Mixing C and Python

Comp-206 : Introduction to Software Systems
Lecture 20

Alexandre Denault
Computer Science
McGill University
Fall 2006

Programming Strategy

- When building your C function that will be called from Python, you want to follow these step
  - Take the Python input and transform it into C variable.
  - Do whatever manipulation/calculation you need to do.
  - Transform the output into Python objects and return them.

Building a Python module

- To build a Python module in C, three things a needed:
  - An initialization function.
  - A list of methods you want accessible.
  - An implementation of those methods.

List of methods

- The list of methods to be accessible must be stored as an array of declaration.
- Each of these declaration include four elements:
  - The name of the function in python.
  - The corresponding C function.
  - A constant denoting how arguments should be passed.
  - An string describing what the function does.

```
static PyMethodDef CPythonMethods[] = {
    {"execute",  cpython_execute, METH_VARARGS,
     "Execute generic demo"},
    {NULL, NULL, 0, NULL}        /* Sentinel */
};
```

- The array of declaration must be terminated by a sentinel (an empty declaration).

Init Function

- This function initializes the module and registers it (and its functions) with the Python interpreter.

```
PyMODINIT_FUNC initcpython(void)
{
    (void) Py_InitModule("cpython",
  CPythonMethods);
}
```

- To simplify things, the name of the module and the filename of the library should be identical.

Function itself

- The functions should have the following signature :

```
static PyObject *
cpython_execute(PyObject *self, PyObject *args)
```

  - Every function returns one Python object.
  - The reference to the object that called the function is found in self.
  - The arguments of the function are passed as a tuple.

Interpreting the Arguments

- The C/Python API provides a function to easily parse the arguments from a function.

```
int i;
double d;
const char * s;

PyObject * outObj;

if (!PyArg_ParseTuple(args, "ids", &i, &d, &s))
      return NULL;
```

- The method PyArg_ParseTuple functions a lot like scanf
  - Except for string where storage is already allocated.
- The interesting part is the format string.

Format String

- "s" (string) [char *]
- "s#" (string) [char *, int]
- "b" (integer) [char]
- "i" (integer) [int]
- "c" (string of length 1) [char]
- "f" (float) [float]
- "d" (float) [double]
- "O" (object) [PyObject *]

Returning multiple values

- In Python, you can return multiple value from a function using a tuple :

```
PyObject * outObj;
outObj = PyTuple_New(6);
PyTuple_SetItem(outObj, 0, PyInt_FromLong(i) );
PyTuple_SetItem(outObj, 1, PyFloat_FromDouble(d) );
PyTuple_SetItem(outObj, 2, PyString_FromString(s) );
PyTuple_SetItem(outObj, 3, Py_BuildValue("i", i) );
PyTuple_SetItem(outObj, 4, Py_BuildValue("d", d) );
PyTuple_SetItem(outObj, 5, Py_BuildValue("s", s) );
return outObj;
```

Shared vs Static Modules

- Shared modules are compiled seperatly and are designed to be loaded dynamically when needed.
- Static modules are compiled inside the executable.
- Can you give me advantages to each strategy?

Gcc Arguments

- The following will compile the object code to the module (cpython.o).

```
gcc -I /usr/include/python2.4 -c cpython.c
```

  - Note the addition of an include directory.

- The following will compile the object code into a shared module (cpython.so)

```
gcc -shared cpython.o -o cpython.so
```

Python and Images

- Manipulating images is much easier in Python than in C.
- For our assignment, we will use the Python Image Library (PIL).

Simple PIL Example

```
import Image

sizeImage = (640, 480)
img = Image.new("L", sizeImage)

putpixel = img.im.putpixel

for x in range(sizeImage[0]):
   for y in range(sizeImage[1]):
      putpixel((x, y), (255,0,0))

img.save(“file.png”, "PNG")
```

What's ahead?

- Function Pointers
- Process/Signals
- Fork
- Interprocess Communication (Pipes)
- Threads
- Sockets

Mercury

- Course evaluations are one of the most important communication means for students to provide constructive feedback to their instructors. Course evaluation results are also used by the Faculty when making decisions concerning tenure and promotions.
- Accessing Mercury
    1. Log in to Minerva for students
    2. Student Menu > Mercury – McGill Online Evaluations

Reminders

- During the main evaluation period, students will receive
  - 4 automatic email reminders
  - a pop-up window when logging into MINERVA and WebCT Vista
  - regular reminders from me

Privacy

Evaluation results for instructors will be available once ALL the final grades for ALL of their courses in a term have been submitted.