Boost-Python

First take

Boost.Python

David Abrahams

Stefan Seefeld

Synopsis

Welcome to Boost.Python, a C++ library which enables seamless interoperability between C++ and the Python programming language. The library includes support for:

- References and Pointers
- Globally Registered Type Coercions
- Automatic Cross-Module Type Conversions
- Efficient Function Overloading
- C++ to Python Exception Translation
- Default Arguments
- Keyword Arguments
- Manipulating Python objects in C++
- Exporting C++ Iterators as Python Iterators
- Documentation Strings

The development of these features was funded in part by grants to `Boost Consulting` from the Lawrence Livermore National Laboratories and by the Computational National Laboratories.

Contents

Boost.Python Tutorial

Joel de Guzman

David Abrahams

Table of Contents

QuickStart

The Boost Python Library is a framework for interfacing Python and C++. It allows you to quickly and seamlessly expose C++ classes functions and objects to Pyth
C+ [illegible] It is designed to wrap C++ interfaces non-intrusively, so that you should not have to change the C++ code at all in order to wrap it, making Boost.Pyth
The library's use of advanced metaprogramming techniques simplifies its syntax for users, so that wrapping code takes on the look of a kind of declarative interface

QuickStart

Hello World

Following C/C++ tradition, let's start with the "hello, world". A C++ Function:

```
char const* greet()
{
   return "hello, world";
}
```

can be exposed to Python by writing a Boost.Python wrapper:

```
#include <boost/python.hpp>

BOOST_PYTHON_MODULE(hello_ext)
{
    using namespace boost::python;
    def("greet", greet);
}
```

That's it. We're done. We can now build this as a shared library. The resulting DLL is now visible to Python. Here's a sample Python session:

```
>>> import hello_ext
>>> print hello_ext.greet()
hello, world
```

Next stop... Building your Hello World module from start to finish...

Last revised: December 05, 2018 at 20:03:11 GMT

```
// Copyright Ralf W. Grosse-Kunstleve 2002-2004. Distributed under the Boost
// Software License, Version 1.0. (See accompanying
// file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

#include <boost/python/module.hpp>
#include <boost/python/def.hpp>
#include <string>

namespace { // Avoid cluttering the global namespace.

  // A couple of simple C++ functions that we want to expose to Python.
  std::string greet() { return "hello, world"; }
  int square(int number) { return number * number; }
}

namespace python = boost::python;

// Python requires an exported function called init<module-name> in every
// extension module. This is where we build the module contents.
BOOST_PYTHON_MODULE(getting_started1)
{
    // Add regular functions to the module.
    python::def("greet", greet);
    python::def("square", square);
}
```

http://mrbook.org/blog/tutorials/make/ on standard UNIX

```
# location of the Python header files

PYTHON_VERSION = 2.7
PYTHON_INCLUDE = /usr/include/python$(PYTHON_VERSION)

# location of the Boost Python include files and library

BOOST_INC = /usr/include
BOOST_LIB = /usr/lib

# compile
TARGET = hello_ext

$(TARGET).so: $(TARGET).o
   g++ -shared -Wl,--export-dynamic $(TARGET).o -L$(BOOST_LIB) -lboost_python-$
(PYTHON_VERSION) -L/usr/lib/python$(PYTHON_VERSION)/config -lpython$(PYTHON_VERSION) -o $
(TARGET).so

$(TARGET).o: $(TARGET).C
   g++ -I$(PYTHON_INCLUDE) -I$(BOOST_INC) -fPIC -c $(TARGET).C
```

on macs

```
   clang++ -shared -undefined dynamic_lookup $(TARGET).o -L$(BOOST_LIB) -lboost_python -L/
usr/lib/python$(PYTHON_VERSION)/config -lpython$(PYTHON_VERSION) -o $(TARGET).so

$(TARGET).o: $(TARGET).C
   clang++ -I$(PYTHON_INCLUDE) -I$(BOOST_INC) -fPIC -c $(TARGET).C
```

```
from distutils.core import setup
from distutils.extension import Extension

greetings_ext = Extension(
    'greetings_ext',
    sources=['greetings.cpp'],
    libraries=['boost_python27-mt'],
)

setup(
    name='greetings-test',
    version='0.1',
    ext_modules=[greetings_ext])
```

python setup.py build_ext —inplace

http://mrbook.org/blog/tutorials/make/