Faculty of Science, Engineering & Technology

Object Oriented Programming

C++ Setup: Installing Netbeans, GNU Compilers, and CppUnit

Please read through this entire guide once before starting.

The steps that you follow will depend on which operating system you are using.

This guide is divided into the following sections:

SWINBURNE UNIVERSITY OF TECHNOLOGY

Installing Netbeans and the GNU Compiler Collection

1. Download and install Netbeans 8.0.1 with the C/C++ bundle
   https://netbeans.org/downloads
   - On Mac OS, you may have to install the latest Java Development Kit (JDK) before you can install Netbeans

2. Install and set up the GNU Compiler Collection
   https://netbeans.org/community/releases/80/cpp-setup-instructions.html#compilers
   - On Mac OS, if you are using XCode 5.0 or later, you may have to install the Command Line Tools by running `xcode-select --install`
   - Most Linux variants come with the GNU compiler collection already installed.
   - On Windows, install Minimalist GNU for Windows (MinGW)
     https://netbeans.org/community/releases/80/cpp-setup-instructions.html#mingw

3. Verify that you can compile and run a simple program
   - Open NetBeans and select **File** -> **New Project**.
   - At **Step 1. Choose Project**
     - under **Categories:** select **C/C++**
     - under **Projects:** select **C/C++ Application**
     - click **Next >**
   - At **Step 2. Project Name and Location**
     - change the **Project Name** to **Rover**
     - click **Finish**
   - In the **Projects** tab, expand the **Source Files** sub-folder and double click **main.cpp**
   - Change **main.cpp** to the following

```
#include <iostream>
using namespace std;

int main(int argc, char** argv)
{
    cout << "Hello World" << endl;
    return 0;
}
```

   - Run -> Run Project ... if it doesn't work, try some of the Troubleshooting tips at
     https://netbeans.org/community/releases/80/cpp-setup-instructions.html#ts
     If you are still stuck, post a question on the Discussion Board or see your tutor.

Installing CppUnit and Testing the Netbeans - CppUnit Interface

4. Install CppUnit:
   - On Mac: download and install Homebrew, then install CppUnit using **brew install cppunit** (from the command line).
   - Many Linux variants come with CppUnit already installed.
   - For Windows, install as follows:
     - Download the CppUnit source code http://downloads.sourceforge.net/cppunit/cppunit-1.12.1.tar.gz
     - Open a command line terminal and run the command "sh" and type the followings:

```
tar -xzvf cppunit-1.12.1.tar.gz
cd cppunit-1.12.1
./configure --prefix=/c/mingw
make
make install
```

5. In the Rover project that is currently open in Netbeans, create a CppUnit Test
   - select **File** -> **New File**.
   - At **Step 1. Choose File Type**
     - under **Categories** select **C/C++ Tests**
     - under **File Types** select **CppUnit Test**
   - In the **New CppUnit Test** pop-up window, create a test named **RoverTest** with a test class named **TestRover** and a test runner file named **test_rover.cpp**.

Be sure to set **Folder** to an empty string. The default is to create a sub-folder named **tests**; however, this causes trouble down the road. Your NewCppUnit Test pop-up window

6. Try to build by selecting **Run** -> **Test Project** (or **Ctrl-F6**).

   On Mac OS, the build will most likely fail; the log should include an error message like

```
/bin/sh: cppunit-config: command not found
```

   This can be fixed by opening a Terminal and creating a symbolic between the location where **homebrew** installed CppUnit and the location where Netbeans expects to find it:

```
sudo ln -s /usr/local/bin/cppunit-config /usr/bin
```

   If the build succeeds, then you should see that one out of the two tests failed.
   You are now ready to go!

Trouble-shooting and Work-arounds

If you cannot get Netbeans to work with CppUnit, don't give up. Ask for help from class mates, your tutor, and on the Blackboard discussion board. If this doesn't help, then you can fall back on the following work-around:

1. Wherever you are asked to create a unit test using CppUnit, simply add a new method to your own unit testing class by hand. For example, at the start of Pass Task 15, forget about using the CppUnit framework and simply create the following class:

```
class TestRover
{
public:
  void testAttach ();
}
```

2. Wherever you need to use CPPUNIT_ASSERT, simply use the C language built-in `assert` function; e.g.

```
#include <assert.h>

[...]


void TestRover::testAttach()
{
[...]
    assert( rover->deviceCount() == 1 );
}
```

3. To run the unit tests, simply create another executable (i.e. a source code file that defines the main() function) and use it to call the methods of the unit testing class; e.g.

```
#include “TestRover.h”

int main ()
{
  TestRover* test = new TestRover;
  test->testAttach ();
  [...]
  cout << “All unit tests completed successfully” << endl;
  return 0;
}
```