Python与C/C++模块交互

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 修订者 | 修订描述 |
| 2017.12.08 | v1 | 林 清 | 初稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1.1 介绍 2](#_Toc497911959)

# 扩展Python

通过Python和C/C++交互能够完成以下两件事：

* 实现新的内置对象类型；
* 可以调用C库函数和系统调用；

C扩展接口是CPython专用的，其他版本的Python不能通过C扩展接口和C语言交互。为了保证移植性，应当考虑使用ctypes或者cffi模块实现同样功能。

## A Simple Example

下面的案例实现一个简单的模块spam，通过该模块可以进行如下调用：

import spam

status = spam.system("ls -l")

system函数的入参是一个以null结尾的字符串，返回值是一个integer。

spam对应的C文件的文件名为spammodule.c或者 spam.c，并且引入Python.h这个头文件（因为Python可能会定义一些影响某些系统上标准头文件的预处理器定义，所以在包含任何标准头文件之前，必须包含Python.h）。

Python.h定义的所有用户可见符号都有Py或PY的前缀，并且包含一些标准的头文件：<stdio.h>，<string.h>，<errno.h>和<stdlib.h>。Centos中安装了python-devel之后在/usr/include/python2.7/目录能找到该文件。

**static** PyObject \*

spam\_system(PyObject \*self, PyObject \*args)

{

**const** char \*command;

int sts;

**if** (!PyArg\_ParseTuple(args, "s", &command))

**return** NULL;

sts = system(command);

**return** Py\_BuildValue("i", sts);

}

从Python中的参数列表到传递给C函数时，C函数总是有两个参数，为self和args。

self参数：对于模块中的普通函数，为Null或者模块初始化时指向该模块的指针？对于成员函数，指向对应的变量实例。PyArg\_ParseTuple将对应指针指向转换后的值，并且返回True。

*args*参数：指向包含参数的Python元组对象的指针。Python API中的函数PyArg\_ParseTuple检查参数类型并将其转换为C值。它使用模板字符串来确定参数的Python类型以及存储转换值的C类型。

## 异常控制

在Python解释器中，当一个函数失败时，会存储三个变量（对应到Python中就是sys.exc\_type, sys.exc\_value and sys.exc\_traceback）。

# 参考

https://docs.python.org/2/extending/extending.html