**1. 安装（Windows）**

下载：http://www.swig.org/download.html

解压把swig.exe的地址写入到环境变量的环境变量的Path变量中。

**2. 例子(c语言)**

**#2.1 用c语言编写头文件和源文件为**

/\* File: example.h \*/

int fact(int n);

[复制代码](javascript:void(0);)

/\* File: example.c \*/

#include "example.h"

int fact(int n) {

if (n < 0){ /\* This should probably return an error, but this is simpler \*/

return 0;

}

if (n == 0) {

return 1;

}

else {

/\* testing for overflow would be a good idea here \*/

return n \* fact(n-1);

}

}

[复制代码](javascript:void(0);)

**#2.2 写swig模块写一个文件**

[复制代码](javascript:void(0);)

/\* File: example.i \*/

%module example

%{

#define SWIG\_FILE\_WITH\_INIT

#include "example.h"

%}

int fact(int n);

[复制代码](javascript:void(0);)

**#2.3 为了建python模块，利用-python参数执行swig**

|  |  |
| --- | --- |
| 1 | swig -python example.i |

执行完命令后生成两个不同的文件：example\_wrap.c和example.py。

自动生成文件名的原则：生成的c文件名与写的c文件名有关（例如写的c文件名为example.c则生成example\_wrap.c）；生成的python文件即.i文件中%module后面的名字。

**#2.4 利用distutils生成动态库**

python自带一个distutils工具，可以用它来创建python的扩展模块。使用它也很简单，只需要先定义一个配置文件，通常是命名为setup.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | """  setup.py  """    from distutils.core import setup, Extension      example\_module = Extension('\_example',                             sources=['example\_wrap.c', 'example.c'],                             )    setup (name = 'example',         version = '0.1',         author      = "SWIG Docs",         description = """Simple swig example from docs""",         ext\_modules = [example\_module],         py\_modules = ["example"],         ) |

**注**：头文件和源文件都是example.\*，那么setup.py脚本中Extension的参数必须为“\_example”

**#2.5 编译**

|  |  |
| --- | --- |
| 1 | python setup.py build |

会在本目录下build/lib\*/下生成\_example.pyd模块。

或者用下面方法将其安装到Python中：

$ python setup.py install

可以直接使用，例如

|  |  |
| --- | --- |
| 1  2  3  4 | >>>import example  >>>example.fact(4)  24  >>> |

可以吧动态模块直接生成当前目录下

|  |  |
| --- | --- |
| 1 | python setup.py build\_ext --inplace |

**3. 例子（c++）**

和c一样，稍微区别

**#3.1 用c语言编写头文件和源文件为**

/\* File: example.h \*/

int fact(int n);

[复制代码](javascript:void(0);)

/\* File: example.cpp \*/

#include "example.h"

int fact(int n) {

if (n < 0){ /\* This should probably return an error, but this is simpler \*/

return 0;

}

if (n == 0) {

return 1;

}

else {

/\* testing for overflow would be a good idea here \*/

return n \* fact(n-1);

}

}

[复制代码](javascript:void(0);)

**#3.2 写swig模块写一个文件**

[复制代码](javascript:void(0);)

/\* File: example.i \*/

%module example

%{

#define SWIG\_FILE\_WITH\_INIT

#include "example.h"

%}

int fact(int n);

[复制代码](javascript:void(0);)

**#3.3 为了建python模块，利用-python参数执行swig**

|  |  |
| --- | --- |
| 1 | swig -c++ -python example.i |

执行完命令后生成两个不同的文件：example\_wrap.**cxx**和example.py。

**#3.4 利用distutils生成动态库**

python自带一个distutils工具，可以用它来创建python的扩展模块。使用它也很简单，只需要先定义一个配置文件，通常是命名为setup.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | """  setup.py  """    from distutils.core import setup, Extension      example\_module = Extension('\_example',                             sources=['example\_wrap.cxx', 'example.cpp'],                             )    setup (name = 'example',         version = '0.1',         author      = "SWIG Docs",         description = """Simple swig example from docs""",         ext\_modules = [example\_module],         py\_modules = ["example"],         ) |

**注**：头文件和源文件都是example.\*，那么setup.py脚本中Extension的参数必须为“\_example”

**#3.5 编译**

|  |  |
| --- | --- |
| 1 | python setup.py build\_ext --inplace |