leaks as in this example:

```
TEST(MemoryLeakWarningTest, Ignore1)
{
    EXPECT_N_LEAKS(1);
    char* arrayToLeak1 = new char[100];
}
```

Example Main

```
#include "CppUTest/CommandLineTestRunner.h"

int main(int ac, char** av)
{
  return RUN_ALL_TESTS(ac, av);
}
```

Example Test

```
#include "CppUTest/TestHarness.h"
#include "ClassName.h"

TEST_GROUP(ClassName)
{
```

```
  ClassName* className;

  void setup()
  {
    className = new ClassName();
  }
  void teardown()
  {
    delete className;
  }
}

TEST(ClassName, Create)
{
  CHECK(0 != className);
  CHECK(true);
  CHECK_EQUAL(1,1);
  LONGS_EQUAL(1,1);
  DOUBLES_EQUAL(1.000, 1.001, .01);
  STRCMP_EQUAL("hello", "hello");
  FAIL("The prior tests pass, but this one doesn't");
}
```

There are some scripts that are helpful in creating your initial h, cpp, and Test files. See scripts/README.TXT