15.18 传递已打开的文件给C扩展¶

问题¶

你在Python中有一个打开的文件对象，但是需要将它传给要使用这个文件的C扩展。

解决方案¶

要将一个文件转换为一个整型的文件描述符，使用 `PyFile_FromFd()` ，如下：

```
PyObject *fobj;     /* File object (already obtained somehow) */
int fd = PyObject_AsFileDescriptor(fobj);
if (fd < 0) {
   return NULL;
}
```

结果文件描述符是通过调用 `fobj` 中的 `fileno()` 方法获得的。 因此，任何以这种方式暴露给一个描述器的对象都适用（比如文件、套接字等）。 一旦你有了这个描述器，它就能被传递给多个低级的可处理文件的C函数。

如果你需要转换一个整型文件描述符为一个Python对象，适用下面的 `PyFile_FromFd()` ：

```
int fd;     /* Existing file descriptor (already open) */
PyObject *fobj = PyFile_FromFd(fd, "filename","r",-1,NULL,NULL,NULL,1);
```

`PyFile_FromFd()` 的参数对应内置的 `open()` 函数。 NULL表示编码、错误和换行参数使用默认值。

讨论¶

如果将Python中的文件对象传给C，有一些注意事项。 首先，Python通过 `io` 模块执行自己的I/O缓冲。 在传递任何类型的文件描述符给C之前，你都要首先在相应文件对象上刷新I/O缓冲。 不然的话，你会打乱文件系统上面的数据。

其次，你需要特别注意文件的归属者以及关闭文件的职责。 如果一个文件描述符被传给C，但是在Python中还在被使用着，你需要确保C没有意外的关闭它。 类似的，如果一个文件描述符被转换为一个Python文件对象，你需要清楚谁应该去关闭它。 `PyFile_FromFd()` 的最后一个参数被设置成1，用来指出Python应该关闭这个文件。

如果你需要从C标准I/O库中使用如 `fdopen()` 函数来创建不同类型的文件对象比如 `FILE *` 对象， 你需要特别小心了。这样做会在I/O堆栈中产生两个完全不同的I/O缓冲层 （一个是来自Python的 `io` 模块，另一个来自C的 `stdio` ）。像C中的 `fclose()` 会关闭Python要使用的文件。 如果让你选的话，你应该会选择去构建一个扩展代码来处理底层的整型文件描述符， 而不是使用来自<stdio.h>的高层抽象功能。