

Python and other languages

- Python is a really useful scripting language
 - plenty of modules
- sometimes it is necessary to interface Python to C, C++ ...
 - for games the obvious reason is for speed enhancement
 - however it maybe to link against legacy libraries
 - consider in the future it might be possible to employ Python scripted AI bots in one of the Free Software Quake3 projects

Writing C code to be used by Python

- writing a C module for Python, used to be viewed as complex
 - now the [swig project](http://www.swig.org) (<http://www.swig.org>) seems to have changed that!

Advantages of SWIG

- taken from the excellent [SWIG documentation](http://www.swig.org/Doc1.3/Introduction.htm) (<http://www.swig.org/Doc1.3/Introduction.htm>)
- writing a user interface is painful in a compiled language
- testing is time consuming (the compile/debug cycle)
- not easy to reconfigure or customise without recompilation
- modularisation can be tricky
- security concerns (buffer overflow for instance)

Advantages of SWIG

- until recently the prevailing wisdom was for different languages for different tasks
 - the complete task
 - meant that edges of certain tasks were difficult to achieve
 - but the complexity was a price worth paying for the overall goodness of the rest of the project
 - very few projects ever mix languages
- example, quake3 and its scripting
 - the scripting compiles into C and is compiled at game start up and dynamically linked at start up

Advantages of SWIG

- example GCC uses a lisp set of rules which are converted into C during compilation
- in both cases the projects have ventured into another language - yet convert the lisp or quake scripts into C
 - most other projects use a single language
- SWIG offers to remove these barriers
- a lot of people use SWIG because they want to break out of the traditional monolithic C programming model

Advantages of mixing languages via SWIG

- incorporating C/C++ into a scripting language often results in a more modular design, less code, better flexibility, and increased programmer productivity
- SWIG tries to make the problem of C/C++ integration as painless as possible
- allows you to focus on the underlying C program and using the high-level language interface, but not the tedious and complex chore of making the two languages talk to each other

Simple C example

```

/* File : example.c */

double My_variable = 3.0;

/* Compute factorial of n */
int fact(int n) {
    if (n <= 1) return 1;
    else return n*fact(n-1);
}

/* Compute n mod m */
int my_mod(int n, int m) {
    return(n % m);
}

```

- you can download this file [here](#) (example.c)

SWIG interface file

```

/* File : example.i taken from the swig documentation */
%module example
%{
    /* Put headers and other declarations here */
    extern double My_variable;
    extern int fact(int);
    extern int my_mod(int n, int m);
}%

extern double My_variable;
extern int fact(int);
extern int my_mod(int n, int m);

```

- you can download this file [here](#) (example.i)

Building the module

- firstly check the Python version you are using

```
$ python
Python 2.4.4 (#2, Oct 19 2006, 23:03:48)
[GCC 4.1.2 20061007 (prerelease) (Debian 4.1.1-16)] on li:
Type "help", "copyright", "credits" or "license" for more
>>>
```

- now use the following command line actions to generate the shared library: `_example.so`

Building the module

```
$ swig -python example.i
$ gcc -c -fpic example.c example_wrap.c -I/usr/include/py
$ gcc -shared example.o example_wrap.o -o _example.so
```

- obviously you may need to modify `/usr/include/python2.4` to another value depending upon which version of Python you have installed

Testing the module

```
$ python
Python 2.4.4 (#2, Oct 19 2006, 23:03:48)
[GCC 4.1.2 20061007 (prerelease) (Debian 4.1.1-16)] on li:
Type "help", "copyright", "credits" or "license" for more
>>> import example
>>> print example.fact(4)
24
>>> print example.my_mod(30,7)
2
>>>
```

SWIG Disadvantages

- converting `char **` into a list of strings is possible - but you have to be very careful and the interface file becomes much more complex
- [further reading](http://www.swig.org/Doc1.3/Python.html#Python_nn2) `<http://www.swig.org/Doc1.3/Python.html#Python_nn2>`