

# Automatic Binding of C/C++ Libraries

September 19, 2009

It is possible to use an automatic tool to generate wrapping of C/C++ libraries directly from their header files. This tool is normally included in Dao releases as *autobind.dao* under the fold of *tools*. This document will briefly explain how to use this tool.

The arguments of *autobind.dao* :

- **inputfiles** : the list of header file names delimited by comma without white spaces. these header files will be parsed by the tool to generate wrapping codes.

mgl\_data.h,mgl\_font.h,mgl.h,mgl\_ab.h,mgl\_zb.h

If these header files are dependent on other headers, which however should be excluded from wrapping, one can also include them in the parameter, and separate them by colon from the other files:

mgl\_data.h:mgl\_font.h,mgl.h,mgl\_ab.h,mgl\_zb.h

This way can be used to divide a big library into smaller parts and wrap them separately. Here one may wrap *mgl\_data.h* separately, which is compiled into *DaoMglData* (with suffix *so* or *dll*), and wrap the rest dependently on *mgl\_data.h* and compile the wrapping as *DaoMGL*, so that, *DaoMglData* can be used independently, while using *DaoMGL* has to require *DaoMglData*. Namely, when one load *DaoMGL*, one has to use,

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

- **wrap\_name** : the wrapping name. By default, it is taken as the name of the first wrapped file in parameter **inputfiles**. One may use this argument to set a preferred wrapping, for example, in above example, one may set *wrap\_name=MGL*.

- **lang** : the language of the source codes in the header files. Currently it only supports *c* and *cpp* .
- **fixing** : a text file containing patterns which should be used to handle library-specific codes.

In each line of this file, there should be a pair of patterns enclosed in two pairs of ' and delimited by a tab. In each pair of patterns, the first should be a pattern written as Dao regular expression, and the second should be a normal string that may contain back references to the groups in the first pattern. For each header files, this tool will search the codes that match the first pattern, and transform them into the second string with substitutions for the back references. If one wants a pair of patterns should be applied certain file, one may put the file name in a single line above that line.

Here is an example taken from the **fixing** of DaoVTK module,

```

'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)'
'([\n]*) vtkGetMacro %s*%(%s*(%w+)%)s*,%s*(%w+)%)s*%'      '%1 virtual %3 Get%2()'

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

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

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

```

In this example, the first two lines indicate that some codes should be removed, and the third and fourth tell the tool how to expand two macros! The other patterns are applied only to certain files.

- **dir\_input** : the fold of the input header files.
- **dir\_output** : the output fold for the wrapped codes.
- **dir\_inc** : include directory, which is the directory that should prefix the header files in the generated codes.

For example, one may has,

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

then the *gl.h* header file will be included in the generated codes as

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

- **include** : for dependent wrapping, it should include the wrapping names of the modules on which this wrapping is dependent on.

For example, when wrapping VTK5, the common part is wrapped as **wrap\_name=vtkCommon** , the part for filtering is wrapped as **wrap\_name=vtkFiltering** , which is dependent on **vtkCommon** , then the **include** should be set as **include=vtkCommon** when wrapping **vtkFiltering** .