youalab
youa tech blog

使用python做c库的测试的一种途径

Posted on 2010/09/22 by youaadmin

前言

测试函数库，与测试模块就较大的区别，case复用率小，代码量大，还有一堆复杂的编译，从而使测试函数库相对变得慢。

往往建立一个函数库，是因为多个模块的需要，库有问题，则所有模块都要重新编译提测，所以库的测试就会变成我们测试过程中的单点，从而降低整个测试进度。

当然一个好的框架必然可以加快测试，如cppunit,gtest等.

试想用脚本语言做函数库的测试，显然可以大大加快测试速度，并且可以减小后期维护case的成本。但用脚本做库测试有多个问题,怎么可以在脚本中调用c/c++的接口,怎么解决数据类型的转化问题,等.

python是一个语法组织很清晰的强类型解释语言，被称为胶水语言，和c很好的兼容，有着大量优秀的库可以被直接调用以高效地完成不同需求的工作，其中有一个模块ctypes,还有其成熟的测试框架unittest,可以为我们测试c库提供一种新的路径.

ctypes简单使用介绍

ctypes是python的一个非常有用的模块，通过它我们可以调用动态函数库，它还定义了一些能兼容c的数据类型，很大程度上解决了测试在数据类型上的问题。

ctypes详细文档：http://docs.python.org/library/ctypes.html?highlight=ctypes#module-ctypes

简单介绍一下

我们可以使用CDLL加载我们需要的动态库，如果是静态库则需要编译成动态库加载，所以静态库需要使用-fPIC编译

```
#!/usr/bin/env python

from ctypes import *

libc = CDLL( "libc.so.6" )
printf = libc.printf
printf( "Hello, world!n" )
```

运行上面这段脚本，输出

```
[songhuaqing@tc-sierra64.tc.baidu.com testctypes]$ python printf.py
Hello, world!
```

CDLL函数用于加载动态库

再来一个例子

```
#!/usr/bin/env python

from ctypes import *

libc = CDLL( "libc.so.6" )
printf = libc.printf
```

```
strcpy = libc.strcpy
s = c_char_p( "Hello, world!n" )
printf( "%s", s )
t = create_string_buffer( 32 )
strcpy( t, s )
printf( "%s", t )
```

运行上面这段脚本，输出

```
[songhuaqing@tc-sierra64.tc.baidu.com testctypes]$ python strcpy.py
Hello, world!
Hello, world!
```

这个例子中c_char_p是ctypes中字符串指针，相当于c中的char *，其他类型详见ctyeps的文档，我们知道python中的字符串内容是不能被修改的，而有些c函数需要传入一个或多个pointer，修改对应地址的内容作为函数的返回，这个例子中的create_string_buffer函数允许在python中创建可以被修改一段buff。

对于c中的复杂的结构体，ctypes也有对应的机制如：

```
struct point{
    int x,y;
};
```

对应在python的类中可以为：

```
class POINT(Structure):
    _fields_ = [("x", c_int),("y", c_int)]
```

对于被函数库中大量的结构体，我们可以使用gccxml，ctypeslib自动生成能在python中使用的类

安装使用gccxml, ctypeslib

gccxml是解析c/c++声明的开源工具，ctypeslib基于gccxml通过xml生成python的声明

gccxml的官方网站为：http://www.gccxml.org/HTML/Index.html

目前，gccxml官方的Release版本太老了，建议直接下载源码编译

(注：安装gccxml需要cmake编译工具，下载地址：http://www.cmake.org/)

ctypeslib安装相对比较简单，先export PYTHONPATH=your python path

```
$ easy_install --install-dir=/home/songhuaqing/python/ ctypeslib==dev
```

(Python中的easy_install,类似Php中的pear,Ruby中的gem,Perl中的cpan,http://peak.telecommunity.com/DevCenter/EasyInstall)

现在我们可以通过ctypeslib生成python的声明

```
$ python -m ctypeslib.h2xml test.h -o test.xml -q -I ./
$ python -m ctypeslib.xml2py test.xml -o test.py
```

h2xml：解析test.h生成xml，-I为头文件所在dir，对于较复杂的头文件，我们可以从rd的Makefile中得到；

xml2py：解析xml生成py的声明

原始的test.h

```
#ifndef  __TEST_H_
```

```
#define  __TEST_H_

typedef struct _point{
    int x, y;
}point;

typedef struct _rect{
    point a, b;
}rect;

#endif  //__TEST_H_
```

产出：test.py

```
from ctypes import *

class _point(Structure):
    pass
_point._fields_ = [
    ('x', c_int),
    ('y', c_int),
]
point = _point
class _rect(Structure):
    pass
_rect._fields_ = [
    ('a', point),
    ('b', point),
]
rect = _rect
__all__ = ['_rect', '_point', 'rect', 'point']
```

静态库编译成动态库

目前我们的测试以静态库居多(至少在eb部门)，如果需要使用python来做测试，需要将静态库编译成动态库。

接着上面的例子，我们有test.c，实现了一个计算矩形面积的函数

```
#include <math.h>
#include "test.h"

int area( rect s ){
    return abs( s.a.x - s.b.x ) * abs( s.a.y - s.b.y );
}
```

将其编译成libtest.a(加-fPIC选项),然后编译成libtest.so

```
[testctypes]$ gcc -fPIC -c test.c
[testctypes]$ ar cr libtest.a test.o
[testctypes]$ gcc -shared -fPIC -Wl,--whole-archive libtest.a -Wl,--no-whole-archive -Wl,-soname -Wl,libtest.so -o libtest.so
```

测试驱动

这里我们简单写个python脚本测试刚才libtest.so中计算矩形面积的函数，用到nuittest测试框架，nuittest详细文档：http://docs.python.org/library/unittest.html?highlight=unittest#module-unittest

mytest.py

```
#!/usr/bin/env python
# encoding: iso-8859-1

import unittest
from ctypes import * #加载ctypes
from test import * #加载test.py

class Myclass( unittest.TestCase ):

    def setUp( self ):
        pass

    def tearDown( self ):
        pass

    def test1( self ):
        rc = rect( point(0, 0), point(1, 1) )
        self.assert_( area( rc ) == 1 )
```

```
if __name__ == "__main__":
    mylib = CDLL( "./libtest.so" )
    area = mylib.area
    unittest.main()
```

执行脚本：

```
$ python mytest.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s
```

OK

总结

文本简单的介绍了下使用python做c库测试的一种路径，基本步骤为：

1，将函数库编译成python能加载的动态库（动态库省略这一步）

2，使用gccxml和ctypeslib转化数据结构，相当于python版的头文件

3，使用上述两步的产出，开发测试脚本

总结下优缺点

优点：

1，和一般脚本一样，开发效率高，速度快，减少测试单点对整个测试进度的影响

2，解释性语言，case可读性强

3，后期case维护成本小（增加/修改测试数据方便）

4，python优秀的库，如unittest等

缺点：

1，测试静态库需要有-fPIC编译，并需要装化为动态库测试

2，被测库的数据结构需要转化

3，不适合用于性能测试

根据被测库特点选者不同的测试方法，对于库比较简单，修改较为频繁，使用python是一个不错的选者。

参考资料

http://docs.python.org/library/ctypes.html?highlight=ctypes#module-ctypes

http://pypi.python.org/pypi/ctypeslib/

http://www.gccxml.org/HTML/Index.html

http://www.cmake.org/

http://peak.telecommunity.com/DevCenter/EasyInstall

http://docs.python.org/library/unittest.html?highlight=unittest#module-unittest

This entry was posted in C/C++, Test, python. Bookmark the permalink.

youalab
Proudly powered by WordPress.