# 5.10 内存映射的二进制文件¶

## 问题¶

你想内存映射一个二进制文件到一个可变字节数组中，目的可能是为了随机访问它的内容或者是原地做些修改。

## 解决方案¶

使用 `mmap` 模块来内存映射文件。 下面是一个工具函数，向你演示了如何打开一个文件并以一种便捷方式内存映射这个文件。

```
import os
import mmap

def memory_map(filename, access=mmap.ACCESS_WRITE):
    size = os.path.getsize(filename)
    fd = os.open(filename, os.O_RDWR)
    return mmap.mmap(fd, size, access=access)
```

为了使用这个函数，你需要有一个已创建并且内容不为空的文件。 下面是一个例子，教你怎样初始创建一个文件并将其内容扩充到指定大小：

```
>>> size = 1000000
>>> with open('data', 'wb') as f:
...     f.seek(size-1)
...     f.write(b'\x00')
...
>>>
```

下面是一个利用 `memory_map()` 函数类内存映射文件内容的例子：

```
>>> m = memory_map('data')
>>> len(m)
1000000
>>> m[0:10]
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
>>> m[0]
0
>>> # Reassign a slice
>>> m[0:11] = b'Hello World'
>>> m.close()

>>> # Verify that changes were made
>>> with open('data', 'rb') as f:
... print(f.read(11))
...
b'Hello World'
>>>
```

`mmap()` 返回的 `mmap` 对象同样也可以作为一个上下文管理器来使用， 这时候底层的文件会被自动关闭。比如：

```
>>> with memory_map('data') as m:
...     print(len(m))
...     print(m[0:10])
...
1000000
b'Hello World'
>>> m.closed
True
>>>
```

默认情况下， `memeory_map()` 函数打开的文件同时支持读和写操作。 任何的修改内容都会复制回原来的文件中。 如果需要只读的访问模式，可以给参数 `access` 赋值为 `mmap.ACCESS_READ` 。比如：

```
m = memory_map(filename, mmap.ACCESS_READ)
```

如果你想在本地修改数据，但是又不想将修改写回到原始文件中，可以使用 `mmap.ACCESS_COPY`：

```
m = memory_map(filename, mmap.ACCESS_COPY)
```

## 讨论¶

为了随机访问文件的内容，使用 `mmap` 将文件映射到内存中是一个高效和优雅的方法。 例如，你无需打开一个文件并执行大量的 `seek()`， `read()`， `write()` 调用， 只需要简单的映射文件并使用切片操作访问数据即可。

一般来讲， `mmap()` 所暴露的内存看上去就是一个二进制数组对象。 但是，你可以使用一个内存视图来解析其中的数据。比如：

```
>>> m = memory_map('data')
>>> # Memoryview of unsigned integers
>>> v = memoryview(m).cast('I')
>>> v[0] = 7
>>> m[0:4]
b'\x07\x00\x00\x00'
>>> m[0:4] = b'\x07\x01\x00\x00'
>>> v[0]
263
>>>
```

需要强调的一点是，内存映射一个文件并不会导致整个文件被读取到内存中。 也就是说，文件并没有被复制到内存缓存或数组中。相反，操作系统仅仅为文件内容保留了一段虚拟内存。 当你访问文件的不同区域时，这些区域的内容才根据需要被读取并映射到内存区域中。 而那些从没被访问到的部分还是留在磁盘上。所有这些过程是透明的，在幕后完成！

如果多个Python解释器内存映射同一个文件，得到的 `mmap` 对象能够被用来在解释器直接交换数据。 也就是说，所有解释器都能同时读写数据，并且其中一个解释器所做的修改会自动呈现在其他解释器中。 很明显，这里需要考虑同步的问题。但是这种方法有时候可以用来在管道或套接字间传递数据。

这一小节中函数尽量写得很通用，同时适用于Unix和Windows平台。 要注意的是使用 `mmap()` 函数时会在底层有一些平台的差异性。 另外，还有一些选项可以用来创建匿名的内存映射区域。 如果你对这个感兴趣，确保你仔细研读了Python文档中 这方面的内容 。