



Boost.Test

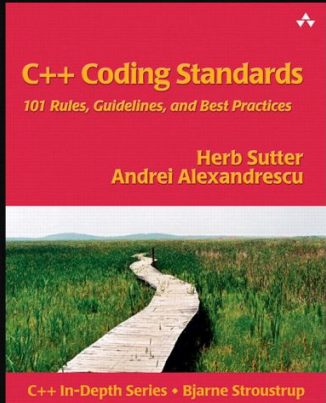
© 2016 Richel Bilderbeek



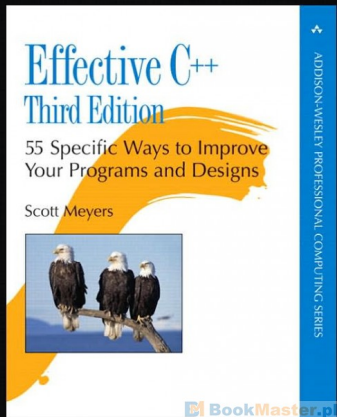
www.github.com/richelbilderbeek/CppPresentations



About Boost



...one of the most highly regarded and expertly designed C++ library projects in the world.



Item 55:
Familiarize yourself with Boost



The obvious solution for most programmers is to use a library that provides an elegant and efficient platform independent to needed services. Examples are Boost... [1]

[1] Bjarne Stroustrup, Abstraction, libraries, and efficiency in C++



Boost.Test

- What: a testing framework
- Why:
 - measure that your code is correct
 - improve your software architecture
 - use an expertly designed testing framework
- Mastery:
 - write tests with high code coverage run by a continuous integration service
 - use other testing frameworks when those fit better, e.g. QTest when using Qt

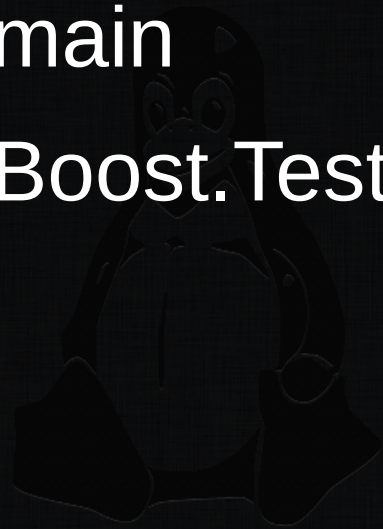
This presentation

- First example: 'add'
- Testing for exceptions: 'divide'
- Add Travis CI



First example

- Function 'add' that adds two integers
- 'add' is used in a main
- 'add' is tested by Boost.Test



Setup

`adder.pro`

`main.cpp`

`my_functions.h`

`my_functions.cpp`

`adder_test.pro`


`my_functions_test.cpp`

`main_test.cpp`

main.cpp

```
#include "my_functions.h"
#include <iostream>

int main()
{
    std::cout << add(40, 2) << '\n';
}
```



my_functions.h

```
#ifndef MY_FUNCTIONS_H
#define MY_FUNCTIONS_H

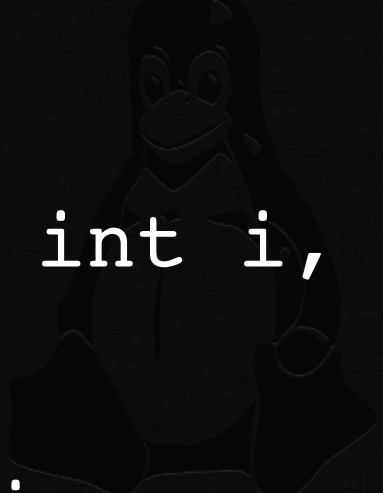
int add(const int i, const int j);

#endif // MY_FUNCTIONS_H
```


my_functions.cpp

```
#include "my_functions.h"

int add(const int i, const int j)
{
    return i + j;
}
```



main_test.cpp

```
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE my_functions_test_module
#include <boost/test/unit_test.hpp>

//No main needed, BOOST_TEST_DYN_LINK creates it
```


my_functions_test.cp

p

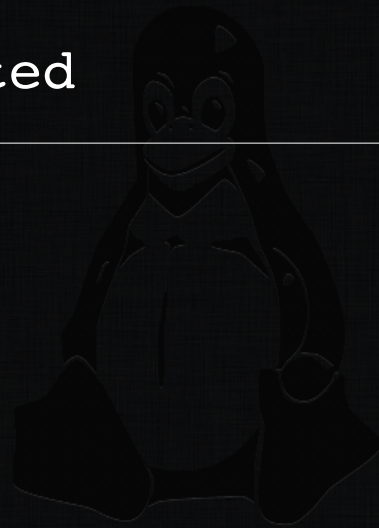
```
#include <boost/test/unit_test.hpp>
#include "my_functions.h"

BOOST_AUTO_TEST_CASE(add_works)
{
    BOOST_CHECK(add(1, 1) == 2);
    BOOST_CHECK(add(1, 2) == 3);
    BOOST_CHECK(add(1, 3) == 4);
    BOOST_CHECK(add(1, 4) == 5);
}
```

Output

```
Running 1 test case...
```

```
*** No errors detected
```



Testing for exceptions

```
#include <stdexcept>

double divide(const double n, const double d)
{
    if (d == 0.0)
    {
        throw std::invalid_argument(
            "divide: denominator cannot be zero"
        );
    }
    return n / d;
}
```


Testing for exceptions

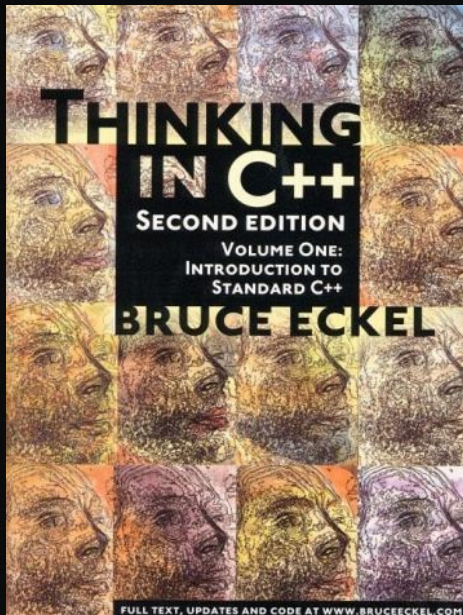
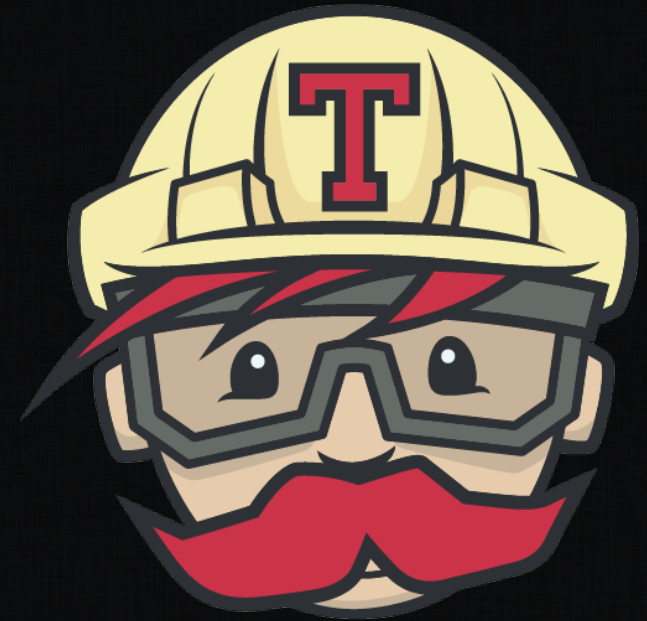
```
BOOST_AUTO_TEST_CASE(divide_use)
{
    {
        const double measured = divide(1.0, 1.0);
        const double expected = 1.0;
        const double error = std::abs(measured - expected);
        const double tolerance = 0.01;
        BOOST_CHECK(error < tolerance);
    }
    {
        const double measured = divide(1.0, 2.0);
        const double expected = 0.5;
        const double error = std::abs(measured - expected);
        const double tolerance = 0.01;
        BOOST_CHECK(error < tolerance);
    }
}
```


Testing for exceptions

```
BOOST_AUTO_TEST_CASE(divide_abuse) {  
    try {  
        divide(0.0, 0.0);  
        BOOST_CHECK(!"Should not get here");  
    }  
    catch (std::invalid_argument& e) {  
        const std::string expected  
            = "divide: denominator cannot be zero";  
        BOOST_CHECK(e.what() == expected);  
    }  
}
```


Add Travis CI


- Will do whatever you want after a 'git push'
- Plays well with GitHub
- Free for FOSS



Automate the running
of your tests through
a makefile or similar tool

.travis.yml

```
sudo: true  
language: cpp  
compiler: gcc  
addons:  
  apt:  
    packages: libboost-all-dev  
  
script:  
  - ./build.sh  
  - ./build_test.sh
```



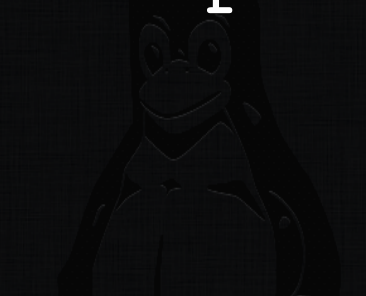
build.sh

```
#!/bin/bash  
qmake adder.pro  
make  
./adder
```



build_test.sh

```
#!/bin/bash  
qmake adder_test.pro  
make  
./adder_test
```



```
▶ 1 Worker information
▶ 6 Build system information
100
▶ 101 $ export DEBIAN_FRONTEND=noninteractive
▶ 107 $ git clone --depth=50 --branch=master
▶ 117 Installing APT Packages (BETA)
414 $ export CXX=g++
415 $ export CC=gcc
416 $ gcc --version
417 gcc (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
418 Copyright (C) 2011 Free Software Foundation, Inc.
419 This is free software; see the source for copying conditions. There is NO
420 warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
421
422 $ ./build.sh
423 make -f Makefile.Release
424 make[1]: Entering directory `/home/travis/build/richebilderbeek/travis_qmake_gcc_cpp98_boost_test'
425 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -
I/usr/include/qt4 -Irelease -o release/main.o main.cpp
426 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -
I/usr/include/qt4 -Irelease -o release/my_functions.o my_functions.cpp
427 g++ -m64 -Wl,-O1 -o travis_qmake_gcc_cpp98_boost_test release/main.o release/my_functions.o -L/usr/lib/x86_64-linux-gnu -
lpthread
428 make[1]: Leaving directory `/home/travis/build/richebilderbeek/travis_qmake_gcc_cpp98_boost_test'
429 42
430
431
432 The command "./build.sh" exited with 0.
433 $ ./build_test.sh
434 make -f Makefile.Release
435 make[1]: Entering directory `/home/travis/build/richebilderbeek/travis_qmake_gcc_cpp98_boost_test'
436 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -
I/usr/include/qt4 -Irelease -o release/main_test.o main_test.cpp
437 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -
I/usr/include/qt4 -Irelease -o release/my_functions_test.o my_functions_test.cpp
438 g++ -m64 -Wl,-O1 -o travis_qmake_gcc_cpp98_boost_test_test release/my_functions.o release/main_test.o release/my_functions_test.o
-L/usr/lib/x86_64-linux-gnu -lboost_unit_test_framework -lpthread
439 make[1]: Leaving directory `/home/travis/build/richebilderbeek/travis_qmake_gcc_cpp98_boost_test'
440 Running 1 test case...
441
442 *** No errors detected
443
444
445 The command "./build_test.sh" exited with 0.
446
447 Done. Your build exited with 0.
```

Top ▲



Conclusion

- Boost.Test is a testing framework, that
 - Makes it easy to add new tests
 - Makes it hard to forget adding tests
 - Has cumbersome exception testing
 - Can easily be used by Travis CI