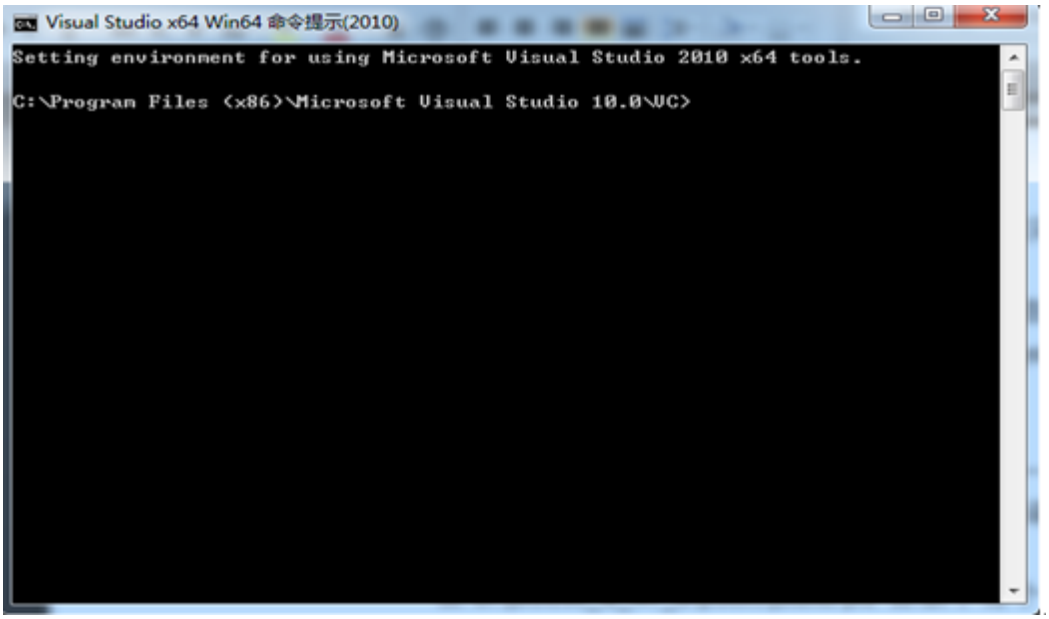
vs2010 x64 下 boost Python 编译

（1）、安装好Python ， 下载boost 的源码，解压到一个目录下，例如d:\
D:\boost_1_47_0

（2）、按照网上很多人说的，首先编译boost 的编译器 bjam.exe， bjam的源码已经被包含在boost 的源码中，

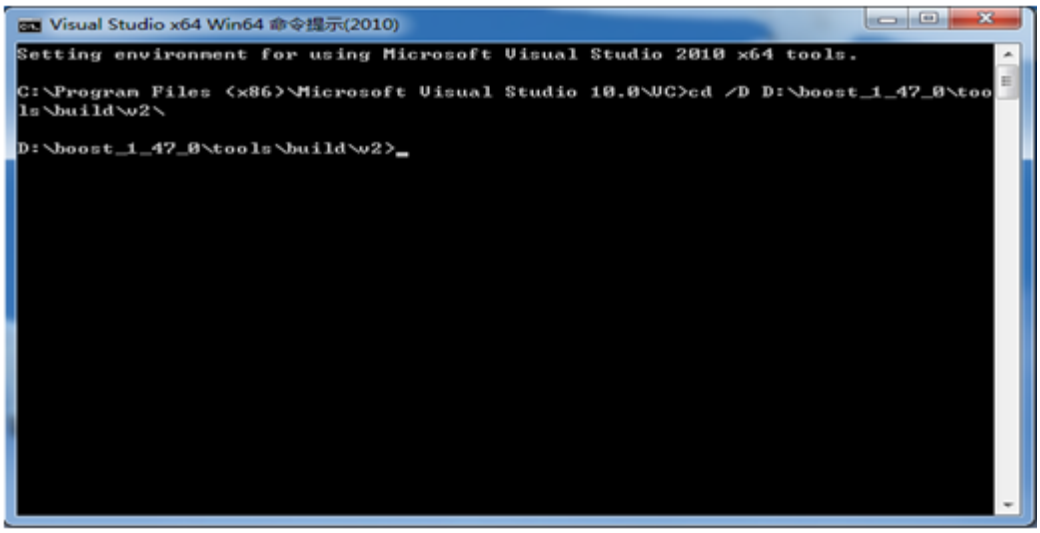
在D:\boost_1_47_0\tools\build\v2 目录下有个 bootstrap.bat ，

运行vs 2010 x64 的命令提示符工具，如图:

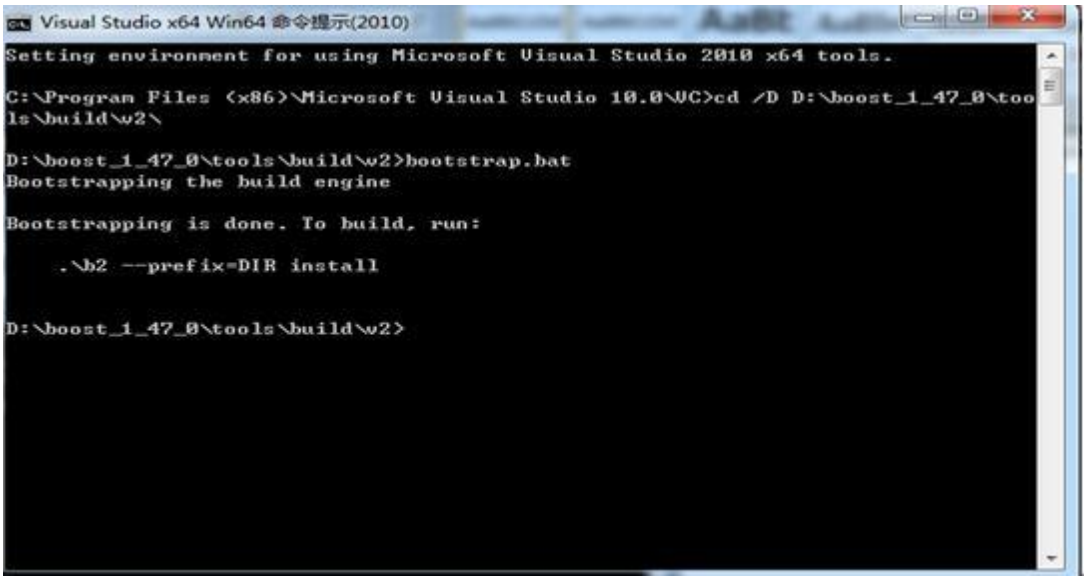


执行命令 : cd /D D:\boost_1_47_0\tools\build\v2\

将命令提示符窗口定位到 D:\boost_1_47_0\tools\build\v2 目录下，



执行命令： bootstrap.bat 开始 bjam 的编译，


Visual Studio x64 Win64 命令提示(2010)
Setting environment for using Microsoft Visual Studio 2010 x64 tools.
C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC>cd /D D:\boost_1_47_0\too
ls\build\v2\
D:\boost_1_47_0\tools\build\v2>bootstrap.bat
Bootstrapping the build engine
Bootstrapping is done. To build, run:
.\b2 --prefix=DIR install
D:\boost_1_47_0\tools\build\v2>

编译bjam完成后如上图所示
在 D:\boost_1_47_0\tools\build\v2\engine\bin.ntx86_64 目录下找到 bjam.exe ，
（3）、修改bjam 的配置文件，在D:\boost_1_47_0\tools\build\v2 用文本编辑器打开user-config.jam 这个文件，
修改其中的设置，将
Configure specific msvc version (searched for in standard locations and PATH).
using msvc : 8.0 ;
改为（因为这里用的是vs 2010）：
Configure specific msvc version (searched for in standard locations and PATH).
using msvc : 10.0 ;
将：

Python configuration.

Configure specific Python version.
using python : 3.1 : /usr/bin/python3 : /usr/include/python3.1 : /usr/lib ;
改为（python2.7 的安装目录和python 版本设置）设置python 的目录和版本等信息：

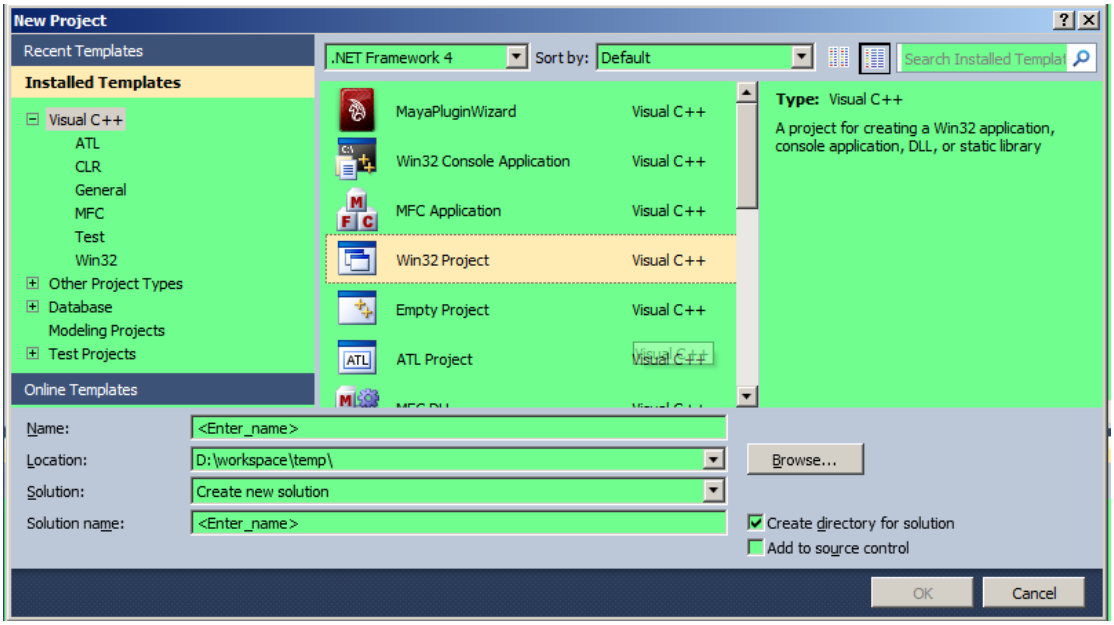
Python configuration.

Configure specific Python version.
using python : 2.7 : C:\\Python27 : C:\\Python27\\include : C:\\Python27\\libs ;
（4）、将刚才的bjam.exe 拷贝到 D:\boost_1_47_0\ 下，准备boost 库的编译。

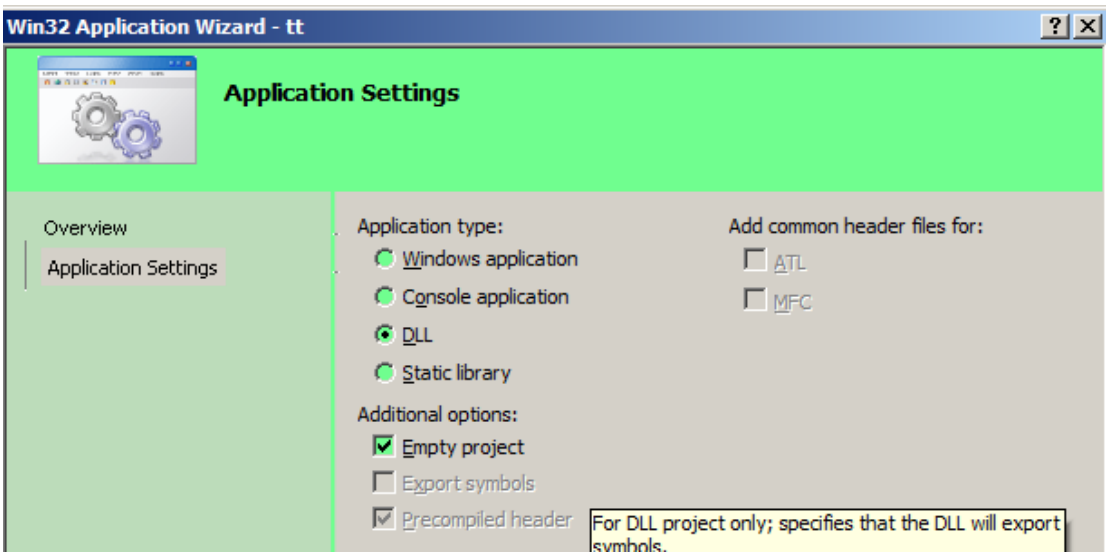
将命令提示符定位到 D:\boost_1_47_0\ 下
执行 bjam 编译命令:
能正确生成 动态库版本的命令写法：
x64 debug
bjam --with-python --prefix=d:\boost stage toolset=msvc-10.0 variant=debug link=shared address-model=64 threading=multi runtime-link=shared install
x64 release
bjam --with-python --prefix=d:\boost stage toolset=msvc-10.0 variant=release link=shared address-model=64 threading=multi runtime-link=shared install

Python中如何调用C++写的扩展模块

1、 vs2010中新建一个空的DLL工程：

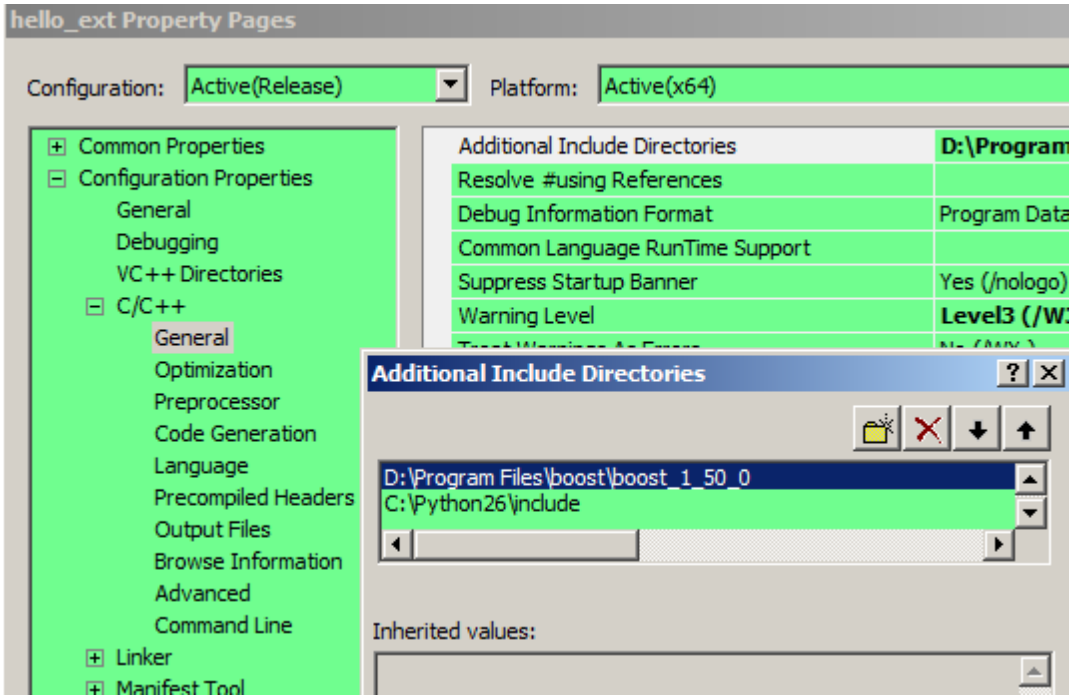


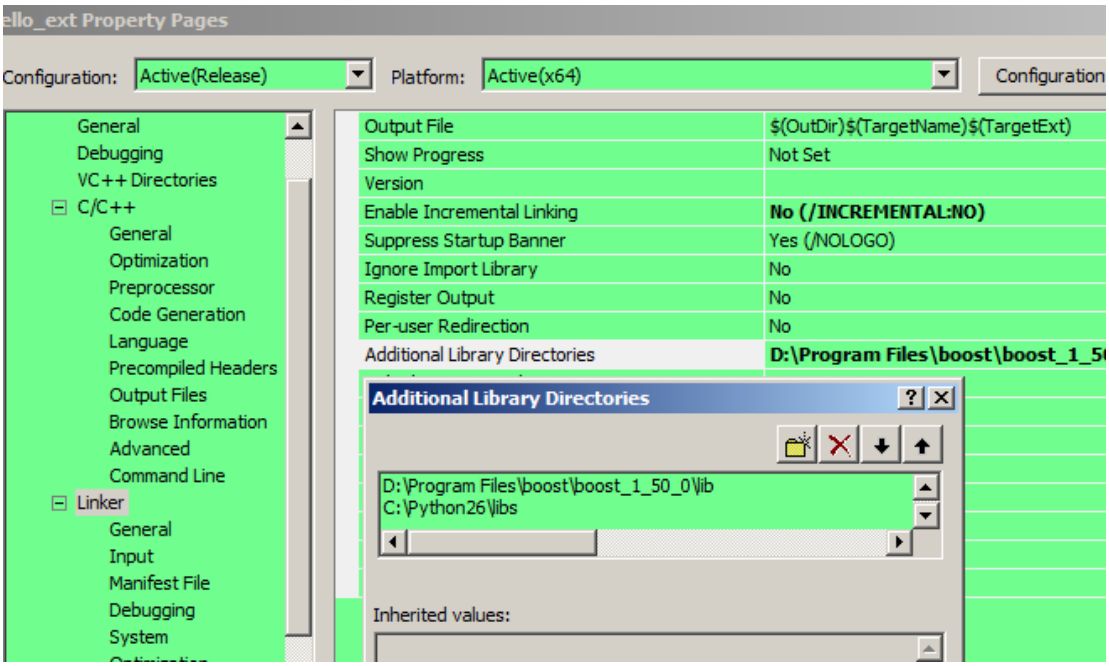
a.

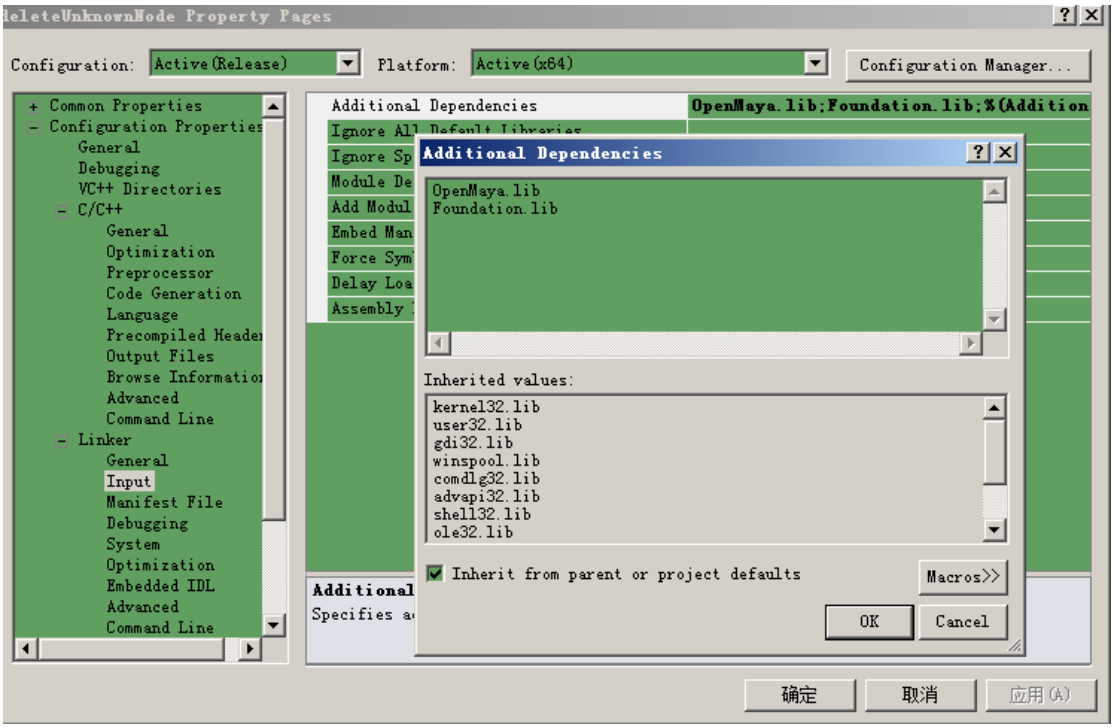


b.

2、 设置项目属性：


hello_ext Property Pages
Configuration: Active(Release)
Platform: Active(x64)
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
Command Line
Linker
Manifest Tool
Additional Include Directories
Resolve #using References
Debug Information Format
Common Language RunTime Support
Suppress Startup Banner
Warning Level
Additional Include Directories
D:\Program Files\boost\boost_1_50_0
C:\Python26\include
Inherited values:


Configuration: Active(Release)
Platform: Active(x64)
Configuration
General
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
Command Line
Linker
General
Input
Manifest File
Debugging
System
Output File
Show Progress
Version
Enable Incremental Linking
Suppress Startup Banner
Ignore Import Library
Register Output
Per-user Redirection
Additional Library Directories
$(OutDir)$(TargetName)$(TargetExt)
Not Set
No (/INCREMENTAL:NO)
Yes (/NOLOGO)
No
No
No
Additional Library Directories
D:\Program Files\boost\boost_1_50_0\lib
C:\Python26\libs
Inherited values:


eleteUnknownNode Property Pages
Configuration: Active(Release)
Platform: Active(x64)
Configuration Manager...
+ Common Properties
- Configuration Properties
General
Debugging
VC++ Directories
- C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Header
Output Files
Browse Information
Advanced
Command Line
- Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Advanced
Command Line
Additional Dependencies
OpenMaya.lib;Foundation.lib;%(Addition
Ignore All Default Libraries
Ignore Sp
Module De
Add Modul
Embed Man
Force Sym
Delay Loa
Assembly
Additional
Specifies a
Additional Dependencies
OpenMaya.lib
Foundation.lib
Inherited values:
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
advapi32.lib
shell32.lib
ole32.lib
Inherit from parent or project defaults
Macros>>
OK
Cancel
确定
取消
应用(A)

新建hello.cpp文件,把下面代码拷进去：

```
#include <boost/python/module.hpp>
#include <boost/python/def.hpp>
char const* greet()
{
return "hello, world";
}
BOOST_PYTHON_MODULE(hello_ext)
{
using namespace boost::python;
def("greet", greet);
}
```

8. 编译，生成，把输出的dll改名为hello_ext.pyd，

9. 将“D:\boost\boost_1_50_0\lib;”添加到系统的“PATH”环境变量
boost_python-vc90-mt-gd-1_48.dll位于“D:\boost\boost_1_50_0\lib”

10. 把hello_ext.pyd和拷贝到python的工作目录下
在python 工作目录下新建hello.py编写如下代码：

```
import hello_ext
hello_ext.greet()
```

For MAYA：
deleteUnknownNode.cpp:

```
#ifdef WIN32
#define NT_PLUGIN
#endif

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

#include <maya/MGlobal.h>
#include <maya/MString.h>

void delete_unknown_node()
{
    MString cmd;
    cmd="{";
    cmd+="string $node,$nodes[]=`ls -type unknown -type unknownDag -type
unknownTransform`;";
    cmd+="for($node in $nodes){";
    cmd+="lockNode -lock off $node;";
    cmd+="if(catch(`delete $node`)){";
    cmd+="  print (\"delete unknown node error: \" + $node);";
```

```
    cmd+="}";
    cmd+="else{";
    cmd+="  print (\"delete unknown node success: \" + $node);";
    cmd+="}";
    cmd+="}";
    cmd+="}";

    MGlobal::executeCommand(cmd);
}

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

BOOST_PYTHON_MODULE(deleteUnknownNode)
{
    using namespace boost::python;
    def("delete_unknown_node", delete_unknown_node);
}
```