# 5.16 增加或改变已打开文件的编码¶

## 问题¶

你想在不关闭一个已打开的文件前提下增加或改变它的Unicode编码。

## 解决方案¶

如果你想给一个以二进制模式打开的文件添加Unicode编码/解码方式， 可以使用 `io.TextIOWrapper()` 对象包装它。比如：

```
import urllib.request
import io

u = urllib.request.urlopen('http://www.python.org')
f = io.TextIOWrapper(u, encoding='utf-8')
text = f.read()
```

如果你想修改一个已经打开的文本模式的文件的编码方式，可以先使用 `detach()` 方法移除掉已存在的文本编码层，并使用新的编码方式代替。下面是一个在 `sys.stdout` 上修改编码方式的例子：

```
>>> import sys
>>> sys.stdout.encoding
'UTF-8'
>>> sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding='latin-1')
>>> sys.stdout.encoding
'latin-1'
>>>
```

这样做可能会中断你的终端，这里仅仅是为了演示而已。

## 讨论¶

I/O系统由一系列的层次构建而成。你可以试着运行下面这个操作一个文本文件的例子来查看这种层次：

```
>>> f = open('sample.txt','w')
>>> f
<_io.TextIOWrapper name='sample.txt' mode='w' encoding='UTF-8'>
>>> f.buffer
<_io.BufferedWriter name='sample.txt'>
>>> f.buffer.raw
<_io.FileIO name='sample.txt' mode='wb'>
>>>
```

在这个例子中，`io.TextIOWrapper` 是一个编码和解码Unicode的文本处理层， `io.BufferedWriter` 是一个处理二进制数据的带缓冲的I/O层， `io.FileIO` 是一个表示操作系统底层文件描述符的原始文件。 增加或改变文本编码会涉及增加或改变最上面的 `io.TextIOWrapper` 层。

一般来讲，像上面例子这样通过访问属性值来直接操作不同的层是很不安全的。 例如，如果你试着使用下面这样的技术改变编码看看会发生什么：

```
>>> f
<_io.TextIOWrapper name='sample.txt' mode='w' encoding='UTF-8'>
>>> f = io.TextIOWrapper(f.buffer, encoding='latin-1')
>>> f
<_io.TextIOWrapper name='sample.txt' encoding='latin-1'>
>>> f.write('Hello')
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
>>>
```

结果出错了，因为f的原始值已经被破坏了并关闭了底层的文件。

`detach()` 方法会断开文件的最顶层并返回第二层，之后最顶层就没什么用了。例如：

```
>>> f = open('sample.txt', 'w')
>>> f
<_io.TextIOWrapper name='sample.txt' mode='w' encoding='UTF-8'>
>>> b = f.detach()
>>> b
<_io.BufferedWriter name='sample.txt'>
>>> f.write('hello')
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
ValueError: underlying buffer has been detached
>>>
```

一旦断开最顶层后，你就可以给返回结果添加一个新的最顶层。比如：

```
>>> f = io.TextIOWrapper(b, encoding='latin-1')
>>> f
<_io.TextIOWrapper name='sample.txt' encoding='latin-1'>
>>>
```

尽管已经向你演示了改变编码的方法， 但是你还可以利用这种技术来改变文件行处理、错误机制以及文件处理的其他方面。例如：

```
>>> sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding='ascii',
...                               errors='xmlcharrefreplace')
>>> print('Jalape\u00f1o')
Jalape&#241;o
>>>
```

注意下最后输出中的非ASCII字符 `ñ` 是如何被 `&#241;` 取代的。