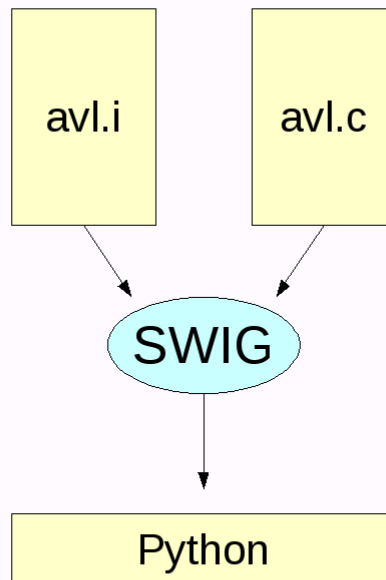


Combinando C/c++ con Python: SWIG *Simplified Wrapper and Interface Generator*

Jorge Andrés Holguín Duque

18 de enero de 2014

C Module and the Interface File



Índice

1. Swig	3
2. Usando y Compilando Swig	3

1. Swig

Para construir los módulos de extensión de Python, SWIG utiliza un enfoque por capas en el que las partes del módulo de extensión se definen en C y otras partes están definidas en Python. La capa C contiene las envolturas de bajo nivel mientras que el código Python se utiliza para definir las características de alto nivel.

Este enfoque por capas reconoce el hecho de que ciertos aspectos de la construcción de la extensión son consuman mejor en cada lenguaje (en lugar de tratar de hacer todo lo que sabemos en C o C++). Por otra parte, mediante la generación de código en dos lenguajes, se obtiene mucha más flexibilidad ya que se puede mejorar el módulo de extensión con código de apoyo en ambos lenguajes.

2. Usando y Compilando Swig

Supongamos que hemos definido un módulo SWIG como:

```
1  /* example.i */
2  %module ejemplo
3
4  %{
5  #define SWIG_FILE_WITH_INIT
6  #include "ejemplo.h"
7  %}
8
9  int fact(int n);
10
```

La línea 5 inserta una macro que especifica que el archivo C resultante debe ser construido como una extensión de Python, insertar el código de inicio del módulo. Este archivo i. envuelve el siguiente archivo C:

```
1  /* ejemplo.c */
2
3  #include "ejemplo.h"
4
5  int fact(int n) {
6      if (n < 0) { /* Devolvería un error. Pero hacer esto es mas simple */
7          return 0;
8      }
9      if (n == 0) {
10         return 1;
11     }
12     else {
13         /* Podríamos probar aquí si hay desbordamiento pero esto es solo un ejemplo*/
14         return n * fact(n-1);
15     }
16 }
17
```

Con el archivo de cabecera:

```
1  /* File: example.h */
2
3  int fact(int n);
4
```

Para construir este modulo en Python usamos la opción python”:

swig -python ejemplo.i

Y si es la construcción de una extensión para C++, solo hay que añadir la opción c++”:

```
swig -c++ -python ejemplo.i
```

Esto crea dos archivos diferentes: uno *ejemplo_wrap.c*, o *ejemplo_wrap.cxx* si es de C++, y un archivo de origen *example.py* Python. *ejemplo_wrap.c* es el archivo que se ha creado para la extensión. Estas funciones actúan como un pegamento entre los dos lenguajes. Ahora estos archivos los tenemos que compilar y un archivo con extensión .so (archivos objeto), una vez creado esto, lo último que hay que hacer es un enlace del archivo objeto:

Compilamos:

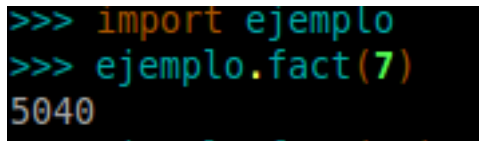
```
gcc -c ejemplo.c ejemplo_wrap.c -I /usr/include/python2.7/
```

Enlazamos los archivos .o para crear el archivo objeto de extensión .so, para no tener que volver a hacer estos pasos:

```
ld -shared ejemplo.o ejemplo_wrap.o -o _ejemplo.so
```

Con esto estamos listos para usar el módulo que hemos creado en C para utilizarlo en Python.

Ejemplo:

A screenshot of a terminal window with a black background and colored text. The first line shows a Python prompt followed by 'import ejemplo'. The second line shows a Python prompt followed by 'ejemplo.fact(7)'. The third line shows the output '5040'.