**"파이썬에서 C모듈을 이용하기"**

<http://batsu05.egloos.com/839472>

파이썬은 그 자체가 C로 작성되었다.

이런 맥락에서 생각해보면 파이썬에서 C모듈을 사용할수도 있을뿐 아니라 C에서 파이썬 모듈을 사용할수도 있을것 같다. -> 이말은 사실이다.

본론으로 들어가기전에 파일의 확장자에 대해 살포시 살펴보기로 하자.

파일의 확장자는 그 파일의 속성과 종류를 손쉽게 알려주는 역할을 한다. (이는 어떻게 보면 있어도 되고 없어도 되는 존재이며, 관습적으로 사용하게 된다.)

우리가 흔히 알고 있는 확장자로는 'exe', 'txt', 'jpg', 'mp3' 등 수없이 많다.

- lib : 정적 라이브러리 파일, 일반적으로 C, C++ 라이브러리다.

- dll : 동적 라이브러리 파일

- pyd : 파이썬이 사용하는 C 라이브러리

여기서는 C로 작성된 코드를 이용하여 pyd를 만들고 이 pyd를 이용해서 C 모듈을 이용하는 방법에 대해서 살펴본다.

**1. 파이썬 코드에서 import를 하면 일어나는 일**

파이썬에서 특정한 모듈을 사용하기 위해 import문을 이용하여 모듈을 지정한다. 파이썬이 사용하고 로드하는 모듈은 아래의 순서로 찾고 그와 일치하면 모듈을 초기화한 후에 지역 이름공간에 이름을 정의한다.

    - 기본적으로 탑재되어 있는 모듈들 : ex)socket, time 등

    - C로 만들어진 dll 파일(윈도에서는 pyd 파일)

    - 자체 제작한 파이썬 모듈

그러면 파이썬에서 import는 어떤 매커니즘으로 동작하는가? 이것을 함 알아보자.

import mylib

위의 코드를 수행할때 파이썬은 mylib 모듈을 로드하기 위해 다음의 절차에 따라 검색한다.

    1) sys.modules 에 등록되어 있는지 확인한다. 등록되어 있으면 로드한다.

    2) 1)에서 존재하지 않으면 sys.path의 디렉토리를 검색하면서 mylib 모듈을 찾는다.

다시 한번 요약하면, 외부 모듈을 import 하게되면 파이썬은

    1) 모듈이름과 일치하는 모듈을 찾는다.

    2) 모듈을 초기화 한다.

    3) 모듈이름을 지역이름공간에 할당한다.

**2. 파이썬에서 이용할 C 모듈 만들기**

여기서 우리는 우리가 만든 C 모듈을 파이썬에서 사용하고 싶다. 즉,

import (우리가 만든 C 모듈) 하고 싶다 이 말이다.

우리가 C로 만든 모듈을 파이썬에서 이용하고자 할때 역시 위에서 서술한 import 절차에 따라 모듈을 찾고 로드하게 된다.

우리가 유의해야 할 곳은 바로 모듈을 초기화 하는 부분이다. 나머지 절차는 파이썬이 다 알아서 해준다.

파이썬에서 사용하는 C모듈은 리눅스에서는 .so 확장자 형태로, 윈도에서는 .pyd 확장자 형태로 빌드되어 파이썬 패키지 디렉토리의 Lib/site-package 디렉로리로 복사되므로서 사용가능하게 된다.

    1) C로 소스코드를 작성한다.

    2) 작성한 C 소스코드를 컴파일하여 파이썬이 이용가능한 .so (리눅스에서) 및 .pyd(윈도용) 파일로 빌드한다.

    3) .so 및 .pyd 파일을 파이썬 디렉토리의 Lib/site-package로 복사한다.

그러면 하나의 실제 C모듈을 만들어 적용하므로서 설명해보자.

파이썬이 C 모듈을 import 하게되면 초기화를 위해 "init+모듈이름" 으로 된 함수를 호출한다. 모듈이름이 mylib 이면 initmylib()을 호출한다는 말이다.

그러므로 우리의 C 소스에는 위에서 설명한 초기화를 위한 함수 init+모듈이름 으로 된 함수가 존재해야만 한다.

예제로 만들 모듈은 입력한 문자열을 pylog.txt 라는 파일에 기록하는 모듈이다. 주석을 달았으므로 주석을 참조하도록 한다.

// mylib.c

#include "Python.h"  
#include <stdio.h>

static PyObject \*ErrorObject;

// 실제 동작하는 함수

static PyObject\* write\_log(PyObject \*self, PyObject \*args) // 인자는 이와같이 고정된다.  
{  
    char\* msg;  
    FILE \*fp;  
      
    if(!PyArg\_ParseTuple(args, "s", &msg))  
        return NULL;  
      
    fp = fopen("c:\\pylog.txt", "wt+");  
    fprintf(fp, msg);

    fclose(fp);  
      
    return Py\_BuildValue("i", 0);      
}

/\* methods 구조체 배열에 지정되는 정보는 {"실제사용할 메쏘드명", 메쏘드명에 대응하는 실제 동작하는 함수명, 인자 종류} \*/

static struct PyMethodDef methods[] =  
{  
    {"wlog", write\_log, METH\_VARARGS},  
    {NULL, NULL}  
};

//

void initmylib()  
{  
    PyObject\* m;

   // Py\_InitModule("모듈명", 이모듈에 적용된 메쏘드들을 담을 구조체배열 포인터)  
    m = Py\_InitModule("mylib", methods);

    ErrorObject = Py\_BuildValue("s", "error");  
}

위의 C 코드를 컴파일하기 위해 setup.py 함수를 작성해야 한다.

# setup.py

from distutils.core import setup, Extension

setup(name = "mylib",  
        version = "1.0",  
        description = "print log",  
        author = "Samsjang",  
        author\_email = "[samsjang@cdnetworks.co.kr](mailto:samsjang@cdnetworks.co.kr)",  
        url = "[http://www.cdnetworks.co.kr](http://www.cdnetworks.co.kr/)",  
        ext\_modules = [Extension("mylib", ["mylib.c"])]  
        )

위의 소스에서 ext\_modules에는 적용할 모듈명과 그 모듈의 구현코드를 기록한다. 나머지 부분의 기타 정보를 기록해둔 메타정보에 불과하다.

리눅스 및 윈도에서 아래와 같이 빌드하고 설치한다.

**python setup.py install**

위의 명령은 리눅스에서는 gcc를 호출하며, 윈도에서는 visual C++ 컴파일러가 구동되면서 빌드한 후 파이썬 디렉토리의 Lib/site-packages 디렉토리에 mylib.so 또는 mylib.pyd  파일이 생성된다.

자 그럼 C모듈인 mylib 모듈을 실제 파이썬 코드에서 적용해보자.

# test.py

import mylib

if \_\_name\_\_ == '\_\_main\_\_':      
    mylib.wlog('I love U')

위의 결과는 C:\\pylog.txt 파일에 "I love U" 라는 글을 남긴다.