

自动封装C/C++库

September 19, 2009

这里介绍一个可以自动为道语言封装C/C++库的工具，此工具一般被包括在道语言发布的tools目录下，名为autobind.dao。

此工具为一个由道语言本身开发的脚本程序autobind.dao，它可带以下命令行参数：

- **inputfiles** : 输入文件，由逗号分割的头文件列表（不行空白字符）。这些头文件将被此工具解析，并生成封装代码。如，

```
mgl_data.h,mgl_font.h,mgl.h,mgl_ab.h,mgl_zb.h
```

如果这些头文件还依赖于其他不用封装的头文件，那些头文件也可被放到inputfiles参数里，放在前部，并用冒号和其他需要封装的头文件分开。

```
mgl_data.h:mgl_font.h,mgl.h,mgl_ab.h,mgl_zb.h
```

这种方式可用来将比较大的库分割成更小的模块封装。如这里mgl_data.h被可以被独立封装并被编译为DaoMglData模块（含文件后缀so或dll），其他文件以分开但依赖于mgl_data.h的方式被封装（并被编译为DaoMGL）。那么当模块DaoMGL被载入时，它将需要用到DaoMglData模块：

```
load DaoMglData;  
load DaoMGL require DaoMglData;
```

- **wrap_name** : 模块封装名。缺省时，它跟inputfiles参数里的第一个文件名一样，如果想用其他名字，需要设定此参数，如在前面的例子里可用wrap_name=MGL。
- **lang** : 头文件里源代码的语言。目前只支持c和cpp。

- **fixing**: 一个包含字符串模式的文本文件，被用了处理被封装库所特有的代码格式。

在此文件里，每行可以包含一对字符串模式，每个模式由单引号引起来，每对由制表符分割。每对模式的第一个模式必须是道语言正则表达式，第二个是一个普通的字符串，但可以包含到第一个模式里的模式组的向后引用。对于每个头文件，此工具将使用第一个模式查找匹配的代码部分，并将它们转换为第二模式里的字符串，并替换其中的向后引用。如果要限定某对模式仅应用于某个头文件里，可在此模式对的上一行写上该头文件的名称。

这里是DaoVTK模块的fixing文件里摘取的几行：

```
'VTK_%w+_EXPORT'      ''
'VTK_BYTE_SWAP_DECL %s* %b() %s* ;'      ''
'([\n]*) vtkSetMacro %s*%(%s*(%w+) %s*, %s*(%w+) %s*)'      '%1 virtual void Set%2(%3
_arg)'
```

```
vtkInstantiator.h
'static [^;]* CreateFunction [^;]*;'      ''
```

```
vtkOStreamWrapper.h
'static [^;]* UseEndl [^%{]}* %b{'      ''
```

```
vtkLightKit.h
'static [^;]* GetSubType [^%{]}* %b()'      ''
```

在此例中，头两行表示将某些代码移除，第三第四行告诉此工具如何将相应于那两个模式的宏展开！其他模式则被限定到某些文件上。

- **dir_input**: 输入头文件所在的目录；
- **dir_output**: 封装代码的输出目录；
- **dir_inc**: include目录；在生成的封装代码里，它将是那些输入头文件的前缀目录。

例如，

```
inputfiles=gl.h dir_input=/usr/include/GL dir_inc=GL
```

那么，在生成的封装代码里gl.h头文件将以如下方式被包括：

```
#include<GL/gl.h>
```

- **include**: 对于依赖于其他模块的封装，它是那些被依赖的模块封装名。

例如在封装VTK5时，其公共部分被封装为 `wrap_name=vtkCommon`，而跟filtering相关的部分则被封装为 `wrap_name=vtkFiltering`。因为`vtkFiltering`依赖于`vtkCommon`，那么在封装`vtkFiltering`时，命令行参数`include`应被设为`include=vtkCommon`。