扩展/嵌入常见问题

目录

可以使用 C 语言创建自己的函数吗?

是的，您可以在C中创建包含函数、变量、异常甚至新类型的内置模块。在文档 扩展和嵌入 Python 解释器 中有说明。

大多数中级或高级的Python书籍也涵盖这个主题。

可以使用 C++ 语言创建自己的函数吗?

是的，可以使用 C++ 中兼容 C 的功能。 在 Python include 文件周围放置 `extern "C" { ... }`，并在Python解释器调用的每个函数之前放置 `extern "C"`。 具有构造函数的全局或静态 C++ 对象可能不是一个好主意。

C很难写，有没有其他选择?

有多个替代选择可用来编写你自己的 C 扩展，具体取决于你想要做什么。 推荐的第三方工具 提供了更简单或更复杂的方式来为 Python 创建 C 和 C++ 扩展。

如何在 C 中执行任意 Python 语句?

执行此操作的最高层级函数为 `PyRun_SimpleString()`，它接受单个字符串参数用于在模块 `__main__` 的上下文中执行并在成功时返回 `0` 而在发生异常 (包括 `SyntaxError`) 时返回 `-1`。 如果你想要更多可控性，可以使用 `PyRun_String()`；请在 `Python/pythonrun.c` 中查看 `PyRun_SimpleString()` 的源码。

如何在 C 中对任意 Python 表达式求值?

可以调用前一问题中介绍的函数 `PyRun_String()` 并附带起始标记符 `Py_eval_input`；它会解析表达式，对其求值并返回结果值。

如何从Python对象中提取C的值?

这取决于对象的类型。 如果是元组，`PyTuple_Size()` 将返回其长度而 `PyTuple_GetItem()` 将返回指定索引号上的项。 对于列表也有类似的函数 `PyList_Size()` 和 `PyList_GetItem()`。

对于字节串，`PyBytes_Size()` 将返回其长度而 `PyBytes_AsStringAndSize()` 将提供一个指向其值和长度的指针。 请注意 Python 字节串对象可能包含空字节因此不应使用 C 的 `strlen()`。

要检测一个对象的类型，首先要确保它不为 `NULL`，然后使用 `PyBytes_Check()`, `PyTuple_Check()`, `PyList_Check()` 等等。

还有一个针对 Python 对象的高层级 API，通过所谓的'抽象'接口提供 —— 请参阅 `Include/abstract.h` 了解详情。 它允许使用 `PySequence_Length()`, `PySequence_GetItem()` 这样的调用来与任意种类的 Python 序列进行对接，此外还可使用许多其他有用的协议例如数字 (`PyNumber_Index()` 等) 以及 PyMapping API 中的各种映射等等。

如何使用Py_BuildValue()创建任意长度的元组?

不可以。应该使用 `PyTuple_Pack()` 。

如何从C调用对象的方法?

可以使用 `PyObject_CallMethod()` 函数来调用某个对象的任意方法。 形参为该对象、要调用的方法名、类似 `Py_BuildValue()` 所用的格式字符串以及要传给方法的参数值:

```
PyObject *
PyObject_CallMethod(PyObject *object, const char *method_name,
                    const char *arg_format, ...);
```

这适用于任何具有方法的对象 —— 不论是内置方法还是用户自定义方法。 你需要负责对返回值进行最终的 `Py_DECREF()` 处理。

例如调用某个文件对象的 "seek" 方法并传入参数 10, 0 (假定文件对象的指针为 "f"):

```
res = PyObject_CallMethod(f, "seek", "(ii)", 10, 0);
if (res == NULL) {
        ... an exception occurred ...
}
else {
        Py_DECREF(res);
}
```

请注意由于 `PyObject_CallObject()` *总是* 接受一个元组作为参数列表，要调用不带参数的函数，则传入格式为 "()"，要调用只带一个参数的函数，则应将参数包含于圆括号中，例如 "(i)"。

如何捕获PyErr_Print()（或打印到stdout / stderr的任何内容）的输出?

在 Python 代码中，定义一个支持 `write()` 方法的对象。 将此对象赋值给 `sys.stdout` 和 `sys.stderr`。 调用 print_error 或者只是允许标准回溯机制生效。 在此之后，输出将转往你的 `write()` 方法所指向的任何地方。

做到这一点的最简单方式是使用 `io.StringIO` 类:

```
>>> import io, sys
>>> sys.stdout = io.StringIO()
>>> print('foo')
>>> print('hello world!')
>>> sys.stderr.write(sys.stdout.getvalue())
foo
hello world!
```

实现同样效果的自定义对象看起来是这样的:

```
>>> import io, sys
>>> class StdoutCatcher(io.TextIOBase):
...     def __init__(self):
...         self.data = []
...     def write(self, stuff):
...         self.data.append(stuff)
...
>>> import sys
>>> sys.stdout = StdoutCatcher()
>>> print('foo')
>>> print('hello world!')
>>> sys.stderr.write(''.join(sys.stdout.data))
foo
hello world!
```

如何从C访问用Python编写的模块?

你可以通过如下方式获得一个指向模块对象的指针:

```
module = PyImport_ImportModule("<modulename>");
```

如果模块尚未被导入（即它还不存在于 sys.modules 中），这会初始化该模块；否则它只是简单地返回 sys.modules["<modulename>"] 的值。 请注意它并不会将模块加入任何命名空间 —— 它只是确保模块被初始化并存在于 sys.modules 中。

之后你就可以通过如下方式来访问模块的属性（即模块中定义的任何名称）：

```
attr = PyObject_GetAttrString(module, "<attrname>");
```

调用 PyObject_SetAttrString() 为模块中的变量赋值也是可以的。

如何在 Python 中对接 C ++ 对象?

根据你的需求，可以选择许多方式。 手动的实现方式请查阅 "扩展与嵌入" 文档 来入门。 需要知道的是对于 Python 运行时系统来说，C 和 C++ 并不没有太大的区别 —— 因此围绕一个 C 结构（指针）类型构建新 Python 对象的策略同样适用于 C++ 对象。

有关C ++库，请参阅 C很难写，有没有其他选择?

我使用Setup文件添加了一个模块，为什么make失败了?

安装程序必须以换行符结束，如果没有换行符，则构建过程将失败。 （修复这个需要一些丑陋的shell脚本编程，而且这个bug很小，看起来不值得花这么大力气。）

如何调试扩展?

将GDB与动态加载的扩展名一起使用时，在加载扩展名之前，不能在扩展名中设置断点。

在您的 .gdbinit 文件中（或交互式）添加命令：

```
br _PyImport_LoadDynamicModule
```

然后运行GDB：

```
$ gdb /local/bin/python
gdb) run myscript.py
gdb) continue # 重复直到你的扩展被载入
gdb) finish   # 你的扩展已被载入
gdb) br myfunction.c:50
gdb) continue
```

我想在Linux系统上编译一个Python模块，但是缺少一些文件。为什么?

大多数打包的 Python 版本都会省略一些编译 Python 扩展所必需的文件。

对于 Red Hat，请安装 python3-devel RPM 来获取必需的文件。

对于 Debian，请运行 apt-get install python3-dev。

如何区分“输入不完整”和“输入无效”？

有时，希望模仿Python交互式解释器的行为，在输入不完整时(例如，您键入了“if”语句的开头，或者没有关闭括号或三个字符串引号)，给出一个延续提示，但当输入无效时，立即给出一条语法错误消息。

在Python中，您可以使用 `codeop` 模块，该模块非常接近解析器的行为。例如，IDLE就使用了这个。

在C中执行此操作的最简单方法是调用 `PyRun_InteractiveLoop()` （可能在单独的线程中）并让Python解释器为您处理输入。您还可以设置 `PyOS_ReadlineFunctionPointer()` 指向您的自定义输入函数。有关更多提示，请参阅 `Modules/readline.c` 和 `Parser/myreadline.c` 。

如何找到未定义的g++符号__builtin_new或__pure_virtual？

要动态加载g ++扩展模块，必须重新编译Python，要使用g ++重新链接（在Python Modules Makefile中更改LINKCC），及链接扩展模块（例如： `g++ -shared -o mymodule.so mymodule.o` ）。

能否创建一个对象类，其中部分方法在C中实现，而其他方法在Python中实现（例如通过继承）？

是的，您可以继承内置类，例如 `int` ， `list` ， `dict` 等。

Boost Python Library (BPL, https://www.boost.org/libs/python/doc/index.html) 提供了一种从 C++ 执行此操作的方式（即你可以使用 BPL 来继承用 C++ 编写的扩展类）。