EBook Gratis

APRENDIZAJE

swig

Free unaffiliated eBook created from
Stack Overflow contributors.

#swig

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: swig

It is an unofficial and free swig ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official swig.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con un trago

Observaciones

SWIG (Simplified Wrapper and Interface Generator) es una herramienta para envolver códigos C y C ++ en una variedad de idiomas de destino, permitiendo que las API de C / C ++ se utilicen en otros idiomas.

SWIG analiza los archivos de encabezado y genera código de una manera que depende del idioma de destino. El desarrollador puede controlar la generación de código en el *archivo de interfaz SWIG* , así como a través de las opciones de la línea de comandos.

En el archivo de la interfaz, el desarrollador le dice a SWIG qué debe envolver y cómo. SWIG tiene su propio sistema de preprocesador y muchas directivas especiales para controlar cómo los datos, las clases y las funciones se envuelven en el idioma de destino. Algunas de estas directivas son generales y otras son específicas del idioma de destino.

Central a cómo funciona SWIG es el *mapa de tipos* . Los mapas de tipo son reglas que especifican cómo se calculan los tipos entre el código C y el idioma de destino. Los mapas de tipo se pueden aplicar globalmente a todo en el archivo de interfaz o localmente caso por caso. También se pueden personalizar si es necesario.

Una vez que se ejecuta SWIG en el archivo de interfaz, produce un archivo C o C ++ que es el contenedor. Este archivo debe compilarse y vincularse con el programa C / C ++ o la biblioteca estática con la que está diseñado el interfaz para generar una biblioteca compartida. Esa biblioteca a su vez es utilizada por el idioma de destino.

RTFM

No se puede enfatizar lo suficiente que SWIG ya viene con un excelente manual de documentación . Esto es muy detallado por una parte, cubre la instalación y presenta muchos ejemplos concretos en forma de fragmentos de código, incluido un ejemplo completo de "hola mundo" SWIG .

Pero lo más importante, también explica 1.7 Cómo evitar leer el manual :

> Si no te gusta leer manuales, echa un vistazo a la " Introducción " que contiene algunos ejemplos simples. Estos ejemplos contienen aproximadamente el 95% de todo lo que necesita saber para usar SWIG. Después de eso, simplemente use los capítulos específicos del idioma como referencia. La distribución SWIG también viene con un gran directorio de ejemplos que ilustran diferentes temas.

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar un trago.

Hola Mundo

Un ejemplo mínimo de usar SWIG.

HelloWorld.i , el archivo de interfaz SWIG

```
%module helloworld    //the name of the module SWIG will create
%{                    //code inside %{...%} gets inserted into the wrapper file
#include "myheader.h" //helloworld_wrap.cxx includes this header
%}

%include "myheader.h"   //include the header for SWIG to parse
```

Luego, en la línea de comando

```
swig -c++ -java HelloWorld.i
```

lo que significa que estamos envolviendo C ++ (en lugar de C) con Java como el idioma de destino especificado por HelloWorld.i. Esto producirá un archivo C ++, helloworld_wrap.cxx, que tiene el código de envoltorio. Este archivo debe compilarse y vincularse con cualquier código con el que se supone que la envoltura (por ejemplo, una biblioteca estática) creará una biblioteca compartida. Con algunos idiomas, como con Java en nuestro ejemplo, se generará un código adicional; en nuestro caso, habrá al menos un archivo de clase Java.

Lea Empezando con un trago en línea: https://riptutorial.com/es/swig/topic/7608/empezando-con-un-trago

Capítulo 2: Introducción a Typemaps

Introducción

Typemaps son el corazón de lo que hace SWIG. Cuando quiera pasar datos entre idiomas, los comportamientos para hacerlo dependen del tipo que SWIG vea. El poder de typemaps es que los trozos de código se aplican muchas veces.

El propio SWIG incluye muchos mapas de tipos útiles en la biblioteca central con la que se suministra, por ejemplo, para tipos primitivos, contenedores de bibliotecas estándar de C ++, etc., por lo que a menudo ni siquiera necesitará escribir ningún mapa de tipos para exponer su código. no significa completo

Sintaxis

- % typemap (NAME) TYPENAME% {CODE%}
- % typemap (NAME, OPTION = VALUE) TYPENAME% {CODE%}
- % typemap (NAME) TYPENAME VARIABLENAME% {CODE%}
- % typemap (NAME) TYPENAME (LOCALVARTYPE LOCALVARNAME)% {CODE%}

Parámetros

Parámetro	Detalles
NOMBRE	El nombre del mapa de tipo define su rol en la generación de un módulo. `in` y la `out` son comunes y se utilizan para la entrada (a las llamadas de función C ++ o C de Python / Java, etc.) y la salida (es decir, los valores de retorno de C o C ++ a Pyton / Java)
ESCRIBE UN NOMBRE	Cada mapa de tipos se aplica a uno o más tipos coincidentes. Estos deben ser enumerados aquí. Los calificadores de tipo (por ejemplo, const) son importantes.
CÓDIGO	Cada mapa de tipos debe convertir entre el tipo C o C ++ y el tipo correspondiente en el lenguaje ajustado. Tendrá que escribir código para hacer que en sus typemaps personalizados, por lo general haciendo uso de variables especiales que consiguen entra al juego. Ej `$input` dentro de una `in` typemap representa el valor de Python / Java, `$result` en una `out` typemap es lo bienestar regresó a Python / Java. `$1` en ambos tipos de mapas de entrada / salida representa la variable C o C ++ del tipo utilizado para coincidir con el mapa de tipos. Así que para `in` typemaps debe asignar a `$1` y para `out` usted leer de él. (Vea los mapas de tipos de argumentos múltiples para obtener más detalles sobre por qué es un número)

Parámetro	Detalles
NOMBRE DE LA VARIABLE	Los mapas de tipos se combinan de lo más específico a lo menos específico en general. Puede definir un mapa de tipo que solo haga coincidir los argumentos de la función con nombres específicos utilizando este formulario opcional. (Ver mapa de coincidencias para más detalles)
(LOCALVARTYPE LOCALVARNAME)	A veces, para particularly `in` typemaps es útil poder para declarar variables locales adicionales para sostener objetos en torno a una llamada. Esta sintaxis opcional nos permite hacer eso. Al usar esta sintaxis en lugar de escribirla en `CODE` se modifica el alcance, pero lo más importante es que SWIG cambia su nombre automáticamente para evitar choques si una función tiene dos argumentos que usan el mismo mapa de tipos.
OPCION = VALOR	El comportamiento de algunos mapas de tipos puede verse influido por la configuración de opciones adicionales utilizando esta sintaxis. Por ejemplo, se puede hacer que un mapa `in` tipos no tome ninguna entrada del idioma de destino configurando `numinputs=0` , en cuyo caso se espera que el mapa de caracteres llene la entrada de forma implícita. (Un caso común para esto podría ser establecer algo en `NULL` o llenarlo desde un valor global)

Examples

Mapa de tipografía básico - Python

Dado el siguiente tipo booleano personalizado que queremos envolver:

```
typedef char MYBOOL;
#define TRUE 1
#define FALSE 0
```

Un enfoque simple podría ser escribir los siguientes mapas de tipos en nuestra interfaz SWIG:

```
%typemap(in) MYBOOL %{
  // $input is what we got passed from Python for this function argument
  $1 = PyObject_IsTrue($input);
  // $1 is what will be used for the C or C++ call and we are responsible for setting it
%}

%typemap(out) MYBOOL %{
  // $1 is what we got from our C or C++ call
  $result = PyBool_FromLong($1);
  // $result is what gets given back to Python and we are responsible for setting it
%}
```

Con estos mapas de tipo, SWIG insertará nuestro código en el envoltorio generado cada vez que

vea que un MYBOOL pasa o sale de una llamada de función.

Lea Introducción a Typemaps en línea: https://riptutorial.com/es/swig/topic/9289/introduccion-a-typemaps

Creditos

S. No	Capítulos	Contributors
1	Empezando con un trago	BenK, Community, m7thon
2	Introducción a Typemaps	Flexo