39-interfacing-with-C

April 29, 2016

1 Interfacing Python and C/C++

- doc

2 calling C/C++ from Python

- also known as FFI, Foreign Function Interface
- works fine, but mistakes in C can corrupt the Python environment, causing mysteries and crashes
- will show examples of calling 'libc' functions, which 'everything' uses
- to call your own C code, build a shared library and load it

```
In [1]: from ctypes import *
```

```
In [2]: # Load the standard C library - full of routines all programs use
        # On linux, this call would be
        # libc = cdll.LoadLibrary("libc.so")
        # call below works on a mac

        lc = cdll.LoadLibrary("libc.dylib")
        lc
```

```
Out[2]: <CDLL 'libc.dylib', handle 7fff6a60b9f8 at 0x105e38f60>
```

```
In [3]: # now have access to everything in the library,
        # but takes some effort to call things correctly

        [lc.strcmp, lc.printf, lc.malloc, lc.sin, lc.time]
```

```
Out[3]: [<_FuncPtr object at 0x105e25430>,
         <_FuncPtr object at 0x105e254f8>,
         <_FuncPtr object at 0x105e255c0>,
         <_FuncPtr object at 0x105e25688>,
         <_FuncPtr object at 0x105e25750>]
```

```
In [4]: # None means no args
        # seconds since 1970

        lc.time(None)
```

```
Out[4]: 1461958920
```

3 Call sin in libc

- sin takes and returns doubles ‘‘‘ NAME sin – sine function

SYNOPSIS #include

```
double
sin(double x);

long double
sinl(long double x);

float
sinf(float x);
```

‘‘‘

```
In [5]: # ultimately calls libc sin routine

        import math
        math.sin(.5)

Out[5]: 0.479425538604203
```

```
In [6]: # get libc.sin function pointer

        s = lc.sin
        s

Out[6]: <_FuncPtr object at 0x105e25688>
```

```
In [7]: # this won't work

        s(.5)


        ---------------------------------------------------------------------------

        ArgumentError                             Traceback (most recent call last)

        <ipython-input-7-530f99118192> in <module>()
          1 # this won't work
          2
    ----> 3 s(.5)


        ArgumentError: argument 1: <class 'TypeError'>: Don't know how to convert parameter 1
```

```
In [8]: # have to convert Python 'float' into C 'double'
        # but it still won't work...garbage result

        s(c_double(.5))

Out[8]: 1022
```

```
In [9]: # ...have to specify how to convert C return type back into float

        s.restype = c_double
        s(c_double(.5))
```

```
Out[9]: 0.479425538604203
```

```
In [10]: # looks like same routine is being called

         s(c_double(.5)) - math.sin(.5)
```

```
Out[10]: 0.0
```

```
In [ ]: # Can define callbacks in python
        # this makes an integer C array class

        IntArray5 = c_int * 5

        # make array object
        ia = IntArray5(5, 1, 7, 33, 99)
        qsort = lc.qsort
        qsort.restype = None

        # write the comparsion function in Python

        def qsortCmp(a, b):
            print("qsortCmp", a[0], b[0] )
            return a[0] - b[0]

        # declaration for comparison function
        CMPFUNC = CFUNCTYPE(c_int, POINTER(c_int), POINTER(c_int))

        qsort(ia, len(ia), sizeof(c_int), CMPFUNC(qsortCmp))
```

```
In [ ]: # list has been sorted by libc.qsort

        list(ia)
```

4 struct - lays out fields like C 'struct' would

- hardware interfaces often need precise byte layouts
- does padding like a C struct would
- doc

```
In [ ]: from struct import *
```

```
In [ ]: # 2 ints and a byte - why is len(p) 12 bytes instead of 9?
        # f is a format spec - what types of things are going in the struct?

        f = 'ici'
        p = pack(f, 2,b'X', 3)
        [p, len(p), unpack(f, p)]
```

```
In [ ]: f = 'ihi'
        p = pack(f, 4, 5, 6)
        [p, len(p), unpack(f, p)]
```

```
In [ ]: list(map(type, unpack(f, p)))
```

5 Embedding Python In a C/C++ application

- can be incredibly useful
- not too hard, but not trivial
- mostly consists of converting C and Python data types back and forth
- [doc]https://docs.python.org/3.5/extending/index.html

6 Example - Blender

- Blender is an open source animation system
- Pasting and running running the code below modifies the position of one vertex in the default cube
- Pretty much every operation in the GUI is available in the Python API
  - you can see the function in the tool tips
- allows programs to build 3D objects and automate animations
- zoom with cntl-two-fingers

```
import bpy
bpy.data.objects["Cube"].data.vertices[0].co.x += 1.0
```